```
	int i,j;
	for(i=0;i<n;i++)
	{
		for(j=0;j<n-i-1;j++)
		{
			if(arr[j]>arr[j+1])
			{
				swap(&arr[j],&arr[j+1]);;
			}
		}
	}
}
int main()
{
	int i,choice;
	int a[10]={2,56,189,7,4,6,123,44,55,10};
	int n=sizeof(a)/sizeof(a[0]);
	printf("Given array is: \n");
	for(i=0;i<n;i++)
	{
		printf("%d\t",a[i]);
	}
```

```
printf("\nSelect any sorting technique to apply on the array\n1.Bubble Sort\n2.Insertion Sort\n3.Selection Sort\n");
scanf("%d",&choice);
switch(choice)
{
	case 1:
		bubbsort(a,n);
		printf("\nSorted array is:\n");
		for(i=0;i<n;i++)
		{
		printf("%d\t",a[i]);
		}
		exit(1);
	case 2:
		insersort(a,n);
		printf("\nSorted array is:\n");
		for(i=0;i<n;i++)
		{
		printf("%d\t",a[i]);
		}
		exit(1);
	case 3:
		selecsort(a,n);
```

```
                printf("\nSorted array is:\n");
                for(i=0;i<n;i++)
                {
                printf("%d\t",a[i]);
                }
                exit(1);
        }
}
```

Output:-

Given array is:

2 56 189 7 4 6 123 44 55 10

Select any sorting technique to apply on the array

1.Bubble Sort

2.Insertion Sort

3.Selection Sort

2

Sorted array is:

2 4 6 7 10 44 55 56 123 189

Discussions:-

Time Complexity:

	Worst Case	Average Case	Best Case
Bubble Sort	$O(n^2)$	$O(n^2)$	$O(n^2)$
Selection Sort	$O(n^2)$	$O(n^2)$	$O(n^2)$
Insertion Sort	$O(n^2)$	$O(n^2)$	$O(n)$

- When the given data is almost sorted or already sorted insertion sort works the best because in the inner loop there are little to no movement of the data that's it does not uses all the passes. That's why the best case of insertion sort is O(n).

Assignment No.3 **Date:-7/3/19**

Problem Statement:- Write a program to sort a list of elements using Quicksort.

Algorithm:-

Algorithm quicksort()

Input: An array 'a' with upper bound 'high' and lower bound 'low'.

Output: Elements are in sorted manner.

Steps:

Set low=1 and high=n

pi=partition(a,low,high)

quicksort(a,low,pi-1)

quicksort(a,pi+1,high)

Stop

In this algorithm there is a procedure called partition() the steps of the procedure is as follows,

Procedure partition()

Steps:

pivot=arr[high]

i=low-1

for(j=low to high) do

    If(a[j]<=pivot) then

        i=i+1

```
            swap(arr[i],arr[j])
        End If
End For
Swap(arr[i+1],arr[high])
Stop
```

Source Code:-

```
//Quicksort
#include<stdio.h>
void swap(int *x,int *y)
{
        int temp;
        temp=*x;
        *x=*y;
        *y=temp;
}
int partition(int arr[],int low,int high)
{
        int i,j,pivot=arr[high];
        i=(low-1);
        for(j=low;j<=high-1;j++)
        {
                if(arr[j]<=pivot)
```

```
            {
                i++;
                swap(&arr[i],&arr[j]);
            }
        }
        swap(&arr[i+1],&arr[high]);
        return (i+1);
}
void quicksort(int arr[],int low,int high)
{
        if(low<high)
        {
            int p=partition(arr,low,high);
            quicksort(arr,low,p-1);
            quicksort(arr,p+1,high);
        }
}
int main()
{
        int i;
        int a[10]={2,56,189,7,4,6,123,44,55,10};
        int n=sizeof(a)/sizeof(a[0]);
```

```
    printf("Given array is: \n");
    for(i=0;i<n;i++)
    {
        printf("%d\t",a[i]);
    }
    quicksort(a,0,n-1);
    printf("\nSorted array is:\n");
    for(i=0;i<n;i++)
    {
        printf("%d\t",a[i]);
    }
    return 0;
}
```

Output:-

Given array is:

2 56 189 7 4 6 123 44 55 10

Sorted array is:

2 4 6 7 10 44 55 56 123 189

Discussions:-

- The worst case occurs when the partition process always picks greatest or smallest element as pivot. If we consider above partition strategy where last element is always picked as pivot, the worst case would occur when the array is already sorted in increasing or decreasing order. The complexity of quicksort in worst case is $O(n^2)$
- The best case occurs when the partition process always picks the middle element as pivot. The Complexity of quicksort in best case is O(nlogn)
- The Average case can be considered as the array is 50% sorted. The Complexity of Quicksort in Average case is O(nlogn)

Assignment No.4 **Date:-7/3/19**

Problem Statement:- Write a program to sort a list of elements using Merge sort.

Algorithm:-

Algorithm merge_sort()

Input: An array ‘a’ with lower bound ‘lb’ and upper bound ‘ub’

Output: Elements are in sorted manner.

Steps:

mid=(lb+ub)/2;

If(lb<ub) then

merge_sort(arr,lb,mid)

merge_sort(arr,mid+1,ub)

merge(arr,lb,mid,ub)

End If

Stop

In this Algorithm,there is a procedure called merge(), the steps of the merge() is as follows

Procedure merge()

Steps:

Set i=lb,j=mid+1,index=lb

While(i<=mid && j<=ub) do //array is split into two parts

```
        If(a[i]<a[j]) then
                temp[index]=a[i]
                i=i+1
        Else
                temp[index]=a[j]
                j=j+1
        End If
        Index=index+1
End While
If(i>mid) then
        While(j<=ub) do
                temp[index]=a[j]
                index=index+1
                j=j+1
        End While
Else
        While(i<=mid)
                temp[index]=a[i]
                index=index+1
                j=j+1
        End While
End If
```

```
For(k=lb to index) do
        a[k]=temp[k]
End For
Stop
```

Source Code:-

```
//merge sort
#include<stdio.h>
#define size 100
void merge(int a[],int lb,int mid,int ub)
{
        int i=lb, j=mid+1, index=lb, temp[size], k;
        while((i<=mid) && (j<=ub))
        {
                if(a[i] < a[j])
        {
                temp[index] = a[i++];
        }
        else
        {
                temp[index] = a[j++];
        }
        index++;
```

```
        }
        if(i>mid)
        {
        while(j<=ub)
        {
                temp[index++] = a[j++];
        }
        }
        else
        {
        while(i<=mid)
        {
                temp[index++] = a[i++];
        }
        }
        for(k=lb;k<index;k++)
                a[k] = temp[k];
}
void mergesort(int arr[],int lb,int ub)
{
        int mid=(lb+ub)/2;
        if(lb<ub)
```

```
        {
                mergesort(arr,lb,mid);
                mergesort(arr,mid+1,ub);
                merge(arr,lb,mid,ub);
        }
}
int main()
{
        int i;
        int a[10]={2,56,189,7,4,6,123,44,55,10};
        int n=sizeof(a)/sizeof(a[0]);
        printf("Given array is: \n");
        for(i=0;i<n;i++)
        {
                printf("%d\t",a[i]);
        }
        mergesort(a,0,n-1);
        printf("\nSorted array is:\n");
        for(i=0;i<n;i++)
        {
                printf("%d\t",a[i]);
        }
```

```
    return 0;
}
```

Output:-

Given array is:

2 56 189 7 4 6 123 44 55 10

Sorted array is:

2 4 6 7 10 44 55 56 123 189

Discussions:-

- Merge sort is a sorting algorithm that uses the divide, conquer, and combine algorithmic paradigm.
- The running time of merge sort in the average case and the worst case can be given as O(n logn). Although merge sort has an optimal time complexity, it needs an additional space of O(n) for the temporary array TEMP.

Assignment No.5 **Date:-25/3/19**

Problem Statement:- Write a program to sort a list of elements using Radix Sort.

Algorithm:-

Algorithm radix_sort()

Input: An array ‘a’ with size ‘n’

Output: Elements are in sorted manner.

Steps:

Set NOP=0, divisor=1

large=largest(a,n)

While(large>0) do

NOP++

large=large/10 //large is divided by 10 because decimal numbers are being sorted

End While

For(pass=1 to NOP) do

For(i=1 to 10) do

bucket_count[i]=0

End For

For(i=1 to n) do

rem=(a[i]/divisor)%10

bucket[remainder][bucket_count[remainder]]=a[i]

```
        bucket_count[remainder]=bucket_count[remainder]+1
        End For
        i=0
        For(k=1 to 10) do
            For(j=0 to bucket_count[k]) do
                a[i]=bucket[k][j]
                i=i+1
            End For
        End For
        divisor=divisor*10
End For
Stop
```

In this algorithm, there is a procedure called largest(), the steps of the largest is as follows

Procedure largest()

Steps:

```
Set large=a[1] //array address starting from 1
For(i=2 to n) do
        If(a[i]>large) then
            large=a[i]
        End For
Print large
```

Stop

Source Code:-

```
#include <stdio.h>
#define size 10
int largest(int arr[], int n);
void radix_sort(int arr[], int n);
void main()
{
	int arr[size], i, n;
	printf("\n Enter the number of elements in the array: ");
	scanf("%d", &n);
	printf("\n Enter the elements of the array: ");
	for(i=0;i<n;i++)
	{
		scanf("%d", &arr[i]);
	}
	radix_sort(arr, n);
	printf("\n The sorted array is: \n");
	for(i=0;i<n;i++)
		printf(" %d\t", arr[i]);
}
```

```
int largest(int arr[], int n)
{
	int large=arr[0], i;
	for(i=1;i<n;i++)
	{
		if(arr[i]>large)
		large = arr[i];
	}
	return large;
}
void radix_sort(int arr[], int n)
{
	int bucket[size][size], bucket_count[size];
	int i, j, k, remainder, NOP=0, divisor=1, large, pass;
	large = largest(arr, n);
	while(large>0)
	{
		NOP++;
		large/=size;
	}
	for(pass=0;pass<NOP;pass++) // Initialize the buckets
	{
```

```
            for(i=0;i<size;i++)
                  bucket_count[i]=0;
            for(i=0;i<n;i++)
            {
            // sort the numbers according to the digit at passth
place
                  remainder = (arr[i]/divisor)%size;
                  bucket[remainder][bucket_count[remainder]] =
arr[i];
                  bucket_count[remainder]++;
            }
            // collect the numbers after PASS pass
            i=0;
            for(k=0;k<size;k++)
            {
                  for(j=0;j<bucket_count[k];j++)
                  {
                        arr[i] = bucket[k][j];
                        i++;
                  }
            }
            divisor *= size;
      }
```

```
}
```

Output:-

```
Enter the number of elements in the array: 5

 Enter the elements of the array: 12345
345
8756
2
99

 The sorted array is:
 2      99      345     8756    12345
```

Discussions:-

- To calculate the complexity of radix sort algorithm, assume that there are n numbers that have to be sorted and k is the number of digits in the largest number. In this case, the radix sort algorithm is called a total of k times. The inner loop is executed n times. Hence, the entire radix sort algorithm takes O(kn) time to execute. When radix sort is applied on a data set of finite size (very small set of numbers), then the algorithm runs in O(n) asymptotic time.

- Radix sort is a very simple algorithm. When programmed properly, radix sort is one of the fastest sorting algorithms for numbers or strings of letters.
- Drawback of radix sort is that the algorithm is dependent on digits or letters. This feature compromises with the flexibility to sort input of any data type. For every different data type, the algorithm has to be rewritten.

Assignment No.6 **Date:-2/4/19**

Problem Statement:- Write a program to sort a list of elements using Heap Sort.

Algorithm:-

Algorithm create_heap()

Input: An array ‘a’ with size ‘n’

Output: Elements are in sorted manner.

Steps:

```
For(i=0 to n) do
        Insert(a[i],n)
End For
```

In this Algorithm, there is a procedure called Insert(), the steps of the Insert() is as follows,

Procedure Insert()

Steps:

```
While(loc>0) do
        parent=(loc-1)/2
        If(num<=a[parent]) then
                a[loc]=num
        End If
        a[loc]=a[parent]
        loc=parent
End While
```

a[0]=num

Stop

Algorithm heap_sort()

Input: An array ‘a’ with size ‘n’

Output: Elements are in sorted manner.

Steps:

For(l=n to 1) do

    Del_root(l)

End For

In this Algorithm, there is a procedure called Del_root(), the steps of the Del_root() is as follows,

Procedure Del_root()

Steps:

a[i]=a[last]

a[last]=temp

left=2*i+1

right=2*i+2

While(right<left) do

    If(a[right]<=a[left]) then

        Swap(a[i],a[left])

        i=left

```
        Else
                Swap(a[i],a[right])
                I=right
        End If
        left=2*i+1
        right=2*i+2
End While
If(left=last-1 && a[i]<a[left]) then
        Swap(a[i],a[left])
End If
Display a
Stop
```

Source Code:-

```
/* HEAP SORT */
#include<stdio.h>
#include<stdlib.h>
int arr[100], n;
void insert (int, int );
void create_heap ();
void heap_sort ();
void del_root (int );
void display ();
```

```
int main()
{
	int i;
	printf("\n\t\t # HEAP SORT #");
	printf("\n----------------------------------------------------");
	printf ("\n\n Number of elements to Insert (Max 100) : ");
	scanf ("%d",&n);
	if (n<0 || n>100)
	{
		printf("\n\tError : Invalid Input...cannot process.");
		exit (0);
	}
	printf("\n\n");
	for (i=0;i<n;i++)
	{
		printf ("\tEnter ARRAY [%d] = ",i);
		scanf ("%d",&arr[i]);
	}
	printf ("\n\n :: The Inputted Array :: \n\n");
	printf("\t");
	display ();
	create_heap (); // heapify - creation of heap
```

```
        printf ("\n\n :: Construction of Heap :: \n\n");
        display ();
        heap_sort (); // sorting - using heap sort
        printf ("\n\n :: The Sorted Array :: \n\n");
        printf ("\t");
        display ();
        return 0;
}
////////////////////////////////////////////////////////////
/* Function definitions */
////////////////////////////////////////////////////////////
void insert (int num, int loc)
{
        int parent;
        while (loc > 0)
        {
                parent=(loc-1)/2;
                if (num <= arr[parent])
                {
                        arr[loc]=num;
                        return;
                }
```

```
            arr[loc]=arr[parent];
            loc=parent;
        }
        arr[0]=num;
}
////////////////////////////////////////////////////////////
void create_heap ()
{
        int i;
        for (i=0;i<n;i++)
            insert (arr[i],i); // insertion into heap
}
////////////////////////////////////////////////////////////
void heap_sort ()
{
        int last;
        for (last=n-1;last>0;last--)
            del_root (last);
}
////////////////////////////////////////////////////////////
void del_root (int last)
{
```

```
int left, right, i, temp;
i=0;
/* exchange of last element with root */
temp=arr[i];
arr[i]=arr[last];
arr[last]=temp;
left =2*i+1; // left child of root
right=2*i+2; // right child of root
while (right < last)
{
        if ((arr[i] >= arr[left]) && (arr[i] >= arr[right]))
                return;
        if (arr[right] <= arr[left])
        {
                temp=arr[i];
                arr[i]=arr[left];
                arr[left]=temp;
                i=left;
        }
        else
        {
                temp=arr[i];
```

```
                arr[i]=arr[right];
                arr[right]=temp;
                i=right;
            }
            left =2*i+1;
            right=2*i+2;
        }
        if ((left == last-1) && (arr[i] < arr[left]))
        {
            temp=arr[i];
            arr[i]=arr[left];
            arr[left]=temp;
        }
 display ();
}
//////////////////////////////////////////////////////////////
void display ()
{
        int i;
        printf("\t");
        for (i=0;i<n;i++)
            printf(" %d ", arr[i]);
```

```
        printf("\n");
}
//////////////////////////////////////////////////////////////
```

Output:-

```
        # HEAP SORT #
---------------------------------------------------

 Number of elements to Insert (Max 100) : 5


    Enter ARRAY [0] = 7
    Enter ARRAY [1] = 45
    Enter ARRAY [2] = 3
    Enter ARRAY [3] = 2
    Enter ARRAY [4] = 21


 :: The Inputted Array ::

        7  45  3  2  21
```

:: Construction of Heap ::

45 21 3 2 7

21 7 3 2 45

7 2 3 21 45

3 2 7 21 45

2 3 7 21 45

:: The Sorted Array ::

2 3 7 21 45

Discussions:-

Heap sort uses two heap operations: *insertion* and *root deletion*. Each element extracted from the root is placed in the last empty location of the array.
In phase 1, when we build a heap, the number of comparisons to find the right location of the new element in H cannot exceed the depth of H. Since H is a complete tree, its depth cannot exceed m, where m is the number of elements in heap H.
Thus, the total number of comparisons g(n) to insert n elements of ARR in H is bounded as:
g(n) <= n log n Hence, the running time of the first phase of the heap sort algorithm is O(n log n).
In phase 2, we have H which is a complete tree with m elements having left and right sub-trees as heaps. Assuming L to be the root of the tree, *reheaping* the tree would need 4

comparisons to move L one step down the tree H. Since the depth of H cannot exceed O(log m), reheaping the tree will require a maximum of 4 log m comparisons to find the right location of L in H.

Since n elements will be deleted from heap H, reheaping will be done n times. Therefore, the number of comparisons to delete n elements is bounded as:

$h(n) <= 4n \log n$

Hence, the running time of the second phase of the heap sort algorithm is O(n log n).

Each phase requires time proportional to O(n log n). Therefore, the running time to sort an array of n elements in the worst case is proportional to O(n log n).

Therefore, we can conclude that heap sort is a simple, fast, and stable sorting algorithm that can be used to sort large sets of data efficiently.

Assignment No.7 **Date:-2/4/19**

Problem Statement:- Write a program to sort a list of elements using Shell Sort.

Algorithm:-

Algorithm shell_sort()

Input: An array ‘a’ with size ‘n’

Output: Elements are in sorted manner.

Steps:

```
Set flag=1, g_size=n
While(flag=1 && g_size>1) do
    flag=0
    g_size=(g_size+1)/2
    For(i=0 to n-g_size) do
        If (a[i+g_size]>a[i]) then
            Swap(a[i+g_size],a[i])
            flag=0
        End If
    End for
End While
```

Source Code:-

```
#include<stdio.h>
void main()
{
```

```
int arr[10]={-1};
int i, j, n, flag = 1, gap_size, temp;
printf("\n Enter the number of elements in the array: ");
scanf("%d", &n);
printf("\n Enter %d numbers: ",n); // n was added
for(i=0;i<n;i++)
scanf("%d", &arr[i]);
gap_size = n;
while(flag == 1 || gap_size > 1)
{
flag = 0;
gap_size = (gap_size + 1) / 2;
for(i=0; i< (n - gap_size); i++)
{
if( arr[i+gap_size] < arr[i])
{
temp = arr[i+gap_size];
arr[i+gap_size] = arr[i];
arr[i] = temp;
flag = 0;
}
}
```

```
}
printf("\n The sorted array is: \n");
for(i=0;i<n;i++){
printf(" %d\t", arr[i]);
}
}
```

Output:-

Enter the number of elements in the array: 5

Enter 5 numbers: 1315

3

345

6

29

The sorted array is:

3 6 29 345 1315

Discussions:-

Time complexity of above implementation pf shellsort is $O(n^2)$. In the above implementation gap is reduce by half in every iteration.

Assignment No.8 **Date:-30/4/19**

Problem Statement:- Write a Program to create a Binary Search Tree and include following operations in tree:

(a) Insertion (Recursive and Iterative Implementation)

(b) Deletion by copying

(c) Deletion by Merging

(d) Search a no. in BST

(e) Display its preorder, postorder and inorder traversals Recursively

(f) Display its preorder, postorder and inorder traversals Iteratively

(g) Display its level-by-level traversals

(h) Count the non-leaf nodes and leaf nodes

(i) Display height of tree

(j) Create a mirror image of tree

(k) Check whether two BSTs are equal or not

Algorithm:-

SearchElement (TREE, VAL)
Step 1: IF TREE DATA = VAL OR TREE = NULL
Return TREE
ELSE
IF VAL < TREE DATA
Return searchElement(TREE LEFT, VAL)
ELSE
Return searchElement(TREE RIGHT, VAL)
[END OF IF]
[END OF IF]

Step 2: END

Insert (TREE, VAL)

```
Step 1: IF TREE = NULL
Allocate memory for TREE
SET TREE DATA = VAL
SET TREE LEFT = TREE RIGHT = NULL
ELSE
IF VAL < TREE DATA
Insert(TREE LEFT, VAL)
ELSE
Insert(TREE RIGHT, VAL)
[END OF IF]
[END OF IF]
Step 2: END
```

Delete (TREE, VAL)

```
Step 1: IF TREE = NULL
Write "VAL not found in the tree"
ELSE IF VAL < TREE DATA
Delete(TREE->LEFT, VAL)
ELSE IF VAL > TREE DATA
Delete(TREE RIGHT, VAL)
ELSE IF TREE LEFT AND TREE RIGHT
SET TEMP = findLargestNode(TREE LEFT)
SET TREE DATA = TEMP DATA
Delete(TREE LEFT, TEMP DATA)
ELSE
SET TEMP = TREE
IF TREE LEFT = NULL AND TREE RIGHT = NULL
SET TREE = NULL
ELSE IF TREE LEFT != NULL
SET TREE = TREE LEFT
ELSE
SET TREE = TREE RIGHT
[END OF IF]
FREE TEMP
[END OF IF]
```

Step 2: END

Height (TREE)
Step 1: IF TREE = NULL
Return
ELSE
SET LeftHeight = Height(TREE LEFT)
SET RightHeight = Height(TREE RIGHT)
IF LeftHeight > RightHeight
Return LeftHeight + 1
ELSE
Return RightHeight + 1
[END OF IF]
[END OF IF]
Step 2: END

totalNodes(TREE)
Step 1: IF TREE = NULL
Return
ELSE
Return totalNodes(TREE LEFT) + totalNodes(TREE RIGHT) + 1
[END OF IF]
Step 2: END

totalInternalNodes(TREE)
Step 1: IF TREE = NULL
Return
[END OF IF]
IF TREE LEFT = NULL AND TREE RIGHT = NULL
Return
ELSE
Return totalInternalNodes(TREE LEFT) + totalInternalNodes(TREE RIGHT) + 1
[END OF IF]
Step 2: END

totalExternalNodes(TREE)
Step 1: IF TREE = NULL
Return
ELSE IF TREE LEFT = NULL AND TREE RIGHT = NULL
Return 1
ELSE
Return totalExternalNodes(TREE LEFT) +
totalExternalNodes(TREE RIGHT)
[END OF IF]
Step 2: END

MirrorImage(TREE)
Step 1: IF TREE != NULL
MirrorImage(TREE LEFT)
MirrorImage(TREE RIGHT)
SET TEMP = TREE LEFT
SET TREE LEFT = TREE RIGHT
SET TREE RIGHT = TEMP
[END OF IF]
Step 2: END

deleteTree(TREE)
Step 1: IF TREE != NULL
deleteTree (TREE LEFT)
deleteTree (TREE RIGHT)
Free (TREE)
[END OF IF]
Step 2: END

findSmallestElement(TREE)
Step 1: IF TREE = NULL OR TREE LEFT = NULL
Returen TREE
ELSE
Return findSmallestElement(TREE LEFT)
[END OF IF]

Step 2: END

findLargestElement(TREE)
Step 1: IF TREE = NULL OR TREE RIGHT = NULL
Return TREE
ELSE
Return findLargestElement(TREE RIGHT)
[END OF IF]
Step 2: END

Source Code:-

```
#include <stdio.h>
#include <conio.h>
#include <malloc.h>
struct node
{
int data;
struct node *left;
struct node *right;
};
struct node *tree;
void create_tree(struct node *);
struct node *insertElement(struct node *, int);
void preorderTraversal(struct node *);
void inorderTraversal(struct node *);
void postorderTraversal(struct node *);
struct node *findSmallestElement(struct node *);
struct node *findLargestElement(struct node *);
struct node *deleteElement(struct node *, int);
struct node *mirrorImage(struct node *);
int totalNodes(struct node *);
int totalExternalNodes(struct node *);
int totalInternalNodes(struct node *);
int Height(struct node *);
struct node *deleteTree(struct node *);
```

```
int main()
{
int option, val;
struct node *ptr;
create_tree(tree);
clrscr();
do
{
printf("\n ******MAIN MENU******* \n");
printf("\n 1. Insert Element");
printf("\n 2. Preorder Traversal");
printf("\n 3. Inorder Traversal");
printf("\n 4. Postorder Traversal");
printf("\n 5. Find the smallest element");
printf("\n 6. Find the largest element");
printf("\n 7. Delete an element");
printf("\n 8. Count the total number of nodes");
printf("\n 9. Count the total number of external nodes");
printf("\n 10. Count the total number of internal nodes");
printf("\n 11. Determine the height of the tree");
printf("\n 12. Find the mirror image of the tree");
printf("\n 13. Delete the tree");
printf("\n 14. Exit");
printf("\n\n Enter your option : ");
scanf("%d", &option);
switch(option)
{
case 1:
printf("\n Enter the value of the new node : ");
scanf("%d", &val);
tree = insertElement(tree, val);
break;
case 2:
printf("\n The elements of the tree are : \n");
preorderTraversal(tree);
break;
case 3:
```

```
printf("\n The elements of the tree are : \n");
inorderTraversal(tree);
break;
case 4:
printf("\n The elements of the tree are : \n");
postorderTraversal(tree);
break;
case 5:
ptr = findSmallestElement(tree);
printf("\n Smallest element is :%d",ptr–>data);
break;
case 6:
ptr = findLargestElement(tree);
printf("\n Largest element is : %d", ptr–>data);
break;
case 7:
printf("\n Enter the element to be deleted : ");
scanf("%d", &val);
tree = deleteElement(tree, val);
break;
case 8:
printf("\n Total no. of nodes = %d", totalNodes(tree));
break;
case 9:
printf("\n Total no. of external nodes = %d",
totalExternalNodes(tree));
break;
case 10:
printf("\n Total no. of internal nodes = %d",
totalInternalNodes(tree));
break;
case 11:
printf("\n The height of the tree = %d",Height(tree));
break;
case 12:
tree = mirrorImage(tree);
break;
```

```
case 13:
tree = deleteTree(tree);
break;
}
}while(option!=14);
getch();
return 0;
}
void create_tree(struct node *tree)
{
tree = NULL;
}
struct node *insertElement(struct node *tree, int val)
{
struct node *ptr, *nodeptr, *parentptr;
ptr = (struct node*)malloc(sizeof(struct node));
ptr–>data = val;
ptr–>left = NULL;
ptr–>right = NULL;
if(tree==NULL)
{
tree=ptr;
tree–>left=NULL;
tree–>right=NULL;
}
else
{
parentptr=NULL;
nodeptr=tree;
while(nodeptr!=NULL)
{
parentptr=nodeptr;
if(val<nodeptr–>data)
nodeptr=nodeptr–>left;
else
nodeptr = nodeptr–>right;
}
```

```
if(val<parentptr–>data)
parentptr–>left = ptr;
else
parentptr–>right = ptr;
}
return tree;
}
void preorderTraversal(struct node *tree)
{
if(tree != NULL)
{
printf("%d\t", tree–>data);
preorderTraversal(tree–>left);
preorderTraversal(tree–>right);
}
}
void inorderTraversal(struct node *tree)
{
if(tree != NULL)
{
inorderTraversal(tree->left);
printf("%d\t", tree->data);
inorderTraversal(tree->right);
}
}
void postorderTraversal(struct node *tree)
{
if(tree != NULL)
{
postorderTraversal(tree->left);
postorderTraversal(tree->right);
printf("%d\t", tree->data);
}
}
struct node *findSmallestElement(struct node *tree)
{
if( (tree == NULL) || (tree->left == NULL))
```

```
return tree;
else
return findSmallestElement(tree ->left);
}
struct node *findLargestElement(struct node *tree)
{
if( (tree == NULL) || (tree->right == NULL))
return tree;
else
return findLargestElement(tree->right);
}
struct node *deleteElement(struct node *tree, int val)
{
struct node *cur, *parent, *suc, *psuc, *ptr;
if(tree–>left==NULL)
{
printf("\n The tree is empty ");
return(tree);
}
parent = tree;
cur = tree–>left;
while(cur!=NULL && val!= cur–>data)
{
parent = cur;
cur = (val<cur–>data)? cur–>left:cur–>right;
}
if(cur == NULL)
{
printf("\n The value to be deleted is not present in the tree");
return(tree);
}
if(cur–>left == NULL)
ptr = cur–>right;
else if(cur–>right == NULL)
ptr = cur–>left;
else
{
```

```
// Find the in–order successor and its parent
psuc = cur;
cur = cur–>left;
while(suc–>left!=NULL)
{
psuc = suc;
suc = suc–>left;
}
if(cur==psuc)
{
// Situation 1
suc–>left = cur–>right;
}
else
{
// Situation 2
suc–>left = cur–>left;
psuc–>left = suc–>right;
suc–>right = cur–>right;
}
ptr = suc;
}
// Attach ptr to the parent node
if(parent–>left == cur)
parent–>left=ptr;
else
parent–>right=ptr;
free(cur);
return tree;
}
int totalNodes(struct node *tree)
{
if(tree==NULL)
return 0;
else
return(totalNodes(tree–>left) + totalNodes(tree–>right) + 1);
}
```

```
int totalExternalNodes(struct node *tree)
{
if(tree==NULL)
return 0;
else if((tree–>left==NULL) && (tree–>right==NULL))
return 1;
else
return (totalExternalNodes(tree–>left) +
totalExternalNodes(tree–>right));
}
int totalInternalNodes(struct node *tree)
{
if( (tree==NULL) || ((tree–>left==NULL) && (tree–
>right==NULL)))
return 0;
else
return (totalInternalNodes(tree–>left)
+ totalInternalNodes(tree–>right) + 1);
}
int Height(struct node *tree)
{
int leftheight, rightheight;
if(tree==NULL)
return 0;
else
{
leftheight = Height(tree–>left);
rightheight = Height(tree–>right);
if(leftheight > rightheight)
return (leftheight + 1);
else
return (rightheight + 1);
}
}
struct node *mirrorImage(struct node *tree)
{
struct node *ptr;
```

```
if(tree!=NULL)
{
mirrorImage(tree–>left);
mirrorImage(tree–>right);
ptr=tree–>left;
ptr–>left = ptr–>right;
tree–>right = ptr;
}
}
struct node *deleteTree(struct node *tree)
{
if(tree!=NULL)
{
deleteTree(tree–>left);
deleteTree(tree–>right);
free(tree);
}
}
```

Output:-

```
*******MAIN MENU*******
1. Insert Element
2. Preorder Traversal
3. Inorder Traversal
4. Postorder Traversal
5. Find the smallest element
6. Find the largest element
7. Delete an element
8. Count the total number of nodes
9. Count the total number of external nodes
10. Count the total number of internal nodes
11. Determine the height of the tree
12. Find the mirror image of the tree
13. Delete the tree
14. Exit
Enter your option : 1
```

Enter the value of the new node : 1
Enter the value of the new node : 2
Enter the value of the new node : 4
Enter your option : 3
2 1 4
Enter your option : 14

Discussions:-

A binary search tree, also known as an ordered binary tree, is a variant of binary trees in which the nodes are arranged in an order. In a binary search tree, all the nodes in the left sub-tree have a value less than that of the root node. Correspondingly, all the nodes in the right sub-tree have a value either equal to or greater than the root node. The same rule is applicable to every sub-tree in the tree.

Assignment No.9 **Date:-**

Problem Statement:- Implement Linked List using templates. Include functions for insertion, deletion and search of a number, reverse the list and concatenate two linked lists.

Algorithm:-

Algorithm for insertion:-

Algorithm create_sll(data)

Input: A pointer called ‘start’ and ‘data’which is to be inserted

Output: A singly linked list with with the corresponding nodes with a pointer pointing to the first node called ‘start’

Steps:

Set start=NULL

Allocate new memory for pointer p

(p->info)=data

If(start!=NULL) then // when the list is empty

    Start=p

    (p->link)=NULL

Else

    t=p

    while(t!=NULL) do //when the list is not empty

        t=(t->link) // traverse to the last existing node

    End while

    (t->info)=data

(t->link)=NULL

End If

Stop

Algorithm for search:

Algorithm search_sll(d,start)

Input: The data which is required to search is contained is ‘d’.

And the pointer pointing to the first node called ‘start’

Output: The required data will be shown.

Steps:

Set p=start

Set count=0

While((p->data)!=d) do

p=(p->link)

count=count+1

End while

Print p->data

Print count

Stop

Algorithm for deletion:

Algorithm delete_sll(d,start)

Input: ‘d’ contains the data to be searched and ‘start’ is a pointer which holds the address of the first node

Output: A singly linked list with one less node

Steps:

Set p=start

While(p->data!=d) do

    q=p //a pointer q to hold the address of previous node

    p=p->link

end while

q->link=p->link

deallocate p

Stop

Algorithm for reverse:

Algorithm reverse_sll(start)

Input: A singly linked list with a pointer pointing the first node called start

Output: The given linked list in reverse order

Steps:

Set cur=start

prev=(cur->link)

ptr1=(prev->link)

while(ptr1!=NULL)

    (prev->link)=cur

    cur=prev

    prev=ptr1

    ptr1=(ptr1->link)

end while

(prev->link)=cur

(start->link)=NULL

start=prev

Stop

Algorithm for Merge:

Algorithm merge_sll(a,b)

Input: a and b are two pointers which contains the address of the first node of two separate singly linked list

Output: A singly linked list which contains all the node of both of the lists

Steps:

If(a->link=NULL) then

(a->link)=b

Else

Merge_sll(a->link,b) * recursively calling the next node until the last one*\

Stop

Source Code:

```
#include<stdio.h>
#include<stdlib.h>
typedef struct node{
    int data;
```

```
        struct node *link;
}node;
void display_sll(node *start)
{
        node *ptr;
        if(start==NULL)
        {
                printf("List is empty\nOPERATION FAILED\n");
        }
        ptr=start;
        while(ptr!=NULL){
                printf("%d\n",ptr->data); //show the data part of the
                node
                ptr=ptr->link;  //moves to the next pointer
        }
}
void insert_sll(node *start,int dt) //creating nodes for the list
{
        node *p,*temp;
        p=(struct node*)malloc(sizeof(node));     //allocating
memeory for the node
        if(start==NULL)     //start=NULL means list is empty
        {
```

```
            p->link=NULL;        //first node refers to NULL
            p->data=dt;          //inserting the data into the node
            start=p;             //start pointing to the first node
            printf("\nNode added successfully\n\a");
            printf("\nUpdated list: \n");
            display_sll(start);
        }
        else   //start!=NULL means list is not empty and there are other node present
        {
            temp=start; //temporary node type pointer holds the value of start
            while(temp->link!=NULL)//while it is not the end of the lsit
            {
                temp=temp->link;  //moves to the next node
            }
            temp->link=p; //temp points to the new node
            p->link=NULL; //new node points to NULL to identify the node as the last node
            p->data=dt;   //inserting the data part at the node
            printf("\nNode added successfully\n\a");
            printf("\nUpdated list: \n");
            display_sll(start); //displaying the list
```

```
        }
}
void search_sll(node *start,int dt)
{
        node *p=start;
        int count=0; //count variable is used to tell the position of the data
        while(p->data!=dt) //traverse untill the data is found
        {
                p=p->link;
                count++;
        }
        printf("The following data %d is situated at %d position\n\a",p->data,count);
}
void del_sll(node *start,int dt){
        node *p=start,*q;
        while(p->data!=dt) //traverse until the location is found
        {
                q=p; //q pointer is used to store the previous value
                p=p->link;
        }
        q->link=p->link;
```

```
        free(p);
        printf("Node deleted successfully\n\a");
        printf("Updated List\n");
        display_sll(start);
}
void reverse_sll(node* start)
{
        node *cur=start,*prev=cur->link,*ptr1=prev->link;
        while(ptr1!=NULL)
        {
                prev->link=cur;
                cur=prev;
                prev=ptr1;
                ptr1=ptr1->link;
        }
        prev->link=cur;
        start->link=NULL;
        start=prev;
        printf("Updated List:\n");
        display_sll(start);
}
void merge_sll(node* a,node* b)
```

```
{
	if(a->link==NULL)
	{
		a->link=b; // merging the last node of a with the first node of b
		printf("Both of the list are merged into list a\n\a");
	}
	else
	merge_sll(a->link,b); // recursively calls for the next node
}
int main()
{
	struct node *prev,*a,*b,*p;
	int n,i,option,l,dat;
	printf("Enter the number of nodes to be entered in list a:\t");
	scanf("%d",&n);
	a=NULL;
	for(i=1;i<=n;i++)
	{
		p=(node*)malloc(sizeof(node));
		printf("Enter the data for the node:\t");
		scanf("%d",&p->data);
```

```
        if(a==NULL)
            a=p;
        else
            prev->link=p;
        prev=p;
    }
    printf("Enter the number of elements to be entered in the list b:\t");
    scanf("%d",&n);
    b=NULL;
    for(i=1;i<=n;i++)
    {
        p=(node*)malloc(sizeof(node));
        printf("Enter the data for the node:\t");
        scanf("%d",&p->data);
        if(b==NULL)
            b=p;
        else
            prev->link=p;
        prev=p;
    }
    while(1)
    {
```

```
        printf("Enter any of the given options to do these operations\n1.Insert\n2.Search\n3.Delete\n4.Reverse\n5.Merge \n6.Display\nEnter Any other key to exit\n");
        scanf("%d",&option);
        switch(option)
        {
            case 1:
                printf("Enter the list number to do the operation\n1.a\n2.b\n");
                scanf("%d",&l);
                printf("Enter the data to be inserted:\t");
                scanf("%d",&dat);
                if(l==1)
                {
                    insert_sll(a,dat);
                }
                else
                {
                    insert_sll(b,dat);
                }
                break;
            case 2:
                printf("Enter the list number to do the operation\n1.a\n2.b");
```

```
                scanf("%d",&l);
                printf("Enter the data to search:\t");
                scanf("%d",&dat);
                if(l==1)
                {
                        search_sll(a,dat);
                }
                else
                {
                        search_sll(b,dat);
                }
                break;
        case 3:
                printf("Enter the list number to do the
operation\n1.a\n2.b");
                scanf("%d",&l);
                printf("Enter the data to delete:\t");
                scanf("%d",&dat);
                if(l==1)
                {
                        del_sll(a,dat);
                }
                else
```

```
                {
                        del_sll(b,dat);
                }
                break;
        case 4:
                printf("Enter the list number to do the operation\n1.a\n2.b");
                scanf("%d",&l);
                if(l==1)
                {
                        reverse_sll(a);
                }
                else
                {
                        reverse_sll(b);
                }
                break;
        case 5:
                merge_sll(a,b);
        case 6:
                printf("Enter the list number to do the operation\n1.a\n2.b");
                scanf("%d",l);
```

```
                    if(l==1)
                    {
                        printf("Displaying a:\n");
                        display_sll(a);
                    }
                    else
                    {
                        printf("Displaying b:\n");
                        display_sll(b);
                    }
            }
        }
        }
```

Output:

*****Singly Linked List*****

Enter any of these keywords to do the operations

Press 1 to do the insertion of a node

Press 2 to search a node

Press 3 to do the deletion of the nodes

Press 4 to do the reverse of the list

Press 5 to merge lists

Press 6 to display

1

Enter the data for the node: 23

Node added successfully

Updated list:

1.23

*****Singly Linked List*****

Enter any of these keywords to do the operations

Press 1 to do the insertion of a node

Press 2 to search a node

Press 3 to do the deletion of the nodes

Press 4 to do the reverse of the list

Press 5 to merge lists

Press 6 to display

1

Enter the data for the node: 34

Node added successfully

Updated list:

1.23

2.34

*****Singly Linked List*****

Enter any of these keywords to do the operations

Press 1 to do the insertion of a node

Press 2 to search a node

Press 3 to do the deletion of the nodes

Press 4 to do the reverse of the list

Press 5 to merge lists

Press 6 to display

3

Enter the data to be deleted: 2

Node deleted successfully

Updated list:

1.23

*****Singly Linked List*****

Enter any of these keywords to do the operations

Press 1 to do the creation of a node

Press 2 to do the display of the node

Press 3 to do the insertion of the nodes

Press 4 to do the deletion of the nodes

Press 5 to do reverse the list

Press 6 to get the maximum of all nodes

Press 7 to get the minimum of all node

Press 8 to make an odd and even list element out of the main list

9

Discussions:

1. We are working with two linked list here so that we can do the merge operation.

2. Time Complexity:

- For Insert Operation the complexity will be O(n) because we need to traverse the list to add a node
- For Delete and Search Operation the complexity will be O(n) because we need to traverse to the desired location

3. Space Complexity:

- For Insertion and Operation the complexity will be O(1) because it just requires a single space to get added
- For Merge Operation the complexity will be O(m+n) where m is the size of the 1^{st} list and n is the size of 2^{nd} list

Assignment No.10 **Date:-**

Problem Statement:- Implement Doubly linked list. Include functions Insertion, Deletion, Search of a Number, reverse the list.

Algorithm:-

Algorithm insert_dll()

Input: A pointer called ‘start’ which will point the first node

Output: A doubly inked list with the node added in the desired location.

Steps:

cur=start

Allocate memory for ptr

If(start=NULL) then //data is going to be inserted in the first location

Scan ptr->data

ptr->next=NULL

ptr->prev=NULL

Else

Scan key

While(cur->data!=key && cur!=NULL) do

Cur=cur->next

End while

If(cur->data=key) then //data inserted to a desired location

ptr->next=cur->next

cur->next->prev=ptr

ptr->prev=cur

cur->next=ptr

Else

Print “Key not found”

End If

End If

Stop

Algorithm display_dll()

Input: A Doubly linked list with a pointer pointing the first node called ‘start’.

Output: All the elements present in the list

Steps:

ptr=start

while(ptr->next!=NULL) do

Print ptr->data

End while

Stop

Algorithm del_dll()

Input: 'Key' is the value to be deleted

Output: A Doubly linked list without the node included the value 'key'

Steps:

Scan key

Set ptr=start and prev1=NULL

While(ptr!=NULL&&ptr->data!=key) do

    prev1=ptr

    ptr=ptr->start

End while

If(ptr=start**)** then // if the first node is to be deleted

    Start=ptr->next

End If

If(ptr->data=key) then // deletion at any position

    prev1->next=ptr->next

    deallocate ptr

Else

    Print "Node is not present"

End If

Stop

Algorithm reverse_dll()

Input: A Doubly linked list with a starting pointer ‘start’

Output: Given list in reverse order

Steps:

Set pre=NULL and cur=start

While(cur!=NULL) // traverse the list

    temp=cur->prev //temp is a temporary pointer for swapping

    cur->prev=cur->next

    cur->prev=temp

    pre=cur //update the previous node before moving to the next node

    cur=cur->prev //move to the next node of dll

End while

If(pre!=NULL)

    start=pre //update the start to the last node

Stop

Algorithm search_dll()

Input: ‘key’ is the valued to be searched

Output: If found in the list then the position of the value will be shown.

Steps:

ptr=start

Set count=0

Scan key

While(ptr->data!=key) do

    count=count+1

    ptr=ptr->next

End while

If(ptr=NULL)

    Print “Key not found”

Else

    Print count

End If

Stop

Source Code:-

```
// Doubly linked list opertaions
#include<stdio.h>
#include<stdlib.h>
typedef struct node{
    int data;
    struct node *prev;
    struct node *next;
```

```
}node;
node *start=NULL,*ptr=NULL,*prev1=NULL;
void insert()
{
	int key;
	struct node *cur;
	cur=start;
	ptr=(struct node*)malloc(sizeof(struct node));

	if(start=NULL) // if the list is empty
	{
		printf("\nEnter new value for the node : ");
		scanf("%d",ptr->data);
		ptr->next=ptr->prev=NULL;
		start=prev1=ptr;
	}
	else
	{
		printf("\nEnter the value after which you want to insert new value : ");
		scanf("%d",&key);

		while(cur->data!=key&&cur!=NULL)
```

```
            {
                cur=cur->next;
            }
            if(cur->data==key)
            {
                printf("\nEnter value for new node : ");
                scanf("%d",&key);
                ptr->next=cur->next;
                cur->next->prev=ptr;
                ptr->prev=cur;
                cur->next=ptr; // setting the node after the
selected node
            }
            else
                printf("\nKey not found\n");
        }
}
void display()
{
        ptr=start;
        printf("\nThe elements are\n");
        while(ptr->next!=NULL)
        {
```

```
            printf("%d\t",ptr->data);
            ptr=ptr->next;
        }
}
void del()
{
        int key;ptr=start;prev1=NULL;
        printf("\nEnter a value to delete : ");
        scanf("%d",&key);
        while(ptr!=NULL&&ptr->data!=key)
        {
            prev1=ptr;
            ptr=ptr->next;
        }
        if(ptr==start) // if the first node is to be deleted
        {
            start=ptr->next;
        }
        if(ptr->data==key) // deletion at any position
        {
            prev1->next=ptr->next;
            free(ptr);
```

```
        }
        else
                printf("\nNode is not present\n");
}
void reverse()
{
        struct node *pre=NULL;
        struct node *cur=start;

        // traverse the list
        while(cur!=NULL)
        {
                // swap next and previous pointers
                struct node* temp=cur->prev;
                cur->prev=cur->next;
                cur->prev=temp;
                // update the previous node before moving to the next
node
                pre=cur;
                // move to the next node of doubly linked list
                cur=cur->prev;
        }
        // update the start to the last node
```

```
        if(pre!=NULL){
                start=pre;
        }
}
int search()
{
        ptr=start;
        int count=0,k;
        printf("\nEnter a value to search : ");
        scanf("%d",&k);
        while(ptr->data!=k)
        {
                ++count;
                ptr=ptr->next;
        }
        if(ptr==NULL)
        {
                printf("\nThe given data is not present in the list\n");
        }
        else
        {
                printf("\nThe given data is present at %d
location",count);
```

```
        }
}
int main()
{
        int prompt;
        printf("******Doubly Linked List******\n\n");
        while(1)
        {
                printf("\nEnter the options for the
operations\n1.Insert\n2.Delete\n3.Search\n4.Reverse\n5.Displa
y\nEnter any other key to exit\n");
                scanf("%d",&prompt);
                switch(prompt)
                {
                        case 1:
                                insert();
                        case 2:
                                del();
                        case 3:
                                search();
                        case 4:
                                reverse();
                        case 5:
```

```
                    display();
                default:
                    exit(1);
            }
        }
}
```

Output:-

******Doubly Linked List******

Enter the options for the operations

1.Insert

2.Delete

3.Search

4.Reverse

5.Display

Enter any other key to exit

Enter new value for the node : 21

Enter the options for the operations

1.Insert

2.Delete

3.Search

4.Reverse

5.Display

Enter any other key to exit

2

Enter a value to delete : 31

Node is not present

Enter the options for the operations

1.Insert

2.Delete

3.Search

4.Reverse

5.Display

Enter any other key to exit

9

Discussions:-

- Doubly linked list is used over circular or singly linked list because both way traversal is possible in Doubly linked list
- Reversal of the list is much more easier
- Doubly linked uses too much pointer variable that's why it is difficult to handle
- It requires more memory and time for the operations like insertion, deletion etc.

Assignment No.11 **Date:-**

Problem Statement:- Implement Circular Linked List. Include functions Insertion, Deletion, Search of a number, reverse the list.

Algorithm:-

Algorithm create_cll()

Input: 'head' is a null pointer and 'n' is the number of node required

Output: 'head' will point the first node of the list

Steps:

If(n>=1) then

Allocate memory for 'head' //allocating the first memory

Scan data

head->data=data

head->next=NULL

prevnode=head

End If

For(i=2 to n) do//allocating the memory for the rest of the list

Allocate memory for 'Newnode'

Scan data

Newnode->data=data

Newnode->next=NULL

prevnode->next=Newnode

prevnode=Newnode

End For

prevnode->next=head //last node is pointing to the head

Stop

Algorithm display_cll()

Input: 'head' pointer which points to the first node

Output: All the elements of the list are shown

Steps:

If(head=NULL)

Print "List is Empty"

Else

Current=head

While(Current->next!=head) do

Print Current->data

Current=Current->next

End while

Stop

Algorithm insert_begin()

Input: 'data' is the value to be inserted

Output: A circular linked list with the added node

Steps:

If(head=NULL)

    Print “List is Empty”

Else

    Allocate new memory for “newnode”

    newnode->data=data

    newnode->next=head

    current=head

    while(current->next!=head)

        current=current->next

    End while

    current->next=newnode

    head=newnode

End If

Stop

Algorithm insertAtN()

Input: ‘data’ is the value to be inserted and ‘position’ the place where the data will be inserted

Output: A circular linked list with the added node

Steps:

If(head=NULL)

    Print “List is Empty”

```
Else
    If(position=1)
    insert_begin(data)
    Else
        Allocate memory for ‘newnode’
        newnode->data=data
        current=head
        for(i=2 to position-1) do //Traverse to n-1 node
            current=current->next
        End for
        newnode->next=current->next //Links new node with
                        node ahead of it and previous to it
        current->next=newnode
    End If
End If
Stop
```

Algorithm delete()

Input: ‘key’ is the data to be deleted

Output: If found then A circular linked list without the node containing ‘key’

Steps:

While(curr->data==key) do

    If(curr->next==head)

        Print “Given node is not found”

    End if

    prev=curr

    curr=curr->next

End while

If(curr->next==head) // Check if node is only node

    head=NULL

    deallocate curr

End if

Stop

Algorithm search_cll()

Input: ‘key’ is the value to be searched

Output: position of the value ‘key’ in the list

Steps:

Set index=0

current=head

while(current!=head) do // Iterate till end of list

index=index+1

current=current->next

End while

Print index

Stop

Algorithm reverse()

Input: A Circular linked list with ‘head’ pointer pointing to the first node

Output: The given list in reverse order

Steps:

If(head=NULL)

Print “Cannot reverse an empty list”

End If

last=head // Head is going to be our last node after reversing list

prev=head

cur=head->next

head=head->next

while(head!=last) do // Iterate till you reach the initial node in circular list

head=head->next

cur->next=prev

prev=cur

cur=head

End while

Cur->next=prev

Head=prev // Make last node as head

Stop

Source Code:-

```
#include<stdio.h>
#include<stdlib.h>
struct node {
    int data;
    struct node * next;
}*head=NULL;
void createList(int n)
{
    int i, data;
    struct node *prevNode, *newNode;

    if(n >= 1)
```

```
{

    //Creates and links the head node

    head = (struct node *)malloc(sizeof(struct node));

    printf("enter data of 1 node: ");
    scanf("%d", &data);

    head->data = data;
    head->next = NULL;

    prevNode = head;


     //Creates and links rest of the n-1 nodes

    for(i=2; i<=n; i++)
    {
        newNode = (struct node *)malloc(sizeof(struct node));

        printf("enter data of %d node: ", i);
```

```
            scanf("%d", &data);

            newNode->data = data;
            newNode->next = NULL;

            //Links the previous node with newly created node
            prevNode->next = newNode;
            //Moves the previous node ahead
            prevNode = newNode;
        }

        //Links the last node with first node
        prevNode->next = head;

        printf("\ncircular linked list created successfully\n");
    }
}
void displayList()
{
    struct node *current;
    int n = 1;
```

```
    if(head == NULL)
    {
        printf("list is empty.\n");
    }
    else
    {
        current = head;
        printf("data in the list:\n");

        do {
            printf("Data %d = %d\n", n, current->data);

            current = current->next;
            n++;
        }while(current != head);
    }
}
void insertAtBeginning(int data)
{
    struct node *newNode, *current;

    if(head == NULL)
```

```
{
    printf("list is empty.\n");
}
else
{

     //Creates new node, assign data and links it to head

    newNode = (struct node *)malloc(sizeof(struct node));
    newNode->data = data;
    newNode->next = head;


     //Traverses to last node and links last node
     //with first node which is new node

    current = head;
    while(current->next != head)
    {
        current = current->next;
    }
    current->next = newNode;
```

```
        //Makes new node as head node
        head = newNode;

        printf("node inserted successfully\n");
    }
}
```

```
// Inserts a new node at any position in the list
// @data Data of the new node
//@position Position where to insert new node
void insertAtN(int data, int position)
{
    struct node *newNode, *current;
    int i;

    if(head == NULL)
    {
        printf("list is empty.\n");
```

```
}
else if(position == 1)
{
    insertAtBeginning(data);
}
else
{

      //Creates new node and assign data to it

    newNode = (struct node *)malloc(sizeof(struct node));
    newNode->data = data;


      //Traverse to n-1 node

    current = head;
    for(i=2; i<=position-1; i++)
    {
        current = current->next;
    }
```

```
        // Links new node with node ahead of it and previous to it
        newNode->next = current->next;
        current->next = newNode;

        printf("node inserted successfully.\n");
    }
}
void deleteNode(struct node *head, int key)
{
    if (head == NULL)
        return;

    // Find the required node
    struct node *curr = head, *prev;
    while (curr->data != key)
    {
        if (curr->next == head)
        {
            printf("\nGiven node is not found in the list!!!");
            break;
        }
```

```
        prev = curr;
        curr = curr -> next;
    }

    // Check if node is only node
    if (curr->next == head)
    {
        head = NULL;
        free(curr);
        return;
    }
}
int search(struct node *head, int key)
{
        int index = 0;
    struct node *current = head;

    // Iterate till end of list
    do
    {
            // Nothing to look into
            if (current == NULL)
```

```
                return;

            if (current->data == key)
                return index;

        current = current->next;
            index++;
    } while (current != head);

    // Element not found in list
    printf("\nElement not found\n");
}
void reverseList(struct node *head)
{
    // Temporary helper variables
    struct node *prev, *cur, *next, *last;

    // Cannot reverse empty list
    if (head == NULL)
    {
        printf("Cannot reverse empty list.\n");
        return;
```

```
}



last = head;

prev  = head;
cur   = (head)->next;
head = (head)->next;

// Iterate till you reach the initial node in circular list
while (head != last)
{
    head = (head)->next;
    cur->next = prev;

    prev = cur;
    cur  = head;
}

cur->next = prev;
head = prev;       // Make last node as head
```

```
}
int main()
{
	int prompt,no,data,pos;
	printf("////Circular Linked List\\\\  \n\n");
	while(1)
	{
		printf("Enter any option to do the operation\n1.Create\n2.Insert at beginning\n3.Insert at given position\n4.Delete\n5.Search for an element\n6.Reverse the list\n7.Display\nEnter any other key to exit\n");
		scanf("%d",&prompt);
		switch(prompt)
		{
			case 1:
				printf("\nEnter the no. of nodes required: ");
				scanf("%d",&no);
				createList(no);
				break;
			case 2:
				printf("\nEnter the data to be inserted : ");
				scanf("%d",&data);
				insertAtBeginning(data);
```

```
                    break;
                case 3:
                    printf("\nEnter the position and data for
insertion : ");
                    scanf("%d",&pos);
                    printf("\t");
                    scanf("%d",&data);
                    insertAtN(data,pos);
                    break;
                case 4:
                    printf("\nEnter the data to be deleted : ");
                    scanf("%d",&data);
                    deleteNode(head,data);
                    break;
                case 5:
                    printf("\nEnter the number to be searched :
");
                    scanf("%d",&data);
                    search(head,data);
                    break;
                case 6:
                    reverseList(head);
                    displayList();
```

```
                    break;
                case 7:
                    displayList();
                    break;
                default:
                    exit(1);
            }
        }
        return 0;
}
```

Output:-

////Circular Linked List\\\\

Enter any option to do the operation

1.Create

2.Insert at beginning

3.Insert at given position

4.Delete

5.Search for an element

6.Reverse the list

7.Display

Enter any other key to exit

1

Enter the no. of nodes required: 5

enter data of 1 node: 1

enter data of 2 node: 2

enter data of 3 node: 3

enter data of 4 node: 4

enter data of 5 node: 5

circular linked list created successfully

Enter any option to do the operation

1.Create

2.Insert at beginning

3.Insert at given position

4.Delete

5.Search for an element

6.Reverse the list

7.Display

Enter any other key to exit

2

Enter the data to be inserted : 0

node inserted successfully

Enter any option to do the operation

1.Create

2.Insert at beginning

3.Insert at given position

4.Delete

5.Search for an element

6.Reverse the list

7.Display

Enter any other key to exit

7

data in the list:

Data 1 = 0

Data 2 = 1

Data 3 = 2

Data 4 = 3

Data 5 = 4

Data 6 = 5

Enter any option to do the operation

1.Create

2.Insert at beginning

3.Insert at given position

4.Delete

5.Search for an element

6.Reverse the list

7.Display

Enter any other key to exit

6

data in the list:

Data 1 = 0

Data 2 = 5

Data 3 = 4

Data 4 = 3

Data 5 = 2

Data 6 = 1

Enter any option to do the operation

1.Create

2.Insert at beginning

3.Insert at given position

4.Delete

5.Search for an element

6.Reverse the list

7.Display

Enter any other key to exit

7

data in the list:

Data 1 = 0

Data 2 = 5

Data 3 = 4

Data 4 = 3

Data 5 = 2

Data 6 = 1

Enter any option to do the operation

1.Create

2.Insert at beginning

3.Insert at given position

4.Delete

5.Search for an element

6.Reverse the list

7.Display

Enter any other key to exit

0

Discussions:-

- In circular linked list the immediate left node can be accesed by traversing the list
- In circular linked list there are no null links hence, program is less ambiguous.

Assignment No.12 **Date:-**

Problem Statement:- Write a program to scan a polynomial using linked list and add two polynomials

Algorithm:-

Algorithm create_poly()

Input: ‘n’ is the exponent and ‘c’ is the coefficient of a term

Output: A list which will have all the corresponding terms of the polynomial

Steps:

```
While(n!=-1) do
    If(start=NULL) then
        Allocate new memory for ‘newnode’
        newnode->num=n
        newnode->coeff=c
        newnode->next=NULL
        start=newnode
    Else
        ptr=start
        while(ptr->next!=NULL) do
            ptr=ptr->next
            newnode->num=n
            newnode->coeff=c
            newnode->next=NULL
```

ptr->next=newnode

End while

End If

End While

Stop

Algorithm display_poly()

Input: 'start' pointer of the polynomial

Output: All the terms present in the polynomial

Steps:

ptr=start

While(ptr!=NULL) do

Print ptr->num

Print ptr->coeff

Ptr=ptr->next

End while

Stop

Algorithm add_poly()

Inputs: 'start1'is the pointer to the 1st polynomial and 'start2' is the pointer to the 2nd polynomial

Output: The sum of two polynomials

Steps:

Set ptr1=start1

Set ptr2=start2

```
While(ptr1!=NULL&&ptr2!=NULL) do
        If(ptr1->coeff=ptr->coeff) then
                sum_num=ptr1->num+ptr2->num
                start3=add_node(start,sum_num,ptr->coeff)
                ptr1=ptr1->next
                ptr2=ptr2->next
        Else
                If(ptr1->coeff>ptr2->coeff) then
                        start3=add_node(start3,ptr1->num,ptr1->coeff)
                        ptr1=ptr->next
        Else
                If(ptr1->coeff<ptr2->coeff)
                        start3=(start3,ptr2->num,ptr2->coeff)
                        ptr2=ptr2->next
                End If
                End If
        End If
End while
If(ptr1=NULL) then
        While(ptr2!=NULL) do
                start3=add_node(start3,ptr1->num,ptr1->coeff)
                ptr2=ptr2->next
```

End While

End If

If(ptr2=NULL)

While(ptr1!=NULL)

start3=add_node(start3,ptr->num,ptr1->coeff)

ptr1=ptr1->next

End while

End If

Stop

In add_poly() there is a procedure called add_node(), the steps of this procedure is as follows

Procedure add_poly()

Steps:

If(start=NULL)

Allocate new memory for ‘newnode’

newnode->num=n

newnode->coeff=c

newnode->next=NULL

start=newnode

Else

ptr=start

while(ptr->next!=NULL) do

ptr=ptr->next

newnode->num=n

newnode->coeff=c

newnode->next=NULL

ptr->next=newnode

End while

End If

Stop

Source Code:-

```
#include <stdio.h>
#include <conio.h>
#include <malloc.h>
struct node
{
int num;
int coeff;
struct node *next;
};
struct node *start1 = NULL;
struct node *start2 = NULL;
struct node *start3 = NULL;
struct node *create_poly(struct node *);
```

```
struct node *display_poly(struct node *);
struct node *add_poly(struct node *, struct node *, struct node *);
struct node *add_poly(struct node *, struct node *, struct node *);

int main()
{
int option;
do
{
printf("\n******* MAIN MENU *******");
printf("\n 1. Enter the first polynomial");
printf("\n 2. Display the first polynomial");
printf("\n 3. Enter the second polynomial");
printf("\n 4. Display the second polynomial");
printf("\n 5. Add the polynomials");
printf("\n 6. Display the result");
printf("\n 7.Exit");
printf("\n\n Enter your option : ");
scanf("%d", &option);
switch(option)
{
```

```
case 1: start1 = create_poly(start1);
break;
case 2: start1 = display_poly(start1);
break;
case 3: start2 = create_poly(start2);
break;
case 4: start2 = display_poly(start2);
break;
case 5: start3 = add_poly(start1, start2, start3);
break;
case 6: start3 = display_poly(start3);
break;
}
}while(option!=7);
return 0;
}
struct node *create_poly(struct node *start)
{
struct node *new_node, *ptr;
int n, c;
printf("\n Enter the number : ");
scanf("%d", &n);
```

```
printf("\t Enter its coefficient : ");
scanf("%d", &c);
while(n != -1)
{
if(start==NULL)
{
new_node = (struct node *)malloc(sizeof(struct node));
new_node -> num = n;
new_node -> coeff = c;
new_node -> next = NULL;
start = new_node;
}
else
{
ptr = start;
while(ptr -> next != NULL)
ptr = ptr -> next;
new_node = (struct node *)malloc(sizeof(struct node));
new_node -> num = n;
new_node -> coeff = c;
new_node -> next = NULL;
ptr -> next = new_node;
```

```
}
printf("\n Enter the number : ");
scanf("%d", &n);
if(n == -1)
break;
printf("\t Enter its coefficient : ");
scanf("%d", &c);
}
return start;
}
struct node *add_node(struct node *start, int n, int c)
{
struct node *ptr, *new_node;
if(start == NULL)
{
new_node = (struct node *)malloc(sizeof(struct node));
new_node -> num = n;
new_node -> coeff = c;
new_node -> next = NULL;
start = new_node;
}
else
```

```
{
ptr = start;
while(ptr -> next != NULL)
ptr = ptr -> next;
new_node = (struct node *)malloc(sizeof(struct node));
new_node -> num = n;
new_node -> coeff = c;
new_node -> next = NULL;
ptr -> next = new_node;
}
return start;
}
struct node *display_poly(struct node *start)
{
struct node *ptr;
ptr = start;
while(ptr != NULL)
{
printf("\n%d x %d\t", ptr -> coeff, ptr -> num);
ptr = ptr -> next;
}
return start;
```

```
}
struct node *add_poly(struct node *start1, struct node *start2,
struct node *start3)
{
struct node *ptr1, *ptr2;
int sum_num, c;
ptr1 = start1, ptr2 = start2;
while(ptr1 != NULL && ptr2 != NULL)
{
if(ptr1 -> num == ptr2 -> num)
{
sum_num = ptr1 -> coeff + ptr2 -> coeff;
start3 = add_node(start3, ptr1->num, sum_num);
ptr1 = ptr1 -> next;
ptr2 = ptr2 -> next;
}
else if(ptr1 -> num > ptr2 -> num)
{
start3 = add_node(start3, ptr1 -> num, ptr1 -> coeff);
ptr1 = ptr1 -> next;
}
else if(ptr1 -> num < ptr2 -> num)
{
```

```
start3 = add_node(start3, ptr2 -> num, ptr2 -> coeff);
ptr2 = ptr2 -> next;
}
}
if(ptr1 == NULL)
{
while(ptr2 != NULL)
{
start3 = add_node(start3, ptr2 -> num, ptr2 -> coeff);
ptr2 = ptr2 -> next;
}
}
if(ptr2 == NULL)
{
while(ptr1 != NULL)
{
start3 = add_node(start3, ptr1 -> num, ptr1 -> coeff);
ptr1 = ptr1 -> next;
}
}
return start3;
}
```

Output:-

******* MAIN MENU *******

1. Enter the first polynomial
2. Display the first polynomial
3. Enter the second polynomial
4. Display the second polynomial
5. Add the polynomials
6. Display the result
7.Exit

Enter your option : 1

Enter the number : 1

Enter its coefficient : 5

Enter the number : 2

Enter its coefficient : 6

Enter the number : -1

******* MAIN MENU *******

1. Enter the first polynomial
2. Display the first polynomial
3. Enter the second polynomial
4. Display the second polynomial
5. Add the polynomials
6. Display the result
7.Exit

Enter your option : 2

5 x 1

6 x 2

******* MAIN MENU *******

1. Enter the first polynomial
2. Display the first polynomial
3. Enter the second polynomial
4. Display the second polynomial
5. Add the polynomials
6. Display the result
7.Exit

Enter your option : 3

Enter the number : 1

Enter its coefficient : 6

Enter the number : 2

Enter its coefficient : 7

Enter the number : -1

******* MAIN MENU *******

1. Enter the first polynomial
2. Display the first polynomial
3. Enter the second polynomial
4. Display the second polynomial
5. Add the polynomials
6. Display the result
7.Exit

Enter your option : 4

6 x 1

7 x 2

******* MAIN MENU *******

1. Enter the first polynomial
2. Display the first polynomial
3. Enter the second polynomial
4. Display the second polynomial
5. Add the polynomials
6. Display the result
7. Exit

Enter your option : 5

******* MAIN MENU *******

1. Enter the first polynomial
2. Display the first polynomial
3. Enter the second polynomial
4. Display the second polynomial
5. Add the polynomials
6. Display the result
7. Exit

Enter your option : 6

11 x 1

13 x 2

******* MAIN MENU *******

1. Enter the first polynomial
2. Display the first polynomial
3. Enter the second polynomial
4. Display the second polynomial
5. Add the polynomials
6. Display the result
7. Exit

Enter your option : 7

Discussions:-

- In this representation each node has three part one part contains the exponent number, one part contains the coefficents and one part contains the link to another term
- A polynomial can be represented in an array but for more dynamic approach linked list is widely used for it's application

Assignment No.13 **Date:-**

Problem Statement:- Perform Stack operations using Linked List implementation.

Algorithm:-

Algorithm for Push:

Algorithm push()

Input: Integer data ‘dat’ to be pushed into the stack

Output: Stack with the pushed value

Steps:

Set top=NULL //It is a pointer for the stack

Allocate memory for temp

If(temp!=NULL) then

    Print “Stack Overflow”

Else

    (temp->data)=dat

    (temp->link)=top

    top=temp

End If

Stop

Algorithm for pop:

Algorithm pop()

Output: The data which is pushed last in the stack

Steps:

If(top=NULL) then

Print “Stack Underflow”

Else

Set p=top // p is the pointer to hold the position of pop

k=(top->data) //k holds the data which is to be popped

top=(top->link)

deallocate p

Print k

End If

Stop

Source Code:

```
// linked impelementation of stack
#include<stdio.h>
#include<stdlib.h>
typedef struct node{
    int data;
    struct  node *link;
}stack; //declaring a new stack data type
stack *top=NULL;
stack *create_node() //create_node() is for allocating memory for the stack
{
```

```
        stack *k;
        k=(stack*)malloc(sizeof(stack));
        return k;
}
void push(int x)
{
        stack *temp=create_node();
        if(temp==NULL)
        {
                printf("Stack Overflow\nPush Failed\n");
                return;
        }
        temp->data=x;
        temp->link=top; //pusing the data into the stack
        top=temp;
}
int pop()
{
        if(top==NULL)
        {
                printf("Stack Underflow\nPop Failed\n");
        }
```

```
        int k;
        stack *p=top;
        k=top->data; //k holds the popped value
        top=top->link;
        free(p); //deallocating the popped node
        return k;
}
int main()
{
        int prompt,dat;
        while(1)
        {
        printf("\nEnter the choice of opertaion\n1.Push\n2.Pop\n");
        scanf("%d",&prompt);
        switch(prompt)
        {
                case 1:
                        {
                        printf("Enter the data to push into the stack\t");
                        scanf("%d",&dat);
                        push(dat);
                        printf("\n%d is pushed to the stack",dat);
```

```
                    break;
                    }
                case 2:
                    {
                    printf("\nPopped Element is %d",pop());
                    break;
                    }
        }
        }
        return 0;
}
```

Output:-

```
Enter the choice of opertaion
1.Push
2.Pop
1
Enter the data to push into the stack   23

23 is pushed to the stack
Enter the choice of opertaion
1.Push
2.Pop
```

1

Enter the data to push into the stack 23

23 is pushed to the stack

Enter the choice of opertaion

1.Push

2.Pop

2

Popped Element is 23

Enter the choice of opertaion

1.Push

2.Pop

1

Enter the data to push into the stack 33

33 is pushed to the stack

Enter the choice of opertaion

1.Push

2.Pop

2

Popped Element is 33

Enter the choice of opertaion

1.Push

2.Pop

2

Popped Element is 23

Enter the choice of opertaion

1.Push

2.Pop

2

Stack Underflow

Pop Failed

Discussions:

Stack is a linear data structure which follows a particular order in which the operations are performed. Stack follows a LIFO (Last In First Out) order of operations. Stack can be implemented with arrays and linked lists both and complexity of push() and pop() operations are always O(1) because it doesn't require any traversals to perform the operations.

Assignment No.14 **Date;-**

Problem Statement:- Perform Stack operations using array implementation.

Algorithm:-

Algorithm for Push:

Algorithm Push()

Input: An array ‘stack’, a variable ‘top’ which points to the indeces and ‘ITEM’ is value to be inserted

Output: An array with the pushed(inserted) value

Steps:

If(top=MAXSIZE) then

    Print “Stack Overflow” //stack is full

    Exit

Else

    top=top+1 //top is increased

    stack[top]=ITEM

End If

Stop

Algorithm for Pop:

Algorithm Pop()

Output: the value at the ‘top’ position of array

Steps:

If(top=0)

Print “Stack Underflow” //stack is empty

Exit

Else

x=stack[top]

top=top-1 //top is decreased

End If

Print x //popped value is printed

Stop

Source Code:-

```
// stack using array
#include<stdio.h>
#include<stdlib.h>
#define MAX_SIZE 50 // maxsize of array is determined
int stack[MAX_SIZE];
int top=-1;
void push(int data)
{
    if(top==MAX_SIZE-1) //top==MAX_SIZE-1 means stack is maxed out hence push not possible
```

```
	{
		printf("Stack Overflow\nPush Failed");
		return;
	}
	stack[++top]=data; //first top is incremented then the data is inserted to stack[top] position
}
int pop()
{
	if(top==-1) //top==-1 means stack is empty pop is not possible
	{
		printf("Stack Underflow\nPop Failed");
		return;
	}
	return stack[top--]; // here first stack[top element is returned then top is decremented
}
int main()
{
	int prompt,dat;
	while(1)
	{
```

```
        printf("\nEnter the choice of
opertaion\n1.Push\n2.Pop\n**Press any other key to exit**\n");
        scanf("%d",&prompt);
        switch(prompt)
        {
                case 1:
                        {
                        printf("Enter the data to push into the stack\t");
                        scanf("%d",&dat);
                        push(dat);
                        printf("\n%d is pushed to the stack",dat);
                        break;
                        }
                case 2:
                        {
                        printf("\nPopped Element is %d",pop());
                        break;
                        }
                default:
                        exit(1);
        }
        }
}
```

Output:-

Enter the choice of opertaion

1.Push

2.Pop

Press any other key to exit

1

Enter the data to push into the stack 2

2 is pushed to the stack

Enter the choice of opertaion

1.Push

2.Pop

Press any other key to exit

1

Enter the data to push into the stack 3

3 is pushed to the stack

Enter the choice of opertaion

1.Push

2.Pop

Press any other key to exit

1

Enter the data to push into the stack 4

4 is pushed to the stack

Enter the choice of opertaion

1.Push

2.Pop

Press any other key to exit

1

Enter the data to push into the stack 5

5 is pushed to the stack

Enter the choice of opertaion

1.Push

2.Pop

Press any other key to exit

2

Popped Element is 5

Enter the choice of opertaion

1.Push

2.Pop

Press any other key to exit

2

Popped Element is 4

Enter the choice of opertaion

1.Push

2.Pop

Press any other key to exit

1

Enter the data to push into the stack 6

6 is pushed to the stack

Enter the choice of opertaion

1.Push

2.Pop

Press any other key to exit

2

Popped Element is 6

Enter the choice of opertaion

1.Push

2.Pop

Press any other key to exit

2

Popped Element is 3

Enter the choice of opertaion

1.Push

2.Pop

Press any other key to exit

2

Popped Element is 2

Enter the choice of opertaion

1.Push

2.Pop

Press any other key to exit

2

Stack Underflow

Pop Failed

Discussions:-

- Stack is a Last-In-First-Out type of data where the last inserted data is extracted first
- Here Push and Pop operation has a time complexity of O(1)

Assignment No.15 **Date:-**

Problem Statement:- Perform Queue operations using circular array implementation

Algorithm:-

Algorithm for enQueue:

Input: A queue 'ITEM' with 'front' and 'rear' two variables which the location and element which have the value to be inserted.

Output: A queue with the inserted element

Steps:

If((front=rear+1) or (front=0 and rear=MAXSIZE)) then
//MAXSIZE is the maximum size of the queue

    Print "Queue Overflow" //Queue is full

    Exit

Else

    rear=(rear+1)%MAXSIZE //rear is increased

End If

ITEM[rear]=element

Stop

Algorithm for deQueue:

Output: The value of the element in 'front' position

Steps:

If(front=0)

    Print "Queue Underflow"

    Exit

Else

    y=ITEM[front]

    front=(front+1)%MAXSIZE

End If

Print y

Stop

Source Code:-

```
#include <stdio.h>

#define MAXSIZE 5

int items[MAXSIZE];
int front = -1, rear =-1; //initialised to -1

int isFull()
```

```
{
    if( (front == rear + 1) || (front == 0 && rear == SIZE-1))
        return 1;
    return 0;
}

int isEmpty()
{
    if(front == -1) return 1;
    return 0;
}

void enQueue(int element)
{
    if(isFull()) printf("\n Queue is full!! \n");
    else
    {
        if(front == -1) front = 0;
        rear = (rear + 1) % SIZE;
        items[rear] = element;
        printf("\n Inserted -> %d", element);
    }
}
```

```
int deQueue()
{
    int element;
    if(isEmpty()) {
        printf("\n Queue is empty !! \n");
        return(-1);
    } else {
        element = items[front];
        if (front == rear){
            front = -1;
            rear = -1;
        } /* Q has only one element, so we reset the queue after dequeing it. ? */
        else {
            front = (front + 1) % SIZE;

        }
        printf("\n Deleted element -> %d \n", element);
        return(element);
    }
}
```

```
int main(){
	int prompt,dat;
	while(1)
	{
	printf("\nEnter the choice of opertaion\n1.Insert\n2.Delete\n**Press any other key to exit**\n");
	scanf("%d",&prompt);
	switch(prompt)
	{
		case 1:
			{
			printf("Enter the data to insert in queue\t");
			scanf("%d",&dat);
			enQueue(dat);
			printf("\n%d is entered in the queue",dat);
			break;
			}
		case 2:
			{
			printf("\nDeleted Element is %d",deQueue());
			break;
			}
```

```
            default:
                    exit(1);
    }       } }
```

Output:-

```
Enter the choice of opertaion
1.Insert
2.Delete
**Press any other key to exit**
1
Enter the data to insert in queue       1

 Inserted -> 1
1 is entered in the queue
Enter the choice of opertaion
1.Insert
2.Delete
**Press any other key to exit**
1
Enter the data to insert in queue       2

 Inserted -> 2
2 is entered in the queue
```

Enter the choice of opertaion

1.Insert

2.Delete

Press any other key to exit

1

Enter the data to insert in queue 3

Inserted -> 3

3 is entered in the queue

Enter the choice of opertaion

1.Insert

2.Delete

Press any other key to exit

1

Enter the data to insert in queue 4

Inserted -> 4

4 is entered in the queue

Enter the choice of opertaion

1.Insert

2.Delete

Press any other key to exit

```
1
Enter the data to insert in queue       5

 Inserted -> 5
5 is entered in the queue
Enter the choice of opertaion
1.Insert
2.Delete
**Press any other key to exit**
1
Enter the data to insert in queue       6

 Queue is full!!

6 is entered in the queue
Enter the choice of opertaion
1.Insert
2.Delete
**Press any other key to exit**
2

 Deleted element -> 1
```

Deleted Element is 1

Enter the choice of opertaion

1.Insert

2.Delete

Press any other key to exit

2

Deleted element -> 2

Deleted Element is 2

Enter the choice of opertaion

1.Insert

2.Delete

Press any other key to exit

2

Deleted element -> 3

Deleted Element is 3

Enter the choice of opertaion

1.Insert

2.Delete

Press any other key to exit

2

Deleted element -> 4

Deleted Element is 4

Enter the choice of opertaion

1.Insert

2.Delete

Press any other key to exit

2

Deleted element -> 5

Deleted Element is 5

Enter the choice of opertaion

1.Insert

2.Delete

Press any other key to exit

2

Queue is empty !!

Discussions:-

- Queue is a data type which operates on First-In-First-Out System
- In a Circular Queue the array is treated circularly so that the remaining memory can be used
- As it is considered circular hence the terminating conditions are slightly different from a normal queue data structure.

Assignment No.16 **Date:-**

Problem Statement:- Perform Queue operations using Array and Linked List implementation.

Algorithm:-

Algorithms for Array:

Algorithm enQueue_arr()

Input: An array ‘Queue’ with variables ‘front’ and ‘rear’ pointing the position and ‘ITEM’ is the data to be inserted

Output: Array with the inserted element

Steps:

If(rear=MAXSIZE) then //MAXSIZE is the maximum size of the list

Print “Queue Overflow” //Queue is full

Exit

Else

If(front=0)

front=1 //front is initialised to 1 for deQueue

End If

End If

rear=rear+1 //rear is increased

Queue[rear]=ITEM //ITEM is placed

Stop

Algorithm deQueue_arr()

Output: The item at the ‘front’ position of the list

Steps:

```
If(front=0)
        Print “Queue Underflow” //Queue is empty
        Exit
Else
        If(front=rear) //only one item is remaining in the list
                front=0
                rear=0 //reset to initial position
        End If
End If
h=Queue[front]
front=front+1
Print h
Stop
```

Algorithm for Linked List:

Algorithm enQueue_ll()

Input: A linked list called ‘Queue’ with ‘front’ and ‘rear’ two pointer initialised to NULL and the ‘ITEM’ is the data to insert

Output: A linked list with the inserted data

Steps:

Allocate memory for ‘temp’

If(temp=NULL) then

Print “Queue Overflow” //Queue is full

Exit

End If

temp->data=ITEM //putting the ITEM into the data part of the node

temp->link=NULL

If(front=NULL)

front=temp

rear=temp

Else

rear->link=temp

rear=rear->link

End If

Stop

Algorithm deQueue_ll()

Output: The item where ‘front’ is positioned

Steps:

If(front=NULL) then //if frony is not initialised yet

Print “Queue Underflow” //Queue is empty

Exit

End If

h=front->data

If(front=rear) then //only one item left

front=0

rear=0

End If

front=front->link

Print h

Stop

Source Code:-

Source Code for Array:

```
// queue using array
#include<stdio.h>
#include<stdlib.h>
#define MAX_SIZE 50 // maxsize of array is determined
int queue[MAX_SIZE];
int front=-1,rear=-1;
void EnQueue(int data)
{
    if(rear==MAX_SIZE-1) //if rear is at the max position execution will be failed
    {
```

```
        printf("Queue Overflow\nEnqueue operation failed\n");
    }
    if(rear==-1) //if rear and front both are not initialised then both them will be incremented for the first time
    {
        ++front;
    }
    queue[++rear]=data; //data is entered in the queue
}
int DeQueue()
{
    if(front==-1) //if front is -1 then there's no value to be deleted
    {
        printf("Queue Underflow\nDequeue operation failed\n");
    }
    int val=queue[front]; //deleted element is stored in val
    if(front==rear)
    {
        front=rear=-1; //if fornt reaches to rear that means all of the elements are deleted hence they're started again
    }
```

```
        else
        {
                ++front; //if they are not at the same place then front
will be incremented
        }
        return  val; //deleted value is returned
}
int main()
{
        int prompt,dat;
        while(1)
        {
        printf("\nEnter the choice of
opertaion\n1.Insert\n2.Delete\n**Press any other key to
exit**\n");
        scanf("%d",&prompt);
        switch(prompt)
        {
                case 1:
                        {
                        printf("Enter the data to insert in queue\t");
                        scanf("%d",&dat);
                        EnQueue(dat);
```

```
                printf("\n%d is entered in the queue",dat);
                break;
                }
        case 2:
                {
                printf("\nDeleted Element is %d",DeQueue());
                break;
                }
        default:
                exit(1);
    }
    }
}
```

Source Code for Linked List:

```
//queue using linked list
#include<stdio.h>
#include<stdlib.h>
typedef struct record{
    int data;
    struct record *link;
}queue;
queue *front=NULL,*rear=NULL;
```

```
queue *create()
{
	queue *p;
	p=(queue*)malloc(sizeof(queue));
	return p;
}
void EnQueue(int x)
{
	queue *temp=create();
	if(temp==NULL) //if temp is not allocated then enQueue is not possible
	{
		printf("Queue Overflow\nEnqueue operation failed\n");
		return;
	}
	temp->data=x;
	temp->link=NULL;
	if(rear==NULL) //If rear is not yet initialised then initialise both front and rear
	{
		front=temp;
		rear=temp;
```

```
        }
        else
        {
                rear->link=temp;
                rear=rear->link; //sending rear to the next node
        }
}
int DeQueue()
{
        if(front==NULL) //if front is NULL then queue is empty
        {
                printf("Queue Underflow\nDequeue operation
failed\n");
                return;
        }
        int val=front->data; //data part is extracted
        if(front==rear) //if last element has been extracted then
front and rear will be set beck to initial stage
        {
                front=NULL;rear=NULL;
        }
        else
        {
```

```
            front=front->link; //front is pointed to the next node
        }
        return val;
}
int main()
{
        int prompt,dat;
        while(1)
        {
        printf("\nEnter the choice of
opertaion\n1.Insert\n2.Delete\n**Press any other key to
exit**\n");
        scanf("%d",&prompt);
        switch(prompt)
        {
                case 1:
                        {
                        printf("Enter the data to insert in queue\t");
                        scanf("%d",&dat);
                        EnQueue(dat);
                        printf("\n%d is entered in the queue",dat);
                        break;
                        }
```

```
            case 2:
                {
                printf("\nDeleted Element is %d",DeQueue());
                break;
                }
            default:
                exit(1);
        }
        }
}
```

Output:-

```
Enter the choice of opertaion
1.Insert
2.Delete
**Press any other key to exit**
1
Enter the data to insert in queue       2

2 is entered in the queue
Enter the choice of opertaion
1.Insert
2.Delete
```

Press any other key to exit

1

Enter the data to insert in queue 3

3 is entered in the queue

Enter the choice of opertaion

1.Insert

2.Delete

Press any other key to exit

2

Deleted Element is 2

Enter the choice of opertaion

1.Insert

2.Delete

Press any other key to exit

2

Deleted Element is 3

Enter the choice of opertaion

1.Insert

2.Delete

Press any other key to exit

2

Queue Underflow

Dequeue operation failed

Discussions:-

- Queue is a data type which works on First-In-First-Out Principle.
- Queue is widely used in CPU Scheduling, Asynchronous data transfer.

Assignment No.17 **Date:-**

Problem Statement:- Write a program to reverse the order of the elements in the stack using additional stack

Algorithm:-

Algorithm rev_st()

Input: A stack array ‘stack1’ with data and ‘top1’ is the pointer to the stack.

Output: The order of array will be in reverse

Steps:

Initialise top1=0,top2=0,top3=0

While(top1>0) do

top2=top2+1 //top2 is the pointer to additional stack

stack2[top2]=stack1[top1] ///placing all the elements in top1

top1=top1-1 //decreasing top

End while

While(top2>0)

top3=top3+1 //top3 is the pointer to another additional stack

stack3[top3]=stack2[top2] //placing all the elements in top3

top2=top2-1

End while

While(top3>0)

top1=top1+1 // again initialising top1

stack1[top1]=stack3[top3] //putting back to stack1

top3=top3-1

End while

Display stack1

Stop

Source Code:

```
#include<conio.h>
 #include<stdio.h>
 void reverse(int [],int [],int [],int *,int *,int *);
 void display(int [],int);
 void input(int [],int *,int);
 void main()
 {
 int total;
 int item,t,i,stack1[100],stack2[100],stack3[100];
 int top_1=-1,top_2=-1,top_3=-1;
 clrscr();
 printf("Enter size of stack::");
 scanf("%d",&total);
 input(stack1,&top_1,total);
 display(stack1,top_1);
 reverse(stack1,stack2,stack3,&top_1,&top_2,&top_3);
```

```
printf("\nAfter reverse.....");
display(stack1,top_1);
getch();
}
void input(int stack[],int *top,int size)
{
int i,item;
for(i=0;i<size;i++)
{
 *top=*top+1;
 printf("Enter value of for position %d ::",*top);
 scanf("%d",&item);
 stack[*top]=item;
}
}
void reverse(int stack1[],int stack2[],int stack3[],int *t1,int *t2,int
*t3)
{
while(*t1>-1)
{
 *t2=*t2+1;
 stack2[*t2]=stack1[*t1];
 *t1=*t1-1;
```

```
}
while(*t2>-1)
{
 *t3=*t3+1;
 stack3[*t3]=stack2[*t2];
 *t2=*t2-1;
}
while(*t3>-1)
{
 *t1=*t1+1;
 stack1[*t1]=stack3[*t3];
 *t3=*t3-1;
}
}
void display(int stack[],int top)
{
int i;
while(top>-1)
{
 printf("\nValue at %d is %d",top,stack[top]);
 top=top-1;
}
```

```
}
```

Output:-

```
Enter size of stack::3
Enter value of for position 0 ::1
Enter value of for position 1 ::2
Enter value of for position 2 ::3

Value at 2 is 3
Value at 1 is 2
Value at 0 is 1
After reverse.....
Value at 2 is 1
Value at 1 is 2
Value at 0 is 3
```

Discussions:

- The reverse is implemented with two additional stacks.
- Additional queue can be implemented for the reverse

Assignment No.18 **Date:-**

Problem Statement:- Write a program to reverse the order of a stack using additional queue

Algorithm:-

Algorithm st_rev_qu()

Input: A stack ‘st’ with a pointer to the stack called ‘top’

Output: Elements of the stack will be in reverse order

Steps:

Initialise top=0,f=0,r=0 //f and r are queue pointers

Set f=1 //Initialise for dequeue

While(t>0)

    r=r+1

    qu[r]=stack[t] // placing the values in queue

    t=t-1

End while

While(f<=r)

    t=t+1

    stack[t]=qu[f] //replacing again in the stack

    f=f+1

End while

Display stack

Stop

Source Code:-

```
#include<conio.h>
 #include<stdio.h>
 #define MAX 20
 void show(int stack[],int size,int top)
 {
 int i;
 for(i=0;i<size;i++)
 {
  printf("\nValue at %d is %d",top,stack[top]);
  top=top-1;
 }
 }
 void reverse(int stack[],int qu[],int *t,int *r,int *f)
 {
 *f=0;
 while(*t>-1)
 {
  *r=*r+1;
  qu[*r]=stack[*t];
  *t=*t-1;
 }
```

```
while(*f<=*r)
{
 *t=*t+1;
 stack[*t]=qu[*f];
 *f=*f+1;
}
}
void main()
{
int size;
int item,t,i,stack[MAX],quee[MAX];
int top=-1,front=-1,rear=-1;
clrscr();
printf("Enter size of stack::");
scanf("%d",&size);
for(i=0;i<size;i++)
{
 top=top+1;
 printf("Enter value of for position %d ::",top);
 scanf("%d",&item);
 stack[top]=item;
}
```

```
show(stack,size,top);
reverse(stack,quee,&top,&rear,&front);
printf("\nAfter reverse.....");
show(stack,size,top);
getch();
}
```

Output:-

Enter size of stack::3

Enter value of for position 0 ::3

Enter value of for position 1 ::2

Enter value of for position 2 ::1

Value at 2 is 1

Value at 1 is 2

Value at 0 is 3

After reverse.....

Value at 2 is 3

Value at 1 is 2

Value at 0 is 1

Discussions:-

- The additional queue is implemented to do the reverse

Name : Sayani Mondal

University Reg. No. : 057 – 1212 - 0252 - 20

University Roll No. : 203057 – 11 – 0041

Subject : Chemistry Assignment

Honours or general : Honours

Total no. of page :06

Male \ Female : Female

Contact no. : 9330142183

Soyani Mondal
Chemistry assignment

C.U. reg. no: 057-1212-0252-20 C.U. roll no.: 203057-11-0041

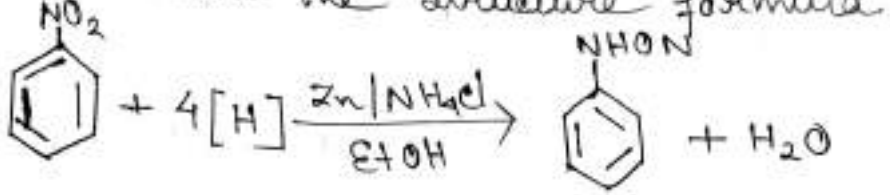
1) Arrange aniline, N-methyl aniline and N,N-dimethyl aniline in increasing order of basicity?

⇒ N-N dimethyl aniline > N-methyl aniline > aniline

when, hydrogen atoms of the amino group of arylamines are replaced by electron donating alkyl group, the basic character of the resulting amine increases.

2) $C_6H_5NO_2 \xrightarrow[EtOH]{Zn/NH_4Cl} [D]$

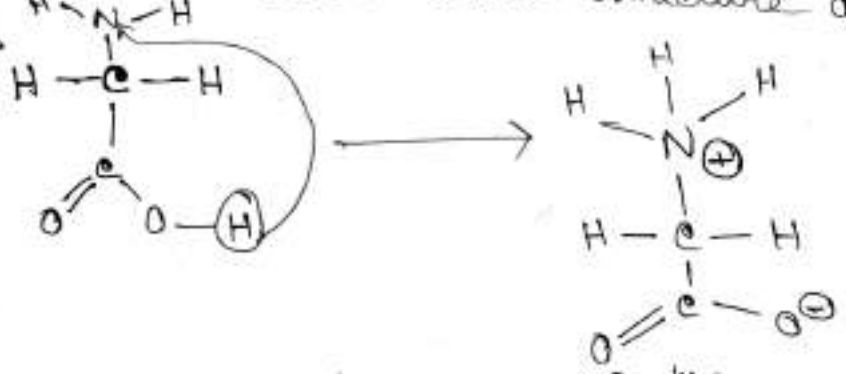
write down the structure formula for [D]

⇒ 

N-Phenyl hydroxylamine

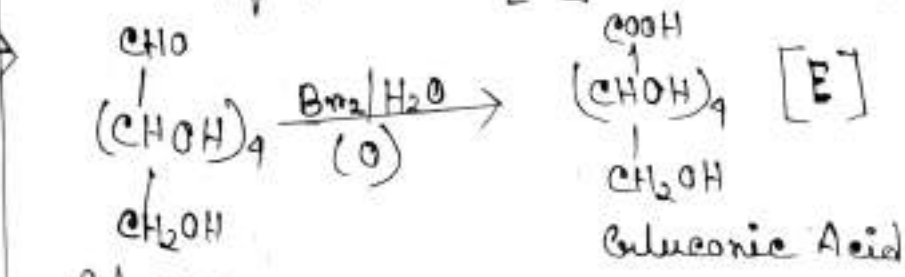
structural formula C_6H_7NO

3) Give the zwitterionic structure of glycine:

⇒ 

4) $OHC-(CHOH)_4-CH_2OH \xrightarrow{Br_2/H_2O} [E]$

Identify product [E]

⇒ 

Glucose

Gluconic Acid

5)  $\xrightarrow{KI/\Delta}$ [F]

write down the structural formula (F)

⇒  $\xrightarrow{KI/\Delta}$ C_6H_5I (F)

6) What's Peptide bond?

⇒ Peptide bond is a type of covalent bond formed between two amino acid.

7) What do you mean by isoelectric point of an amino acid?

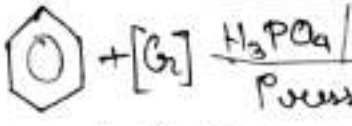
⇒ Isoelectric point is pH at which a molecule carries no net electrical charge or is electrically neutral in statistical mean.

8) Write short note on Schotten-Baumann reaction?

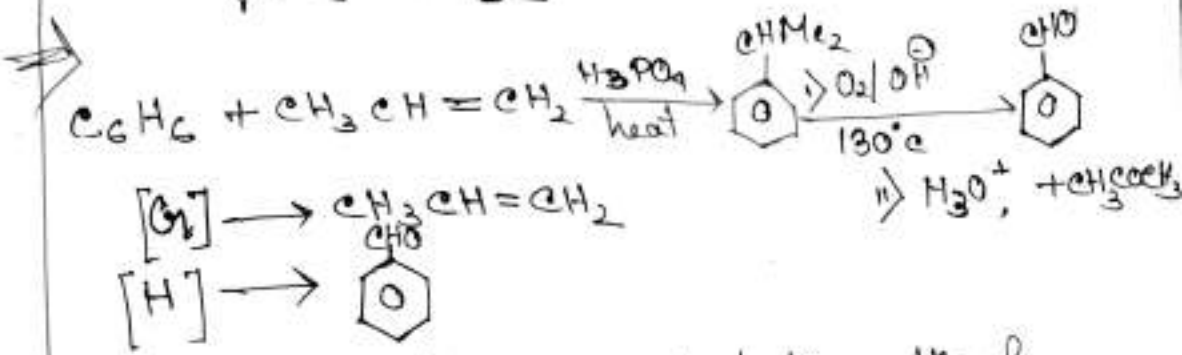
⇒ The Schotten-Baumann reaction is a method to synthesize amide from amines acid chlorides.

It also refer to the conversion of acid chloride to esters. The reaction was first described in 1883 by German chemist Carl Schotten and Eugen Baumann. The name 'Schotten-Baumann' reaction conditions often indicate the use of two-phase solvent system consisting of water and an organic solvent.

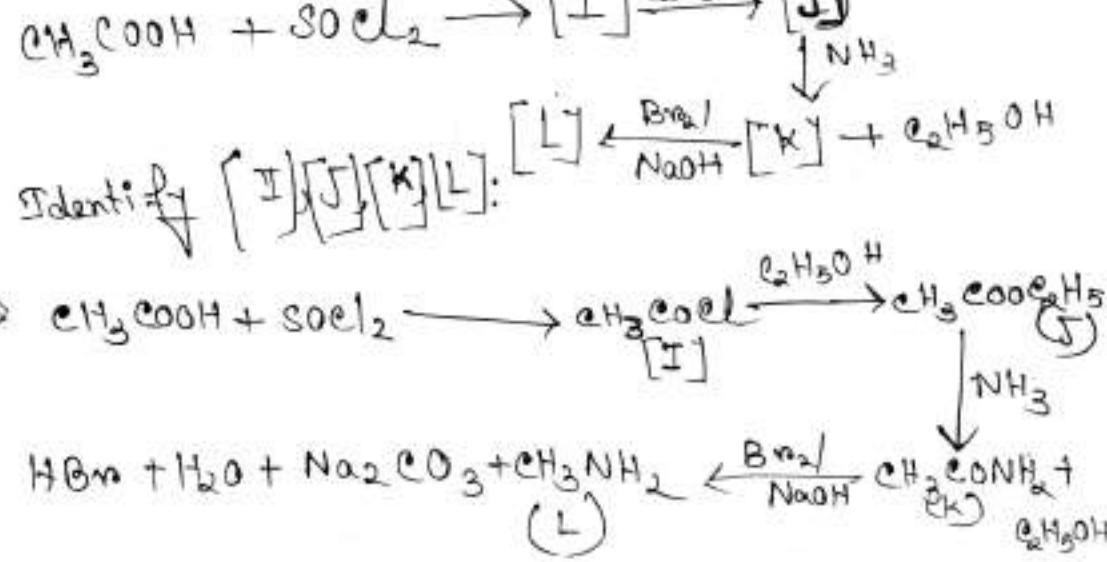
The base within the water phase neutralizes the acid, generated in the reaction while the starting materials and product remain in the organic phase, often dichloromethane diethyl ether.

9) 

Identify [G] and [H]



10) What happens when an alcoholic sol^n of benzaldehyde is boiled with a small amount sodium cyanide (NaCN)? Write down the reac^n involves?

11) 

Identify [I][J][K][L]:

12) Write a short note on diazocoupling reac^n

⇒ Diazonium salts acts as an electrophile and brings about substitution electron-rich aromatic rings such as phenol and amines. The result is the formation of dye

The compounds containing N=N - groups which are known as diazo compounds. Their general structure is R−N=N−R' where R and R' are probably aromatic groups and the azo group is thus stablised by becoming part of extended delocalised system. They are prepared by coupling reaction between a diazonium salt and a coupling reagent. E.g : Phenol reacts with benzene diazonium chloride is given a yellow - orange azo compound. The reaction is base-catalysed.

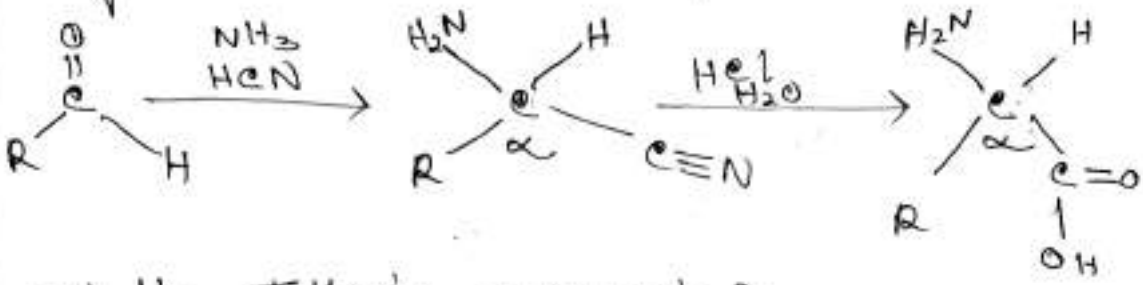
13) What do you mean by mutarotation of glucose?

⇒ If α-d-glucose dissolves in water, the rotation of the solution slowly changes from the initial value of +18.7° to the same equilibrium value of +54°. This gradual change in rotation to an equilibrium point is known as mutarotation of glucose.

14) How alanine can be prepared by Strecker's synthesis? Write down the reaction involved?

⇒ The reaction begins with the addition of cyanide ion to an imine, forming an alpha-amino nitrile. This is then hydrolyzed to give an alpha amino acid. By varying the R- group

on the whole, a wide variety of amino acids may be made this way.



15) What's Tollen's reagent?

⇒ Tollen's reagent is a ammonical mixture of silver nitrate. On heating this reagent with aldehydes form silver mirror of the wall of test tube. So, it's also called silver mirror that,

Reaction:

$$AgNO_3 + NH_4OH \longrightarrow AgOH + NH_4NO_3$$

$$AgOH + 2NH_4NO_3 \longrightarrow [Ag(NH_3)_2]OH + 2H_2O$$

$$2[Ag(NH_3)_2]OH + R-CHO \longrightarrow R-COONH_4 + 2Ag + H_2O + 3NH_3$$

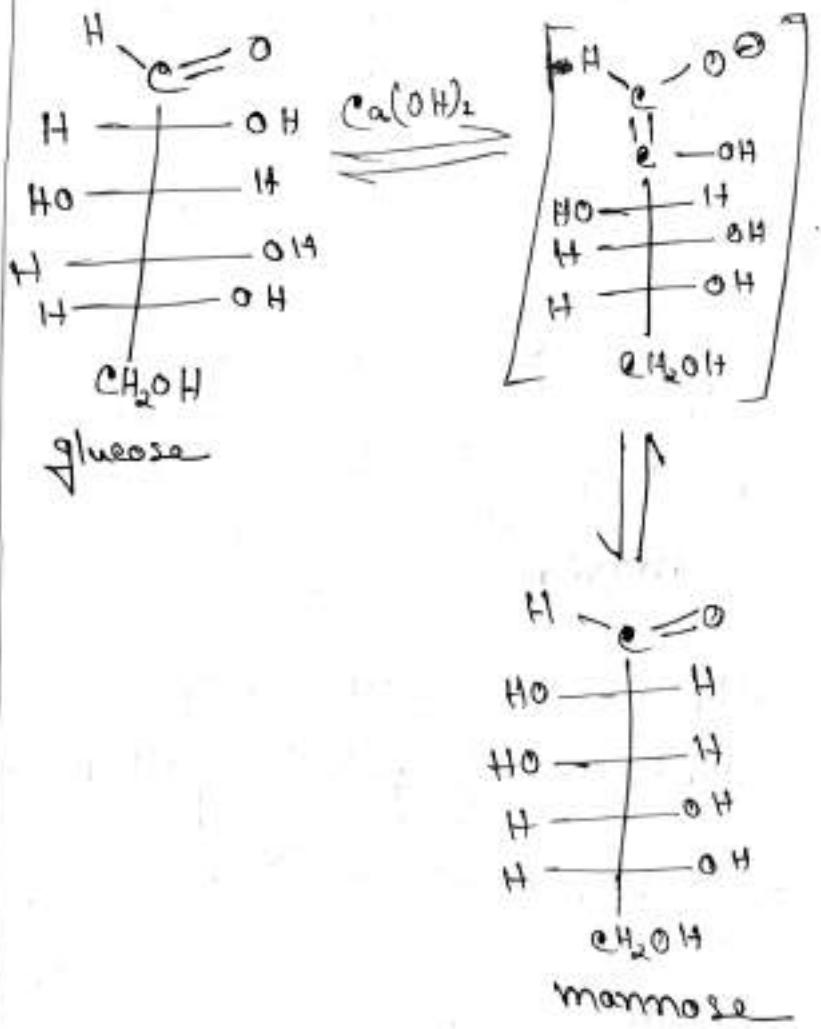
↓ Tollen's reagent

16) Write short notes on i) Epimerization and ii) Osazone formation reaction of carbohydrate:

⇒ i) Epimerization → It's the process by which two compounds formed are epimers that's two compound suffer in configuration only at one chiral centre such configuration of two compounds differ in configuration at C-1 then

they are called as anomers.

Eg.:



1) Describe spikelet inflorescene.

→ অনুমঞ্জরী, ছোটো মঞ্জুরিদণ্ড বিশিষ্ট একপ্রকার মঞ্জুরীবিন্যাস। এর নীচের দিকে কয়েকটি বিশেষ প্রকারের মঞ্জুরীপত্র থাকে। একে বর্মাকার মঞ্জুরীপত্র বলে। পুষ্পের সংখ্যা অনুযায়ী মঞ্জুরীদণ্ডের উপর আরও একটি, দুটি অথবা অনেকগুলি বর্মাকার মঞ্জুরীপত্র থাকে। এদের কক্ষে পুষ্প উৎপন্ন হয় বলে এদের লেমা বলে। অপুষ্পক বর্মাকার মঞ্জুরী পত্রের আয়ানা উপরে ও বিপরীত দিকে অপেক্ষাকৃত ক্ষুদ্র একটি দ্বি-শিরাযুক্ত বর্মাকার মঞ্জুরী পত্রিকা থাকে। একে প্যালিয়া বলে।

উদা:- ধান (Oryza sativa)

2) write down the floral formula of Poaceae and orchidaceae.

→ Poaceae :-

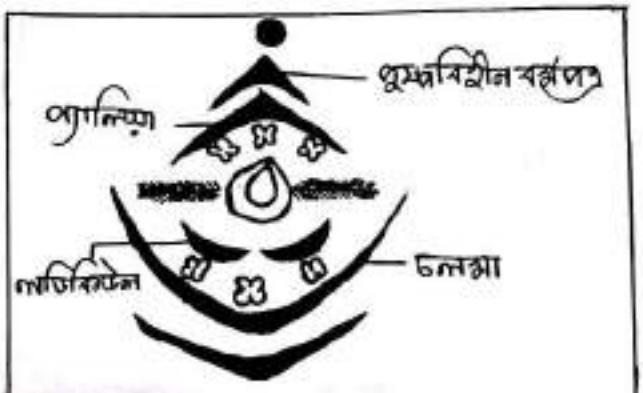
পুষ্প সংকেত – $\oplus \♀ P_0$ or 2-3, A_3 or 3+3, $\underline{G_1}$ or $\underline{(3)}$

Orchidaceae :-

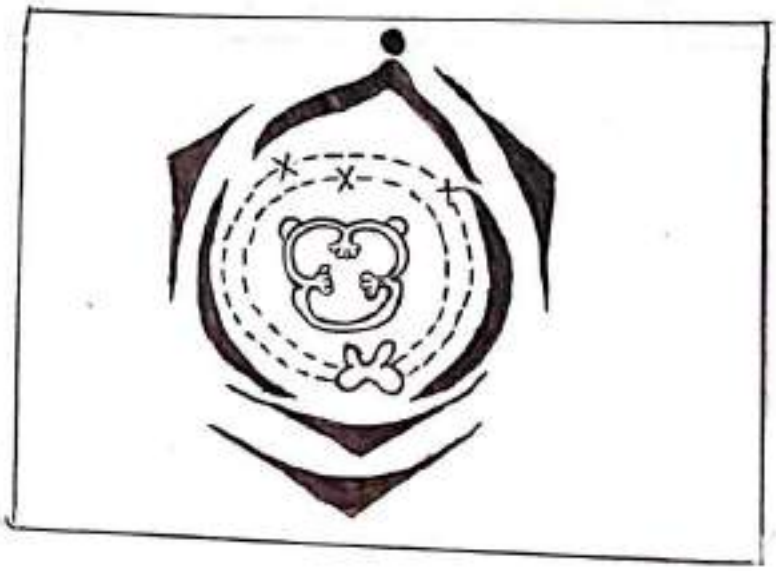
পুষ্পসংকেত - $\oplus \♀ P_{3+3}, A_{1,}$ or 2. $\overline{G_{(3)}}$

3) Draw the floral diagram of Poaceae and orchidaceae.

→ Poaceae :-



Orchidaceae:-



4) Name the most advanced family in the world.

→ Orchidaceae

5) Name the most advanced monocot family?

→ Orchidaceae.

6) Name the most advanced dicot family?

→ Asteraceae.

Debraj Roy Roll No. 07

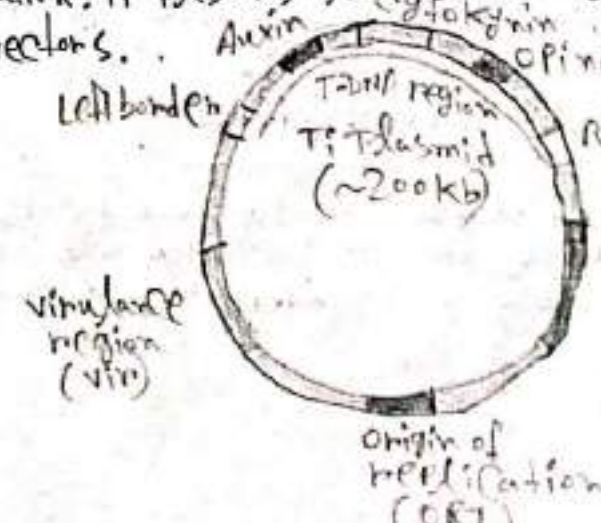
Agrobacterium-mediated gene transfer

- This is vector based gene transfer method which is found to be the most successful plant transformation system. The vectors are Ti plasmids of Agrobacterium tumefaciens - a soil borne.

If a plant is wounded or damaged, the bacterium infects through the wound site and a tumour develops at the junction of root and stem. The agent responsible for crown gall is a plasmid present in A. tumefaciens known as the Ti plasmid (Ti stands for tumour-inducing). During infection, a small portion of Ti plasmid is transferred from the bacterium into the plant cells, where it becomes covalently integrated into the genome of the host plant.

The T-DNA carries the genes that encode proteins responsible for both hormones biosynthesis and biosynthesis of novel plant metabolites, called opines (most common are octopine and nopaline) - amino acid derivatives and agropines. The production of auxin and cytokinin enhances cell proliferations and hence crown galls are formed. The opines and agropines produced by the proliferating cells are used as sole carbon/nitrogen source for inducing Agrobacterium strain.

- **The Ti plasmid:** - The Ti plasmid (~200kb) of A. tumefaciens has a central role in crown gall formation in many plants. Actually the T-DNA part of the Ti plasmid is responsible for the tumorous phenotype. Although only one or more copies of the T-DNA could be integrated into the plants genome, but in general, the regions of T-DNA absolutely required for its transfer and integration into the plant genome are border regions which are short repeat sequences of 24 bp. Hence the Ti plasmids are excellent vectors used in gene transfer in plants. The genes of Ti plasmid responsible for tumour formation must be removal in order to make Ti plasmids as vectors for plant transformation. Ti plasmids lacking tumorigenic functions are called disarmed vectors.



A Ti plasmid

The Common features of Ti plasmids:-

i) Contain one (or more) T-DNA regions.
ii) Contain an origin of replication.
iii) Contain a vir (virulence) region.
iv) Contain a region enabling conjugative transfer.
v) Contain genes for the catabolism of opines.

Nopaline strains of Ti plasmid contain one T-DNA of about 20 kb, while octopine strains contain two T-DNA regions - T_L (14 kb) and T_R (7 kb). Only T_L region is oncogenic and T_R region contains genes for opine biosynthesis. The T-DNA of nopaline Ti plasmid comprises of 13 open reading frames (ORFs) while the T_L T-DNA of octopine plasmids contains eight ORFs. The T_L octopine T-DNA shows striking homology with the nopaline T-DNA in the core region. This region contains the following genes.

- aux genes [oncogenes] = encode enzymes involved in auxin biosynthesis.
- cyt (or tmr or ipt) gene [oncogene] = encodes isopentyl transferase enzyme involved in cytokinin production.
- tml gene = involved in regulating tumour size.
- ocs (octopine synthase) gene - encodes opine synthesis.

The vir region :-

The genes responsible for transfer of T-DNA called virulence genes (vir genes) are also situated on the Ti plasmid, in a region of about 40 kb outside the T-DNA. Agrobacterium infection requires wounded plant tissue because vir genes are induced by phenolic compounds and sugars released by the injured plant cells. There are at least 11 vir gene operon having their specific functions.

The Process of T-DNA transfer and integration:-

The mechanism of Agrobacterium T-DNA transfer and integration into the plant genome can be divided into the following steps.

step1 : Signal recognition by Agrobacterium:-

The wounded or injured plant cells release phenolics and sugars. Those substances act as signals perceived by the Agrobacterium and indicate the presence of plant cells which are competent for transformation.

step:2 → Attachment of plant cell :-

It is a two-step process, firstly a Polysaccharide helps in initial attachment of Agrobacterium to plant cell and secondly, the bacterium produces mesh of cellulose fibres. several chv genes are involved in attachment of bacterium to the plant cell. For example the production and secretion of cyclic β-1,2-glycans are regulated by chvA and chvB genes, while release of succinoglycan is controlled by PscA gene.

step:3 → Induction of vir genes :-

The virA is a membrane-linked sensor kinase which senses phenolic released by wounded plant cells. The virA activates virG by phosphorylation and virG induces the expression of all the genes in the vir region on the Ti plasmid. The virG induction is also enhanced by another chromosomal vir gene-called ChvE through the release of sugars.

step-4 :- T-Strand production :-

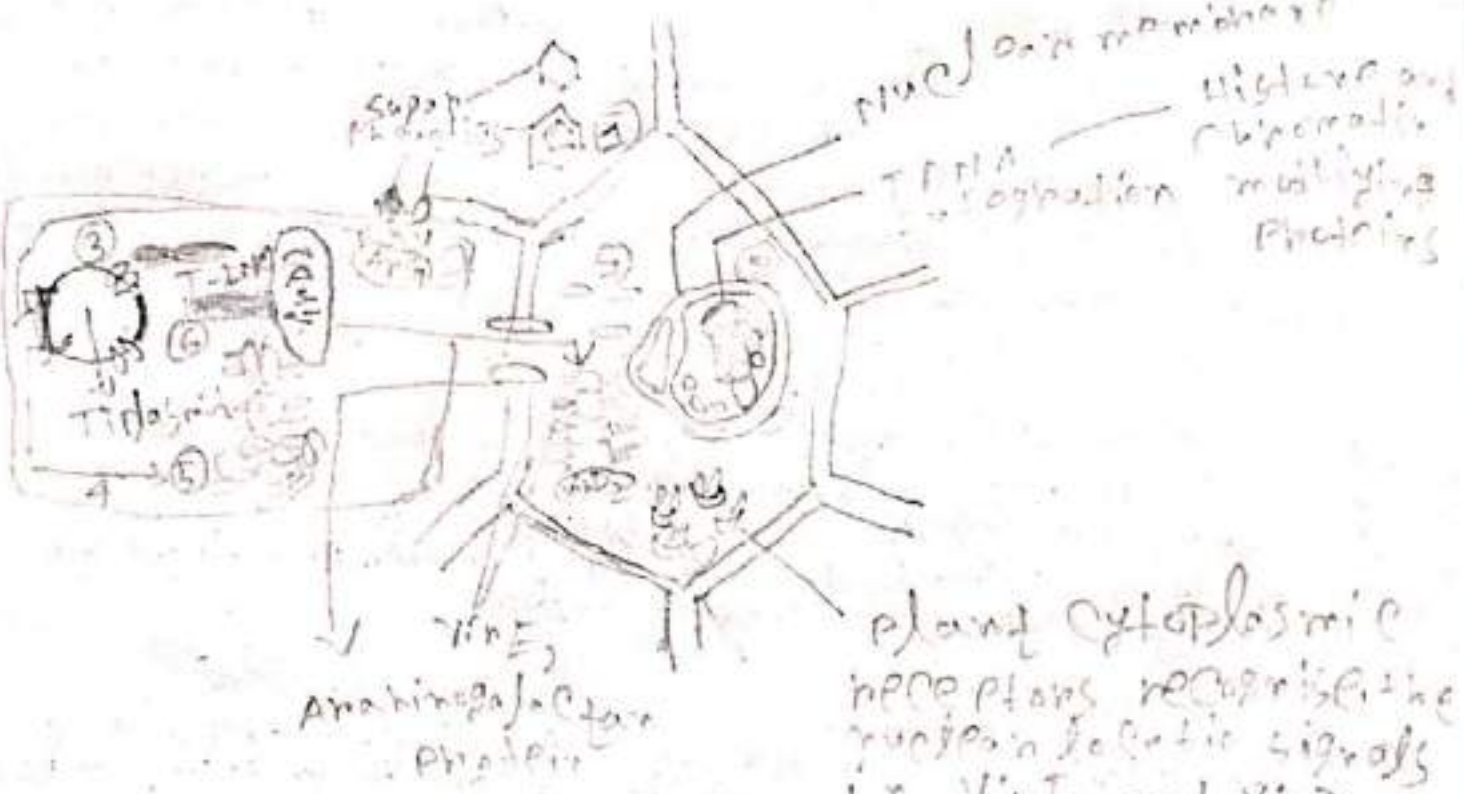
The left and right borders are sensed by a virD1-D2 complex. The virD2 produces single stranded (ss) T-DNA and a T-DNA - virD2 complex is formed. virC1 is also involved in this process.

step5 :- Transfer of T-DNA out of the bacterial cell :-

A membrane-channel secretory system called T-pilus is formed with several proteins encoded by the virB operon and virD2. The T-DNA-virD2 complex and virE2 and also virF are exported from the bacterial cell through the transfer apparatus.

step-6 :- Transfer of the T-DNA and vir Proteins into the plant cell and nuclear localization

Inside plant cell the T-DNA becomes coated with virE2 Proteins and is called a mature T-Complex. The virE2 molecules actually protect T-DNA from nuclease activity and facilitate nuclear localization. virD2 interacts with a group plant proteins. called the karyopherin family. Various plant Protein interact with virD2 interacts with a group of plant proteins interact with virD2 and virE2, which are attached to the T-DNA and influenced transport and integration. The T-DNA-virD2-virF2-plant protein complex enters the nucleus through the nuclear pore complex. The T-DNA strand of Agrobacterium is integrated into the host plant genome by illegitimate recombination. Histone and chromatin modifying proteins are also involved in T-DNA integration.



T-DNA transfer and integration process

1. Wounded plant cell releases signaling molecules like Phenolic Compound and sugars.
2. ~~Sugars~~ signaling molecules are recognised by the dimeric transmembrane receptor complex virA-chv.
3. VirG binds with vir operon and regulates expression of other virgenes.
4. Virgenes are involved in variety of process.
5-8. ssT-DNA production, protection and export.
9. T-DNA integration.

AGROBACTERIUM Based Generation of Transgenic Plants:-

In Agrobacterium-mediated gene transfer, the following criteria are to be taken into consideration:

- The host plants must produce acetosyringone to be required for induction of vir genes. Otherwise, Agrobacterium should be pretreated with synthetic acetosyringone.
- Agrobacterium should have access to the competent cells for transformation. Hence, wounded or dedifferentiated cells or fresh explants are used.
- Transformation competent cells must be able to regenerate into whole plants.

⇒ The most Agrobacterium-mediated transformation protocols follow a similar path. These include:

- Identify a suitable explant.
- Co-cultivate with the Agrobacterium.
- Kill the bacterium with a suitable antibiotic that does not harm the host plant tissue.
- Select for transformed plant cells.
- Regenerate whole plant.

Somil Mandal Roll - 20

1. What do you mean by organic evolution?

⇒ organic evolution means the origin and appearence of new living organisms on earth as a result of slow and steady natural process of continues change. It may be also defined as "cumulative change in the characteristics of organisms occuring in successive generations related by descent."

2. Name two books which is written by Darwin and write when was it published?

⇒ ~~Ans~~ "on the origin of Species by Means of Natural Selection" on November 24, 1849 and "The voyage of Beagle" Published in 1837-38.

3. Define 'special creation'?

⇒ According to the Theory of Special Creation each and every life form was created by some supernatural power as a separate and fixed entity. Each life form was unable to change itself or ~~two~~ to give rise to other types of life forms and was therefore not related by ~~descent~~ descent to any other.

5) What is the full name of Darwin? Where did he get the idea about his book?

⇒ The full name of Darwin is Charles Robert Darwin. He got the idea about his book when he visit to the Galapagos Islands in 1835 which lie on the equator off the coast of Ecuador, and he published his theory in his book. "The Origin of Spices in 1859".

26) What are the two major theories about evolution as given by Darwin in his book "O. of Species"?

⇒ The two major theories about evolution as given by Darwin in his book "origin of Species" are:

» The first is Darwin's idea descent with modification It holds that all species living ~~and~~ have descended, without interruption. from one or a few original forms of life. Species that diverge from a common ancestor are at first very similar accumulate difference over great spans of time, so that they may come to differ radically from one another. Darwin's Conception of the Course of evolution is Profoundly different Lamarck's. in which the Concept of Common ancestry played almost no role.

» The second theory in The origin of Species is natural selection, which Darwin Proposed is the chief cause of evolutionary Change. e summarized it in the following way. "If variations useful to any organic being ever occur. assuredly individuals thus Characterized will have the best ~~Chance~~ Chance of being Preserved in the Struggle for life. And from the strong Principle of inheritance. these will tend to produce off spring similarly Characterized. The

Principle of preservation, or the survival of the fittest. I have called natural selection".

7) who gave the transformational theory?

⇒ Lamarck gave the transformational theory.

8) what are the main components of Darwin's theory?

⇒ The main components of Darwin's theory are:

i) Evolution as such is the simple proposition that characteristics of organisms change over time. Darwin was not the first to have this idea. but he so convincingly marshaled the evidence for evolution that most s. ____ soon accepted that it has indeed occured.

ii) Common descent: Differing radically from Lamarck. Darwin was the first to argue that species had diverged from common ancestors and the species could be portrayed as one great family tree respecting actual ancestry.

iii) Gradualism is Darwin's proposition that the differance between even radically different organisms have evolved by small steps through intermediate forms not by leaps.

iv) Population Change is Darwin's hypothesis that evolution occurs by change in the propo of different variant kinds ~~to~~ of individuals within a population. This profoundly important

(iv) Completely original idea contrast ~~what~~ with the sudden origin of new species by Saltation and with Lamarckian transformation of individuals. For Darwin the average was a statistical abstraction. there exist only varied individuals and there are no fixed limit to the variation that a species may undergo.

v) Natural selection was Darwin's brilliant hypothesis, independently conceived by wallace that accounts for adaptations, features that appear "Disinged to fit organisms to there environment.

Q9) Give a detailed account of theory of natural selection.

⇒ Natural selection: According to Charles Darwin natural selection is the chief process by which evolutionary change occurs. It is a naturally occuring, machanistic model in which those organisms that possess traits that make it more likely for them to survive are more likely to pass those trait along to there offspring, there by altering the genetic distribution in there population over multiple generation and resulting in adaptation to the environment.

According to Mayr Darwin's extensive discussion of natural selection can be

distilled to five facts and three associated inference.

Darwin's Concept of natural selection was based on several key observation:

i) Traits are often heritable in living organisms many characteristics are inherited or passed to from parent to offspring (Darwin knew this was the case, even though he didn't know that traits were inherited via genes).

ii) More offspring are produced than can survive. organisms are capable of producing more offspring than there environments can support Thus there is competition of limited resource in each generation.

iii) offspring very in there heritable traits. The offspring in any generation will be slightly different from one another in there traits and many of these features will be heritable.

10) Define altruistic behaviours.

Group selection is a types of natural selection that acts collectively on all members of a given group. Typically the group under selection is a small cohesive social unit and members interactions are of on altruistic behaviour. Example of behaviours behav. that appear to influence group selection include cooperative

hunting. Such as among lions and other social carnivores.

10) What do you mean by group selection? How is it better from individual selection? Give on example of group selection.

⇨ Group Selection is proposed mechanism of evolution in which natural selection acts at the level of the group instead of at the more conventional level of the individual.

The difference between group selection and individual selection is that group selection, result from a differents between the rates of survival or reproduction of groups rather than individuals in each groups of beetles were not immediate family members. they were more genetically related family members. they were more genetically related to each other than they are two the beetles in other groups. in effect selection that favours certain groups is favour in certain families.

Example of group selection :-

a) Hypothetical example is wyne Edward's theory. Put for word in animal. Dispersion in Relation to social Behaviour (1962), that animals restrain there reproduction in order to over eat the local food

supply. If all the individual in a groups reproduce at the maximum rate, there offspring might over eat the food supply and the group would extinct. They could avoid this role. by collectively restraining their reproduction to maintaining the balance of nature.

12) Who proposed the natural theory of molecular evolution?

⇒ The natural theory of molecular evolution holds that most evolutionary changes at the molecular level, and most of the variation within and between species, are due to random genetic drift of mutant alleles that

12) Who proposed the natural theory of molecular evolution?

⇒ The natural theory of molecular evolution was introduced by the Japanese biologist Motoo Kimura in 1968 and independently by two American biologist Jack Lester King and Thomas Hughes Juckes in 1969, and described in detail by Kimura in 1983 monograph "The Natural Theory of Molecular Evolution".

13) What is Neutral Drift?

⇒ According to Kimura most it is incorporation of mutations which have little or no effect on the Protein in its current ~~environment~~ environment.

14) Define Purifying selection.

⇒ In natural selection. Purifying selections is the selective removal of alleles that are deleterious. This can result in stabilising selection trough the puring of deletorious genetic polymorphisms that arise thorough random mutations.

15) write about theory of natural molecular evolution with one example.

⇒ The natural theory of molecular evolution holds that most evolutionary changes at the molecular level and most of the variation within and between species, are due to random genetic drift of mutant alleles that are selectively neutral.

i) The neutral theory allows for the possibility that some mutations are deleterious.

ii) Mutations do not make significant contributions to variation within and between species at the molecular level

iii) A neutral mutation is one that doses not effect an organisms ability to survive and reproduce.

iv) The neutral theory assumes ~~that~~ that ~~does not at first~~ most mutations that are not deleterious are natural rather than beneficial.

v) The neutral theory suggests that a mutant allele can arise within a population and reach fixation (equilibrium) by chance. rather than by selective advantage.

vi) The ~~not~~ neutral theory asserts that alternative alleles are selectively neutral.

The neutral theory assents that ~~alternat~~ alternative alleles found at his locus are selectively neutral. Neutral Population many Perhaps most, Populations of plants and animals are polymorphic at this locus, i.e. they have two or more alleles with different amino acid sequences.

Example: ~~Example~~

Experiments in Colias butterflies and other organisms have shown that different electrophoretic variant of GPI (Glucose phosphate isomerase) have different enzymatic Capabilities and different thermal stabilities.

4) Give a brief account of Lamarck's hypothesis? Give example.

⇒ Lamarck hypothesized that different organisms originated separately by spontaneous generation from non-living matter. Starting at the bottom of the ~~spon~~ Chain of being A "nervous fluid" acts within each species, he said causing it to progress up the chain Lamarck argued that species differ from one another because they have different needs, and so use certain of their

apandages and organs more than others. Just as muscle becomes strengthened by work. Lamarck belived such alternations aquired during individual's lifetime, is inherited a Principle called inherilence of acquired Characteristics. The theory based on this Principle is called Lamarckism.

Example: According to Lamarck giraffes must have streched their necks to reach foliage above them and so their necks were lengthened. The longer necks were inherited, and over the course of generation, this process was repeated and their necks got longer and longer.

■ Agrobacterium with diagram.

Agrobacterium

The Agrobacterium mediated gene transfer method is the best one in indirect gene transfer category.

The transformation of a plant can be carried out directly by using Agrobacterium Sp. which is a common bacterium causing crown gall tumour in legumes. This bacterium carries a plasmid with T-DNA which is capable of being integrated into the host chromosome. If a foreigen gene is introduced in the plasmid of the bacteria and a plant tissue or cell suspesion is grown in culture along with the bacteria, then ultimately the foreigen gene can be transferred into the nucleus of the plant or more precisely, in the functional Position of the host genome.

Basic Steps for Agrobacterium mediated transformation:

■ Selection of explant: Suitable Palant tissue, to be used as a source of explants is removed from the donor Plant and Sterilized. The explants may be decapitated seedlings, cells, Protoplast or leaf tissue callus etc.

■ Co-cultivation with Agrobacterium: The ~~incubation of~~ the tissue or explant is cut into small pieces and placed into a culture of Agrobacterium for about 30 min, a process known as co-cultivation. During this Period, the bacteria attach to the Plant tissue, and the excess culture is blotted of and placed on medium for co-cultivation.

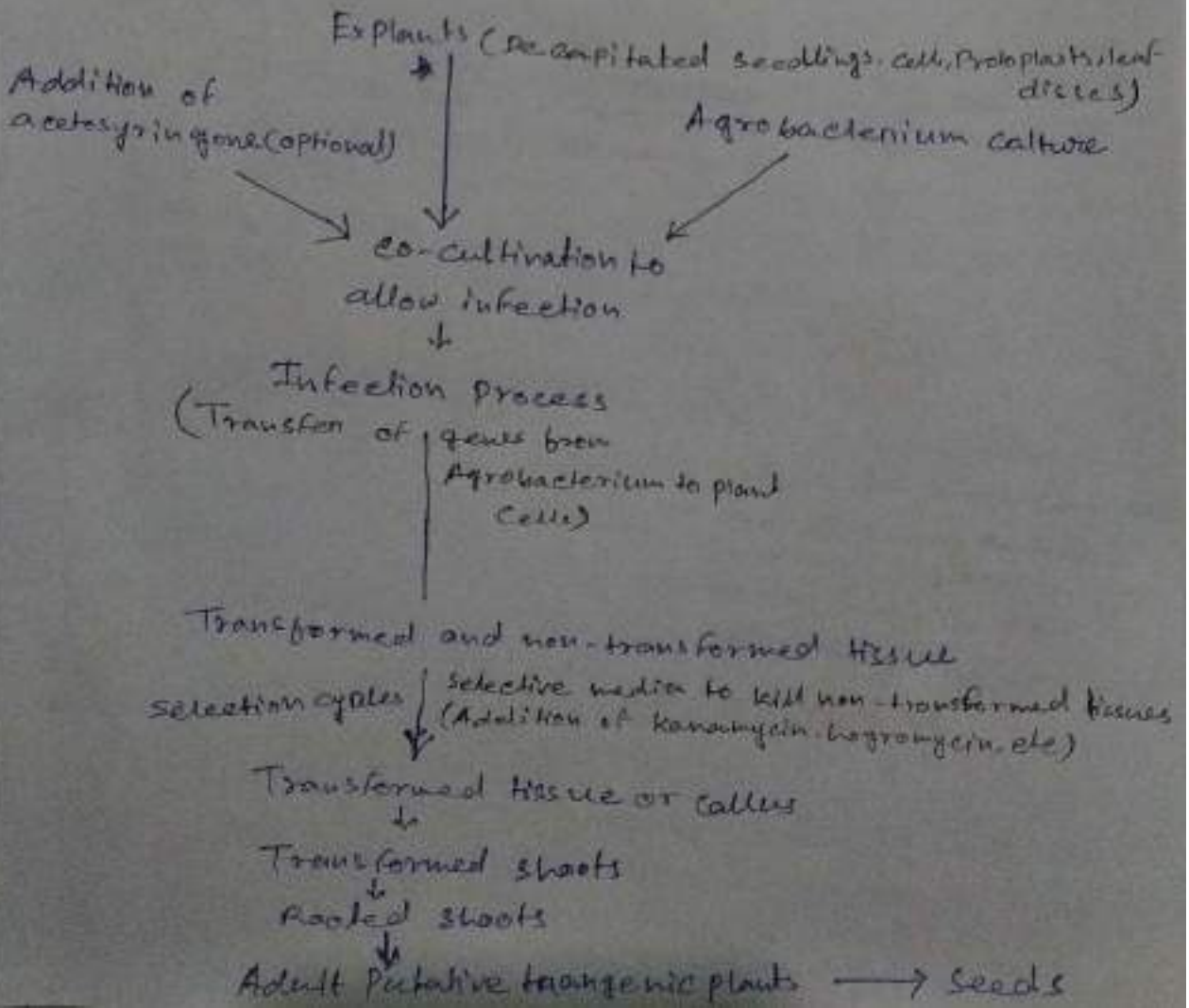
■ Inhibition of Agrobacterium growth: The incubation of the explants with Agrobacterium is allowed to Continue for 2-3 days to permit the transfer of T-DNA to the Plant Cells. Then the explants are removed from the medium and washed in an antibiotic Containing medium to inhibit the growth of Agrobacterium.

It is a fact that the Tiplasmid is a natural vector for genetically engineering plant Cells because it can transfer its T-DNA from bacterium to the plant genome. However, wild type Ti-Plasmid —

are not suitable as general gene vectors because they cause disorganized growth of the recipient plant cells owing to the ~~one~~ oncogenes in the T-DNA. The tumour cells which result from integration of normal T-DNA have proved recalcitrant to attempts to induce regenerations either into normal plantlets, or into normal tissue which can be grafted onto healthy plants.

☑ Selection of transformed plant cells: The explants are then transferred onto the selective media containing the growth of transformed tissue.

☑ Regeneration from the transformed tissue: The selected tissue part grown on selective media, are then transferred onto the regenation media, are transferred onto the regeneration media for shoot regeneration either by organogenesis or by ~~em~~ embryogenesis in presence of proper selective ~~agent~~ agent. The shoot apices come out; these are then transferred in rooting media to get the whole plant.



Confirmation of the putatively transformed plant: The transgene expression is examined either through foreigen protein expression or any phenotypic character expression. The presence of foreign DNA can be examind either through PCR or Dot Blot or southern Blot experiment. The confirmed transgenic plants then to Soil to get the next generation plant.

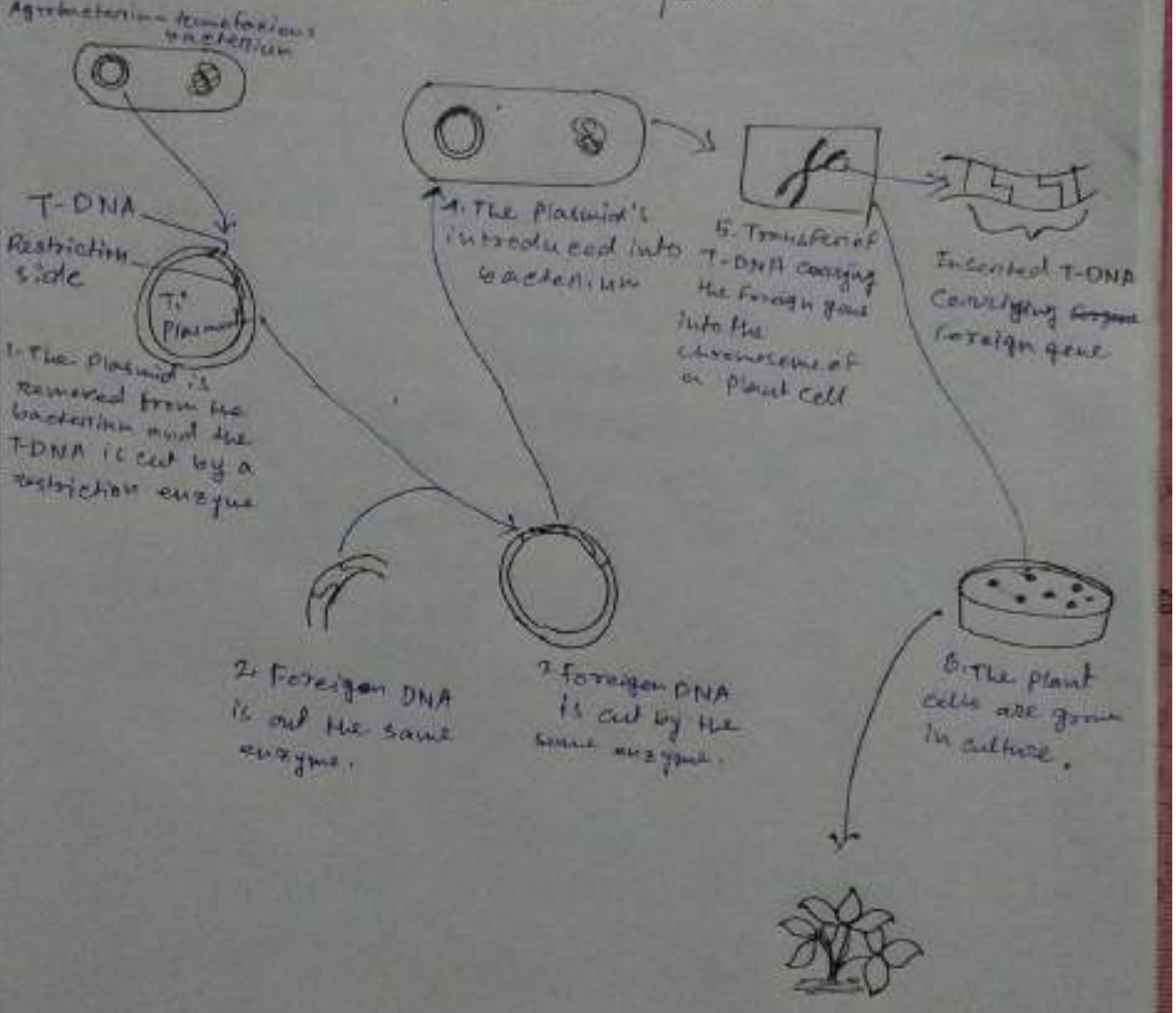


Fig: The plasmid can be cused to transfer foreigen genes into plants.

- **Agrobacterium Mediated Gene Transfer:-**

Agrobacterium mediated gene transfer method is a indirect gene transfer method. The transformation of a plant can be carried out directly by using Agrobacterium sp. which is a common bacterium causing crown gall tumor in legumes. This bacterium carries a plasmid with T-DNA which is capable of being integrated into host chromosome. If a foreign gene is introduced in the plasmid of the bacteria and a plant tissue or cell suspension is grown in culture along with the bacteria, then ultimately the foreign gene can be transferred into the nucleus of a plant.

The Ti plasmid of Agrobacterium sp. has the several features for use as vector.

(i) They contain one or more T-DNA region.

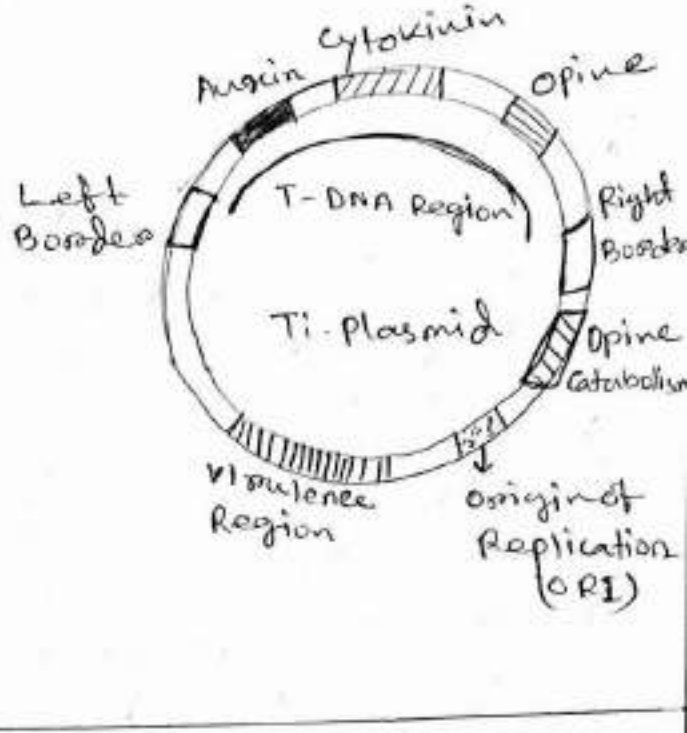
(ii) The plasmid contain vir region.

(iii) The plasmid contains has its own origin of replication.

(iv) They contain a con region enabling conjugative transfer.

(v) They have the genes for catabolism of opines. (a class of amino acids / sugars conjugates)

- The mechanism of T-DNA transfer method →



(a) Signal recognition by Agrobacterium → Agrobacterium perceives signals such as phenolics (acetosyringone) and sugars released from wounded plant cells which are involved in phytoalexin and lignin biosynthesis, these signals indicate the presence of competent cells.

(b) Attachment to plant cell → Attachment of Agrobacterium to plant cells is two step process, involving an initial attachment via polysaccharide which ~~for~~ forms a mesh of cellulose fibres. Several chromosomal virulence (chrV) genes are also involved in the attachment of bacterial cell to plant cells.

(c) Vir gen induction → Vir A (a membrane linked sensor kinase) sense phenolics like acetosyringone, ~~autophy~~ autophosphorylates and activates Vir G. Then Vir G induces expression of all other vir genes. Many sugars like glucose, galactose, xylose — all enhance the vir gene induction; it requires also participation of chromosomal vir gene.

(d) T-strand production → The left and right borders are recognized by Vir D_1/D_2 complex and

Vir D_2 produces single stranded nicks on the DNA, covalently attaches these and replaces the strand by repair synthesis.

(e) Transfer of T-DNA out of the bacterial cell →

The T-DNA / Vir D_2 complex is exported from the bacterial cell by a T-pilus composed of proteins encoded by Vir B operon and Vir D_4.

Explants
(Decapitated seedlings, cells, protoplasts)

↓ Addition of acetosyringone (optional)

Agrobacterium culture ↓

↓ co-cultivation to allow infection

↓

Infection Process (transfer of genes from agrobacterium to Plant cells)

↓

Transformed and transformed tissue

selection cycles ↓ selective media to kill non transformed tissue

Transformed tissue callus

↓

Transformed shoots

↓

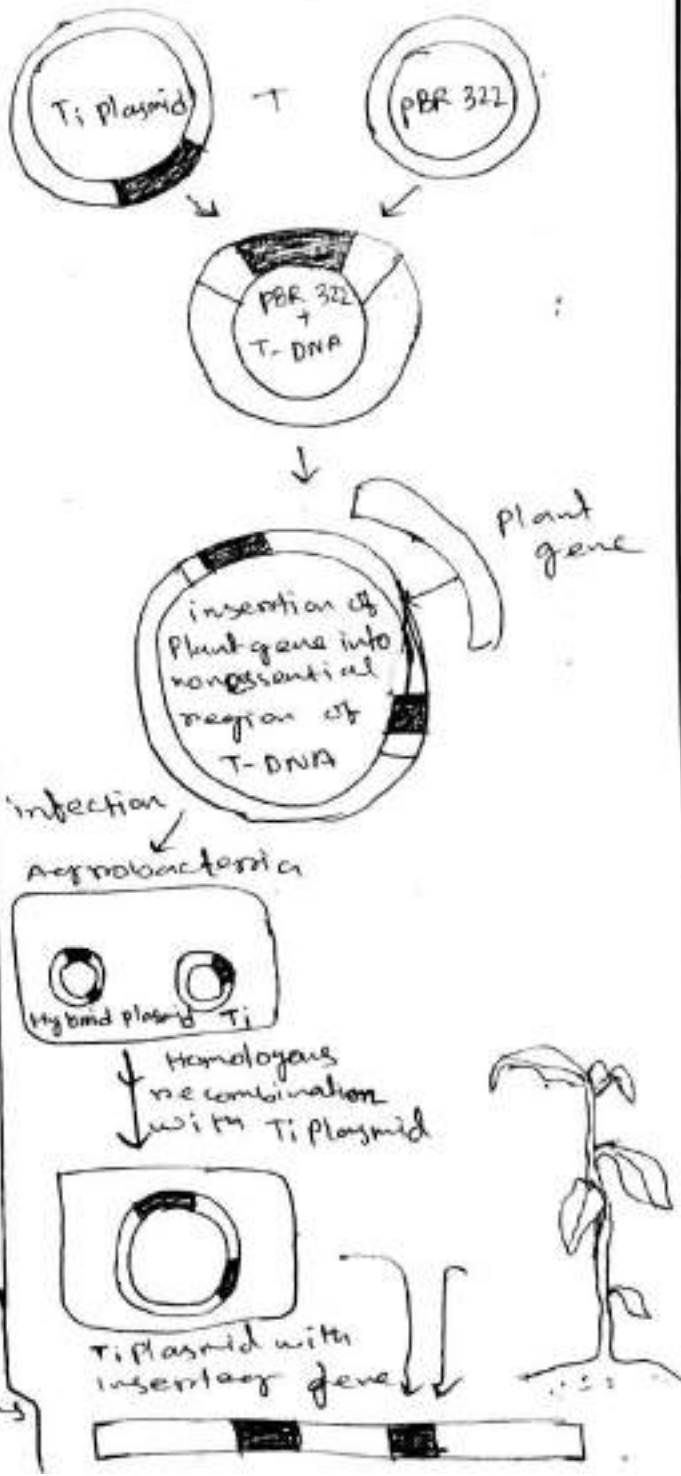
Rooted shoots

↓

Adult putative transgenic Plant

↓

seeds.



Name - Saurik Karmakar.

Reg No - 057-1111-0646-20.

Stream - BSC Bio General.

Subject - Botany (Taxonomy) Assignment.

Semester - 2.

Assignment (Taxonomy)

Family - Compositae

⇒ Characters:

- The stem is erect, branched and cylindrical.
- It may be herbaceous or woody.
- Simple leaves, entire and spathulate.
- The leaves are cauline or normal, alternate or exitipulate.
- They have tap root system and the roots are branched.

→ Floral Formula:-

$$\oplus \; \bar{\varphi} \; K \; \text{Pappus} \; C_{(5)} \; A_{(5)} \; \underline{G}_{(2)}$$

→ Examples:

Helianthus annus

Tagets patula

Chrysanthemum Coronarium

Zinia Eligans

Lactuca Sativa.

P.T.O

Family:- Cucurbitaceae.

→ Characters:-

- Mostly annual or perennial herbs. They are mostly tendril climbers.
- Roots are tap roots.
- Herbaceous, Stems are very soily. It has tendrils.
- Petiolate, Leaves are alternate, simple, exsitipulate, palmately reticulate venation. Some have tendrils in their axiles.
- The flowers are solitary but some has racemose or cymose characteristic clusters.

→ Floral Formula:

Male Flower:- $\oplus \; \bigcirc \; \overrightarrow{\sigma} \; K_{(5)} \; C_5$ or $_{(5)} A_{(3-5)} \; G_{(0)}$

Female Flower:- $\oplus \; \bigcirc \; \overrightarrow{\varphi} \; K_{(5)} \; C_5$ or $_5 A_0 \; G_{(3)}$

→ Examples:-

Cucurbita Pepo

Cucumis melo

Cucumis Sans

Momordica Charantia.

Luffli Acuptica.

Family :- Rubiaceae

→ **Characters** :-

- Plants are usually herbs, sometimes shrubs.
- Leaves are alternate, simple or variously lobed.
- Florets of the capitulum sessile, bisexual or unisexual or the outer female or neuter, bracted or non-bracted.
- Sepals absent or modified into pappus.

Petals are tubular, ligulate, valvate.

→ **Floral Formula** :-

Br ⊡ ⚥ ⚲ $K_{(4-5)}$ $C_{(4-5)}$ $A_{(4-5)}$ $G_{(2-3)}$

→ **Examples** :-

Catunaregam spinosa, Crossaides turgida

Gardenia resinifera, Haldina cordifolia.

Hamelia patens

P.T.O

Family - Malvaciae

→ Characters:-

- Hairs on the body and mucilage in the tissue.
- Alternate phyllotaxy and stipute leaves.
- Multicostate reticulate venation of the leaves.
- Presence of epicalyx.
- Spinous pollen.

→ Floral Formula:-

$\oplus\ \text{⚥}\ Epi_{3-8}\ K_{(5)}\ \widehat{C_{(5)}\ A_{(\infty)}}\ G_{(2-\infty)}$

→ Examples.

Ochroma Pyramidale.

Adamasania Digitala.

Bombax Ceiba.

Brachychiton Rupestris.

Theobroma Cacao.

Family- Leguminaceae.

→ Characters:-

- Leaves of this plant family are placed alternately upto the stem and are pinnate or bipinnate.
- Many members of this family have flowers of the typical 'pea' type.
- It is the seed pods that gives this family its root name.

→ Floral Formula.

$$K_{(5)}\ C_{0}\ \overset{5 \text{ or } 1+2+(2)}{S}\ \overset{9+1,\ 10,\ 10-\infty}{P}.$$

Sexuality- Bisexual.

→ Examples:-

Leguminaceae Astragalus, Leguminaceae Cytisus

Leguminaceae Lotus; Leguminaceae Lupinus

Leguminaceae Trifolium.

Plant tissue culture-Nano silver for removal of bacterial contaminated in valerian(*valeriana officinalis L.)*:A review

Hriday Home Chowdhury, Diya Bhattacharya, Manisha Bhagat

ABSTRACT

Plant tissue cultures are a useful tool for studying cell wall production in live cells.Tissue cultures also provide cells and culture medium where enzymes and cell wall Polymers can easily be sepa-rated for further studies.Tissue cultures with tracheary element differentiation or extracellular lignin formation have provided useful information related to several aspects of xylem and lignin formation.Nanotechnology, at present, is one of the important areas for investigation in sci-ences of modern materials based on nanoparticle (NP) properties that are specific such as size, shape, and distribution.Applications with the use of nanomaterials and NPs are developing fast .In plant tissue culture, there are several research based on nanotechnology,specifically the use of NPs in seed germination,plant growth improvement, plant genetic modification, plant protection, and some others.In this review, methods for nutrient medium preparation, callus culture initiation, and its maintenance, as well as those for protoplast isolation and viability observation, are described. As a case study we describe the establishment of a nanotechnology in plant tissue culture. Bacterial contamination is a serious problem in plant tissue culture procedures .An experiment was conducted to evaluate the potential of nano silver (NS) to silver (NS) to remove bacterial contaminants of valerian nodal explants.This experiment was conducted as a completely random-ized design in a factorial arrangement with four replications and each replicate with ten explants.As this is the first report on application of NS in in vitro culture techniques, further investigations on other plant species are needed to clarify the effectiveness of NS for the removal of bacterial contaminants in tissue culture of other crop.

Keywords: callus culture, initiation, maintenance, protoplast, explant, nanobiotecnology

Introduction

Plant tissue culture is a technique for cultivating plant cells, tissues, and organs on synthetic medium in an aseptic environment with strict light, temperature, and humidity controls.[1]Tissue cultures also provide cells and culture medium where enzymes and cell wall polymers can easily be separated for further studies.The development of plant tissue culture as a fundamental science was closely linked with the discovery and characterization of plant hormones, and has facilitated our understanding of plant growth and development.[2]Furthermore, the capacity to grow plant cells and tissues in culture and control their growth is essential for plant genetic engineering and many other practical applications in agriculture, horticulture, and industrial chemistry.We are living in the 'Nano Age', the era when every aspect of life has a touch of nano, be it the cosmetics we use, the textiles we wear, the appliances we use, the gadgets we employ, the food we eat, or the environment we live in; whether we like it or not, nanomaterials are already in us, on us and around us.In plant tissue culture, there are numerous reports that indicate positive inputs from nanotechnology. [3]Nanoparticles (NPs) have been widely used to improve seed germination, enhance plant growth and yield, enable plant genetic modification, improve bioactive compound production and achieve plant protection.[4]The need to incorporate more of the new-age nanomaterials, such as graphene and carbon buckyballs, and the possibility of creating nano-environments for effective plant tissue culture are speculated upon in the future prospects section.[5]

2.Background of the plant tissue culture

Plant tissue culture dates back to at least 1902, when German botanist Gottlieb Haberlandt (Haberlandt, 1902) hypothesised that single plant cells might be cultivated in vitro.In 1962, Toshio Murashige and Skoog [6] published the composition plant tissue culture medium known as MS (named for the first letters of their last names) medium, which now is the most widely used medium for tissue culture.Commercial tissue culture was born in India in 1987 when A.V. Thomasand Company Kerala (AVT) established their first production unit in Cochin for clonal propagation of superior genotypes of selected cardamom plants.In 1988, a second companyIndo-American Hybrid Seeds at Bangalore, Karnataka, who were in the nursery business in hybrid flowers and vegetables, imported a tissue culture laboratory and greenhouses with a capacity of 10 million plants/ annum)

3.Materials

3.1. Nutrient Medium: A nutritional medium is a supply of nutrients that a plant would ordinarily get from its surroundings receives from the soil There is also a carbon source in the medium typically 1–4% w/v sucrose) and growth regulators that the plant requires in vitro cell division and growth It's possible to add a gelling agent.Solidify the medium(Pierik et al, 1997)

Because different species (even genotypes) have distinct nutritional requirements for optimal growth, a variety of nutrient media have been designed to meet those needs.

Plants that have been cultured in vitro. In order to choose a medium for the presentation,

It is beneficial to conduct a literature search on the topics of interest. Table 1 shows the results.We demonstrate some commonly used media that can be utilised as a starting point.Explants with effective callus initiation are also listed, because the plant's developmental stage has a significant impact on the success of culture initiation.

The nutritional salt composition of the various media is shown in Table 2. Commercially available macro- and microelement mixes can be purchased, or stock solutions can be made from nutrient salts.Microelements, vitamins, and growth regulators can be manufactured as 100–1,000 stock solutions , whereas macroelements can be synthesised as a 10 times concentrated (10) stok Solution.

After combining all medium components except The gelling agent, adjust the pH, adjust for the final volume, add the gelling agent (e.g. agar), and autoclave the medium at 121°C for 20 minutes.Allow the medium to cool to around 60°C. Filter-sterilize the heat-labile compounds in a laminar air-flow cabinet and pour the medium onto Petri dishes (ca. 25 mL medium/Petri dish with a diameter of 9 cm).

Table 1
Types of explants, some widely used nutrient media, and growth regulator concentrations used for successful callus culture initiation

Plant group	Explant	Medium	Growth regulators
Monocotyledonous plants	Immature and mature embryos Leaf, or root segments of aseptically germinated seedlings	MS (14), N6 (24)	Auxin (2,4-D) 1.0–18 µM
Dicotyledonous plants	Young leaf, roots Stem segments Leaf, root, and stem segments of aseptically germinated seedlings	MS, WPM (25)	Cytokinin (BA, 2iP, kinetin, zeatin) 0.1–40 µM + auxin (NAA, 2,4-D) 0.5–10 µM
Gymnosperms	Cambial/xylem strips Shoot tips Zygotic embryos	MS Mod. Brown and Lawrence (3) Mod. N6 (12)	Cytokinin (BA, kinetin) 2–5 µM + auxin (2, 4-D, NAA) 9–16 µM 2.4-D alone: 11 µM (3)

3 2. Surface

Sterilisation: 1. 70% (v/v) ethanol.

2. Diluted Na-hypochlorite: NaClO, 1–2% (v/v) active chlorine,

Supplemented with a couple of drops of Tween 20.

3. Sterile distilled water.

4. 96% (v/v) ethanol for flaming.

5. Forceps.

6. Scalpels.

7. Sterile Petri dishes.

8. Parafilm®.

3.3 Maintenance: 1. Fresh nutrient medium with a gelling agent or, alternatively, without the gelling agent.

Table 2
Nutrient media constituents for plant tissue culture basal media (26)

	B5 (15)		Mod. Brown and Lawrence (3)		MS (14)		N6 (24)		WPM (25)	
Macronutrients	mg/L	mM	mg/L	mM	mg/L	mM	mg/L	mM	mg/L	mM
NH_4NO_3			1,650	20.6	1,650	20.6			400	5
$(NH_4)_2SO_4$	134	1.0					463	3.5		
$Ca(NO_3)_2 \cdot 4H_2O$									556	2.4
KNO_3	2,528	25	1,900	18.8	1,900	18.8	2,830	28		
$MgSO_4 \cdot 7H_2O$	246	1.0	1,900	7.7	370	1.5	185	0.75	370	1.5
KH_2PO_4			340	2.5	170	1.25	400	2.94	170	1.25
$NaH_2PO_4 \cdot H_2O$	150	1.1								
$CaCl_2 \cdot 2H_2O$	150	1.0	22	0.15	440	3.0	166	1.1	96	0.65
K_2SO_4									990	5.7
Micronutrients		µM		µM		µM		µM		µM
H_3BO_3	3.0	49	30.9	500	6.2	100	1.6	26	6.2	100
KI	0.75	4.5	4.15	25	0.83	5.0	0.8	4.8		
$MnSO_4 \cdot 4H_2O$	13.2	59.2	31.2	140	22.3	100	4.4	19.7	22.3	100
$ZnSO_4 \cdot 7H_2O$	2.0	7.0	43.1	150	8.6	30	1.5	5.2	8.6	30
$CuSO_4 \cdot 5H_2O$	0.025	0.1	1.0	4.0[a]	0.025	0.1			0.25	1.0
$Na_2MoO_4 \cdot 2H_2O$	0.25	1.0	1.2	5.0	0.25	1.0			0.25	1.0
$CoCl_2 \cdot 6H_2O$	0.025	0.1	0.13	0.55	0.025	0.1				
$FeSO_4 \cdot 7H_2O$	27.8	100	27.8	100	27.8	100	27.8	100	27.8	100
$Na_2EDTA \cdot 2H_2O$	37.2	100	37.2	100	37.2	100	37.2	100	37.2	100
Organic constituents										
Myo-inositol	100	560	20	111	100	560	100	560	100	560
Nicotinic acid	1.0	8.1	0.5	4.1	0.5	4.1	0.5	4.1	0.5	4.1
Pyridoxine-HCl	1.0	4.9	0.1	0.49	0.5	2.4	0.5	2.4	0.5	2.4
Thiamine-HCl	10	30	0.1	0.3	0.1	0.3	1.0	3.0	1.0	3.0
Glycine					2.0	26.6	2.0	26.6	2.0	26.6
	g/L	mM	g/L	mM	g/L	mM	g/L	mM	g/L	mM
Sucrose	20	58.4	30	87.6	30	87.6	20	58.4	20	58.4
pH	5.7		5.5		5.7		5.7		5.7	

[a]L.B., Davin, personal communication

2. 96% (v/v) ethanol for flaming.

3. Forceps.

4. Parafilm®.

5. Sterile 5 mL pipette tips with cut tips.

6. Sterile measuring cylinders (e.g. 25 mL in volume).

7. Orbital shaker (in the case of liquid cultures).

8. Temperature- and light-adjusted growth chamber.

3.4. Protoplasts: 1. Preplasmolysis solution, enzyme solution, and nutrient medium for protoplast cultivation according to (Table 3).

2. Syringes.

3. Syringe filters (0.2 mm pore size).

4. Forceps.

5. Scalpels.

6. 96% (v/v) ethanol for flaming.

7. Sterile nylon or steel sieves (70–100 mm pore size), screw-cap centrifuge tubes.

8. 20% (w/v) sucrose solution (autoclaved).

9. Fuchs-Rosenthal modified haemocytometer.

10. Microscopic slides.

11. Cover glasses.

12. Agars with low melting point (m.p.)specifically designed for protoplast culturing (e.g. A8678 Agar washed, m.p. 25–27°C; A7921 Agar purified, m.p. 30–35°C, Sigma).

13. Sterile pipette tips.

14. Petri dishes.

15. Parafilm®.

Viability stains:

16. 5–10 mg/mL fluorescein diacetate (FDA) in acetone (stock solution). This is then diluted immediately prior to use by adding 20 mL of the stock solution to 1 mL of 0.65 M mannitol.

17. 0.025–0.25% (w/v) Evans blue (EVB) in 0.65 M mannitol.

18. 0.025–0.25% (w/v) Methanol blue in 0.65 M mannitol.

19. 0.1% (w/v) Phenosafranine in 0.65 M mannitol.

20. 0.01–0.1% (w/v) Tinopal CBS-X (disodium 4,4′-bis[2-sulfostyryl)biphenyl) in 0.65 M mannitol.

Table 3
Solutions for protoplast preparation and cultivation

Preplasmolysis solution	
B5/MS macroelements	
B5/MS Microelements	
sucrose	60 mM
Mannitol/sorbitol	0.3–0.5 M
pH 5.7 (see Note 33)	
Sterilise in autoclave.	
Enzyme solution (make fresh each time)	
0.5% (w/v) Cellulase and 0.2% (w/v) Macerase or 0.1–4% (w/v) Cellulase, 0.05–2% (w/v) Pectolyase/Macerase and 0.1–2% (w/v) Hemicellulase in the preplasmolysis solution	
Mix gently for 15–30 min to dissolve, filter-sterilise through syringe filters (0.2 µm pore size)	
Nutrient medium for protoplast cultivation	
B5/MS macroelements	
B5/MS microelements	
NaFe-EDTA	100 µM
B5/MS vitamins	
Sucrose	60 mM
Mannitol/sorbitol	0.3–0.5 M
Myo-inositol	100 mg/L
Plant growth regulators	
Auxin (2,4-D/NAA/IAA)	1–10 µM
Cytokinin (BA/2iP/kinetin/zeatin)	0.5–2.5 µM
pH 5.7 (see Note 33)	
Sterilise in autoclave	

See Table 2 for B5/MS medium constituents

3.5. Zinnia Cultures

2.5.1. Germination

Of Zinnia Seeds: 1. 0.25% Na-hypochlorite.

2. Mesh strainer.

3. Vermiculite.

4. Plastic trays.

5. Growth chamber.

6. Liquid fertiliser: e.g. HYPONeX;N:P:K=6:10:5 (HYPONeX

Japan, Osaka). Dilute 1:100 before use.

3.5.2. Isolation and Culture

Of Mesophyll Cells: 1. Table 4 shows the composition of the nutrient medium (see

Note 8). Frequency of TE differentiation is optimal when

The nutrient medium is supplemented with 0.89 mM

6-benzyladenine (BA) and 0.54 mM 1-naphtalene acetic acid

(NAA). Medium without BA and/or NAA can be used for

Control cultures in which TE differentiation does not occur.

2. 0.1% Na-hypochlorite with 0.001% (w/v) Triton X-100.

3. Sterile distilled water.

4. Sterile labware: Waring-type blender, stainless-steel cups,

Nylon mesh (50–80 mm pore size), screw-cap centrifuge tubes,

Pipette tips, culture tubes (30 mm internal diameter (i.d.)

×200 mm, 18 mm i.d.×180 mm or 12 mm i.d.×105 mm)

Capped with aluminium foil.

5. Revolving drum.

6. Growth chamber.

3.5.3. Observations

Of Zinnia Cells: 1. Glutaraldehyde.

2. 0.2 mg/mL 4′,6-diamidino-2-phenylindole (DAPI), 1 mM SYTO16 in DMSO (Molecular Probes).

3. Microscopic slides.

4. Cover glasses.

5. Haemocytometer. [7]

4.Methods:-

4.1. Surface

Sterilisation: The goal of surface sterilisation is to eliminate microorganisms from plant materials while leaving the plant tissue alone.

It is critical to choose a healthy plant tissue as an explant for culture initiation. Wash the plant organ with tap water if necessary, then cut it.Cut it into 1 cm pieces. The seeds have not been surface sterilised. Execution Procedures are carried out aseptically in a laminar air-flow system.

1. Pretreat the explants for 30–60 sec in 70% (v/v) ethanol.

2.Transfer the pieces to a diluted Na-hypochlorite solution containing 1–2 percent (v/v) Na-hypochlorite and a few drops of Tween 20. Incubate for 5–30 minutes, stirring occasionally.

3. Rinse the explants three times with sterile distilled water (at least 1 minute of incubation between rinses to remove all surface sterilants). To transfer the fragments from one solution to the other, use alcohol-flamed forceps[8]

4. Because contact with the surface sterilising agent damages the cut surfaces of the plant material, cut the surfaces fresh using a sterile scalpel using half of a sterile Petri dish as a cutting board.

5. Dissect the tissue of interest (e.g. embryo, cambial strips) from the seed/plant organ aseptically and place it on the initiation medium's surface. If necessary, use a stereomicroscope in the laminar air-flow cabinet. Seal the dish with a strip of Parafilm.[8]

Table 4
Medium for xylogenic culture of *Zinnia* mesophyll cells

Constituents	Concentration (mg/L)	Molarity
Macroelements		mM
KNO_3	2,020	20
$MgSO_4 \cdot 7H_2O$	247	1
$CaCl_2 \cdot 2H_2O$	147	1
KH_2PO_4	68	0.5
NH_4Cl	54	1
Microelements I		µM
$MnSO_4 \cdot 4H_2O$	25	110
H_3BO_3	10	60
$ZnSO_4 7H_2O$	10	35
$Na_2MoO_4 \cdot 2H_2O$	0.25	1
$CuSO_4 \cdot 5H_2O$	0.025	0.1
Microelements II		µM
$Na_2EDTA \cdot 2H_2O$	37	100
$FeSO_4 \cdot 7H_2O$	28	100
Organic growth factors I		µM
Myo-inositol	100	560
Nicotinic acid	5	41
Glycine	2	27
Pyridoxine-HCl	0.5	2.4
Thiamine-HCl	0.5	1.5
Biotin	0.05	0.2
Organic growth factors II		µM
Folic acid	0.5	1.1
Growth regulators		µM
NAA	0.1	0.54
BA	0.2	0.89
	g/L	mM
Sucrose	10	29.2
D-Mannitol	36.4	200
pH	5.5	

4.2. Growth Conditions:

Plant species have different temperature and light requirements.

If there is no information in the literature on the in vitro growth circumstances of the species (or similar species) of interest, you might choose the light and temperature conditions in which the plant grows in vivo.

A constant temperature (e.g., +25°C) is typically used; alternatively, the temperature is dropped at night (+25°C during the day, +20°C at night).The choice of lamp determines the quality of the light.Fluorescent warm white bulbs are preferred by some species, such as Norway spruce.[9]

The intensity and rhythm of light are quite essential. Unless any information is accessible in the literature, you can only approximate these numbers by trial and error. Light intensities of 20–200 mmol/m2/s are commonly employed. However, some in vitro cultures are grown in the dark.[9]

4.3. Maintenance:

Callus development appears at the margins of the explant after 2–6 weeks in culture (Fig. 1). You must subculture callus in order to provide fresh nutrients and growth regulators to the cells. Depending on callus growth, subculture cells at 1- to 4-week intervals. At this point, you may need to make changes to the nutrient medium in terms of nutrient and growth regulator concentrations and types.

1. Subculture callus by transferring the freshest cells (typically at the callus' edges) onto fresh medium with flamed for- ceps (see Note 10). The inoculum should be kept at a consistent size (about 0.90.90.5 cm;)

It is not a good idea to transfer inoculums that are too tiny, since it takes longer for cells to start dividing once they have been subcultured. If you subculture inoculums that are too large, the cells divide too quickly and enter the stationary phase too soon (they also fill the growth container). Subculturing will be done more frequently as a result of this.

2. Callus can also be transferred into a liquid culture (Fig. 1b).Make the nutritional medium without the gelling agent and aliquot it into 25 mL aliquots into 100 mL flasks for this.Using a second layer of aluminium foil, seal the flask and auto-clave it.Inoculate the liquid medium with the most friable callus cells (ca. 0.5 g of cells into a 25 mL medium). Whether you get a fine cell suspension with single cells and small cell aggregates or a callus that grows in huge clumps with no cell dissociation depends on the type of callus.

3. Keep the cultures in an orbital shaker (100 rpm) for aeration in the same growth conditions as solid media cultures.

4. Allow cells to settle to the bottom of the flask and subculture at regular intervals into fresh medium (see above).Take out some of the culturalitems.Use a 5-mL cut, autoclaved pipette tip or a measuring cylinder to transfer about 5 mL of cells into 20 mL of fresh media.[10]

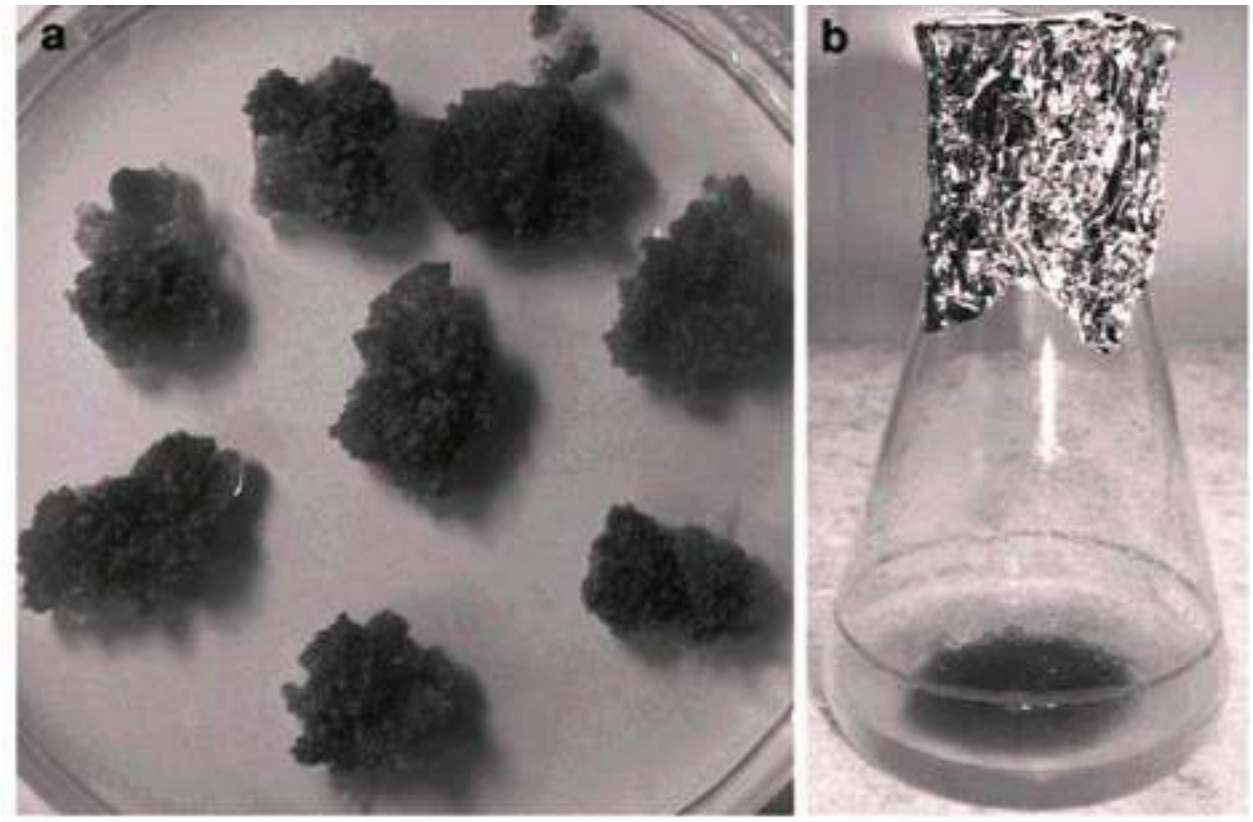


Fig. 1. (**a**) Callus culture of Norway spruce (*Picea abies*). (**b**) Cell suspension culture of Norway spruce composed of single cells and small cell aggregates.

4.4. Protoplasts:

Protoplasts are plant cells that have had their cell walls digested by the enzymes pectinases, hemicellulases, and cellulases, which degrade plant cell walls (Table 5). Enzymatically, protoplasts can be iso-lated in two ways. The cells are first divided into cell suspensions by pectinases, which breakdown the pectinous middle lamella between the cells in a two-step procedure.

Cellulases and hemicellulases are then used to breakdown the residual cell walls.

A combination of pectinases and celtinases is used in the one-step process.At the same time, lysates are produced for cell wall digestion.[10]

Plant components such as root tips and leaves, as well as suspension-cultured and callus cells, can be used to make protoplasts. Because protoplasts lack a cell wall, they are highly susceptible to osmotic stress and must be handled in an isotonic/slightly hypertonic solution. To avoid a rupture Protoplast nutrient medium needs are very similar to those of plant cells in culture.

Extra calcium is added to help stabilise plasma membranes, and the MS and B5 mediums are optimised for diverse applications.Species are frequently required.[11,12]

Protoplasts can be used in plant breeding by protoplast fusion or transformation of related species. Regenerative cells can be stimulated to become plant after their cell walls have developed.

Surveillance.Protoplasts are also a great way to learn about cell walls.across cell membranes for production or transfer.

After a normal 24–36 hour incubation period, protoplasts establish a cell wall and are capable of division. Protoplasts lose their distinctive properties.Once the wall creation is finished, it will take on a spherical shape (Fig. 2).[11,12]

Table 5
Some commercially available cell wall-digesting enzymes utilised in protoplast isolation

Enzyme	Source	Supplier
Pectin-digesting enzymes		
Macerozyme R-10	*Rhizopus* sp.	Yakult Honsha, Japan
Macerase	*Rhizopus* sp.	Calbiochem
Pectinase	*Aspergillus niger*	Sigma
Pectolyase	*Aspergillus japonicus*	Sigma
Hemicellulose-digesting enzymes		
Hemicellulase	*A. niger*	Sigma
Viscozyme	*Aspergillus* sp.	Novozymes Corp.
Cellulose-digesting enzymes		
Onozuka R-10	*Trichoderma viride*	Yakult Honsha, Japan
Cellulysin	*T. viride*	Calbiochem
Driselase	*Basidiomycetes* sp.	Sigma

4..4.1. Protoplast Isolation

Prepare the preplasmolysis and enzyme solutions according to

Table 3. Then continue as described below.

LEAVES:

1. Surface sterilise young, fully expanded leaves as described in

Subheading 3.1.

2. In a Petri dish containing a small volume (10 mL) of the pre-plasmolysis solution, cut the leaf into narrow sections with a sharp scalpel. Peeling of the abaxial epidermis hastens.

Enzymes break down cell walls as they enter the intracellular space, allowing them to more easily reach local regions.

SUSPENSION-CULTURED CELLS:

3. Separate the cells from the culture by centrifuging an actively growing cell suspension culture (10 mL) for 5–10 minutes at 50–100g in the early logarithmic or exponential stage of growth medium.

4. After centrifugation, decant the media and place the cells in a Petri dish with the preplasmolysis solution.

CALLUS CULTURE :

5. Transfer actively growing callus cells (from the edges of callus

Pieces) into a Petri dish containing the preplasmolysis solution.

6. In the preplasmolysis solution, incubate the plant material. Replace the preplasmolysis solution with the enzyme solution after 30 minutes. Incubate in the dark at room temperature for 0.5–20 hours.

7. After incubation, shake the Petri dish gently to see that the

Tissue is digested; if not, incubate for 1–2 more hours.

8. Pipette protoplasts through a nylon or steel sieve (pore size 70–100 mm) into a sterile screw-cap centrifuge tube to remove cell debris.Centrifuge at 50–100g for 5–10 minutes.

9. In the preplasmolysis solution, resuspend the protoplast pellet. Alternatively, separate protoplasts from cell detritus by pipetting the protoplast suspension over a 20% (w/v) ethanol solution.solution of sucrose.

10. Centrifuge at 50–100g for 5–10 minutes.

Protoplasts float at the interface of the sugar layer and the enzyme solution, and cell detritus settles to the tube's bottom. Place the protoplast on the surface.Pour a fresh sucrose solution on top, then wash it for a second time. Three occasions. Protoplasts should be suspended in the nutrient solution.density-appropriate media. [13]

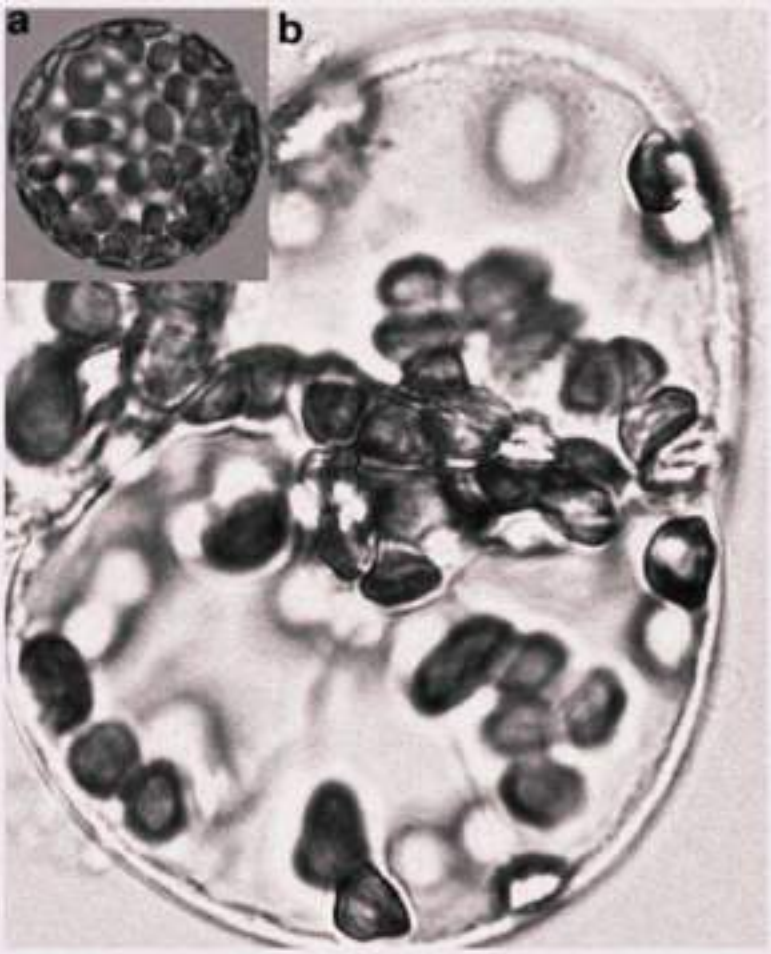


Fig. 2. (**a**) Protoplasts made of *Nicotiana tabacum* leaves. (**b**) After a couple of days in culture, the cell wall has regenerated and the cell has divided. (Photograph courtesy of Enni Väisänen, University of Helsinki).

4.4.2. Protoplast Viability

Tests: The viability of protoplasts can be determined using several dyes that distinguish between viable and non-viable cells. To prevent protoplast rupture, appropriate osmoticum must be added to the staining solution. EVB is an acronym for Electronic Video Broadcasting.Living cells are not dyed blue, and only dead ones are.Methanol blue (MB) enters both living and dead cells, but it is converted to a colourless molecule in living cells.

Phenosafranine (PS) stains dead protoplasts red when it enters them. TheFDA collects fluorescent dyes inside protoplasts. FDA is cleaved to fluorescent fluorescein in vegetative cells by an esterase.Only dead cells are permeable to Tinopal CBS-X.[13,14]

1. Select the dye you will use in your viability staining. Prepare

It as described in Subheading 2.4.

2. On a microscopic slide, mix equal volumes of staining solution

And the protoplast suspension and overlay with a cover glass.

3. Using a light microscope, examine EVB, MB, or PS and count the number of dead protoplasts per all protoplasts in selected fields.

4. Using a fluorescent microscope with excitation and emission wavelengths of 440–490 nm and 510 nm, respectively, observe FDA (FITC, fluorescein isothiocyanate filter combination). The fluorescence of living protoplasts is quite brilliant.

5. For Tinopal CBS-X, use excitation and emission wavelengths of 334–385 nm and 420 nm, respectively. The fluorescence of viable protoplasts is blue.

5.4.3. Culturing

Of Protoplasts:

Protoplasts are commonly cultivated on agar that is semi-solid.in a liquid medium or in a solid medium MS salts or B5 salts [15,16]

Extra osmoticum, sorbitol, and mannitol were added to the medium.

Sucrose or glucoseare commonly used(Table3)

1.Combine molten agar with a double density protoplast suspension at double the concentration recommended in the recipe last culture Make sure the agar isn't too hot since

This will destroy your protoplasts (the agar level should be just above the melt point). Ing point (about 330°C).

3. Pipette quickly as small droplets (100–200 mL) or plate evenly

Onto Petri dish.

4. Cover plates with Parafilm and incubate at room temperature in diffuse light (5–10 mmol/m2 /s).[15]

5.5. Special Case in Zinnia Cultures

Fukuda and Komamine developed an in vitro experimental system in which single Z. elegans mesophyll cells redifferentiate into TEs without the need for cell division (Fig. 3)(Fukuda,et al,

1980).TE isa term that refers to a period of time.Cell wall structures undergo dynamic modifications throughout development, such as localized thickenings and lignification of secondary cell walls are examples of this. Perforation of primary cell walls and partial degradation of primary cell walls.Active cell wall degradation occurs concurrently with secondary cell wall construction in growing TEs in Zinnia xylogenic culture; pectin is one of the most actively degraded components.Thus, using an in vitro xylogenic culture system,mechanisms relating to cell wall structural alterations can be investigated.

5.5.1. Germination

Of Zinnia seeds:

For the in vitro xylogenic culture, the first true leaves of 14-day-old Z. elegans seedlings were used. Mesophyll cells should be harvested from healthy leaves that have been carefully developed under ideal conditions.

1. Surface sterilise seeds of Z. elegans cv. Canary bird or Envy in 0.25% Na-hypochlorite solution for 10 min with occasional shaking.

2. Wash the seeds with running water for 10 min in a mesh strainer.

3. Sow seeds in moistened vermiculite (0.1 g of seeds/100 cm2) in plastic trays.

4. Grow seedlings at 25° C for 14 days with a 14-hour light and 10-hour dark cycle (approximately 100 mmol/m2 /s, white light from fluorescent lamps). The humidity level in the growing chamber should not exceed 45 percent. When the surface of the vermiculite is dry, add water . On the fourth day after sowing, feed a 100-fold diluted liquid fertiliser .[16]

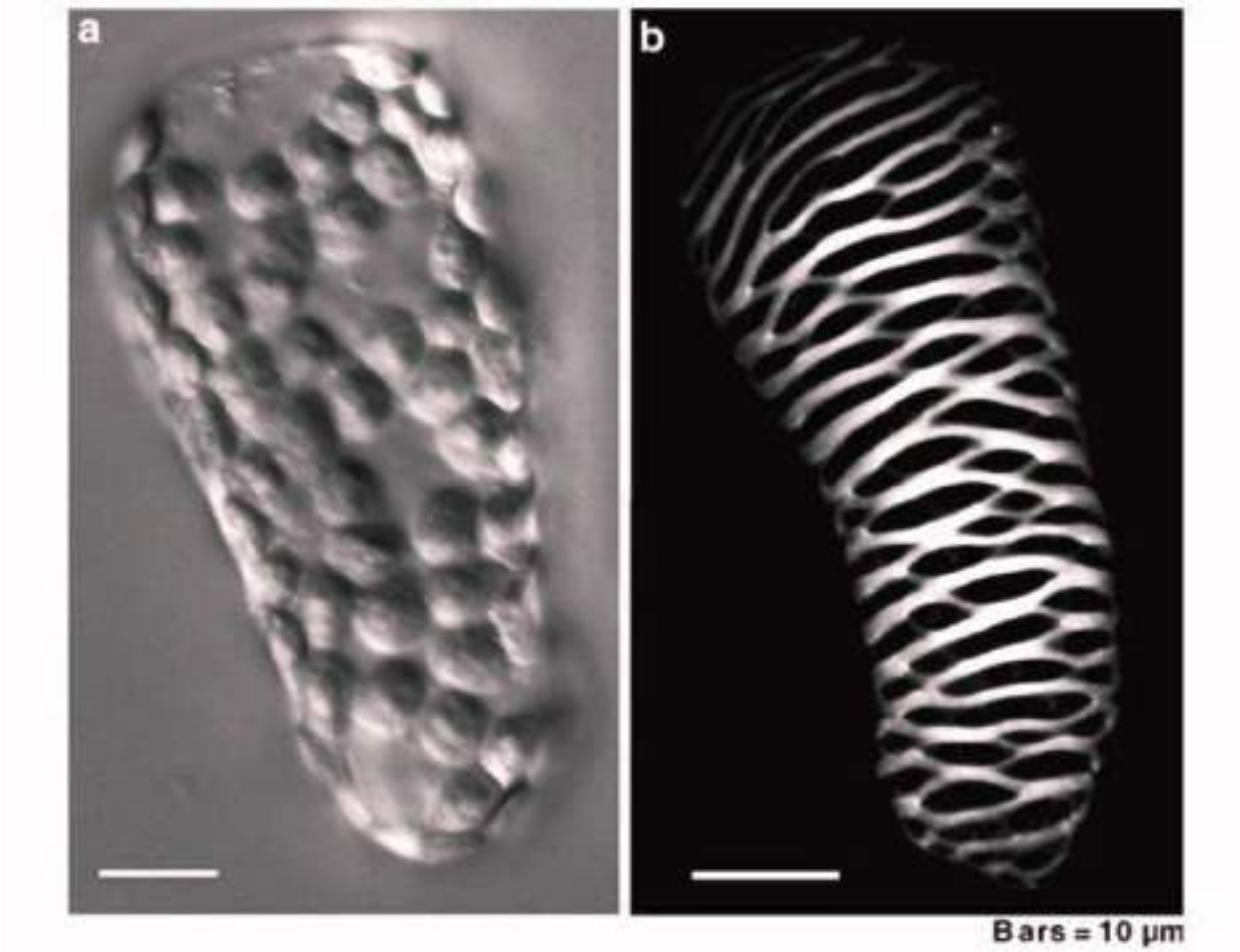


Fig. 3. A mesophyll cell and a TE formed in *in vitro Zinnia* xylogenic culture. (**a**) A single mesophyll cell just after isolation. (**b**) A TE with a thickened secondary cell wall.

5.5.2. Isolation and Culture

Of Mesophyll Cells:

Single Z. elegans mesophyll cells can be isolated by mechanical maceration with a Waring-type blender due to the weak attachment between mesophyll cells. Vascular and epidermalThe cells are removed by passing the cell homogenate through a filter.Because of their high attraction to one another, a nylon mesh is used.The procedure for isolating and cultivating mesophyll cells is discussed below.

1. Harvest first true leaves (80–120 leaves) that are 3–4 cm in length.

2. Surface sterilise leaves for 10 min in 0.1% Na-hypochlorite solution supplemented with 0.001% (w/v) Triton X-100 with occasional stirring.

3. Rinse the leaves with autoclaved water three times.

4. Transfer the leaves into a 100-mL stainless-steel cup containing 60 mL of nutrient medium.

5. Macerate the leaves at 10,000 rpm for 40 s using a Waring-type blender (Fig. 4a, b,).

6. Filter the homogenate through a nylon mesh (Fig. 4c, pore size 50–80 mm) by pipetting using a large-bore pipette. Wash the homogenate that remains on the nylon mesh with 40 mL of additional nutrient medium.

7. Centrifuge the filtrate at 200×g for 1 min.

8. Remove and discard the supernatant with a pipette or by decantation. Suspend the pelleted cells in 80 mL of nutrient medium by gentle shaking.

9. Centrifuge again at 200×g for 1 min.

10. Resuspend the pelleted cells in nutrient medium at a cell density of ca. 8×104 cells/mL.

11. Distribute the cell suspension into culture tubes (20 mL for a tube of 30 mm i.d.×200 mm, 3 mL for a tube of 18 mm i.d.×180 mm, and 1 mL for a tube of 12 mm i.d.×105 mm)

Capped with aluminium foil.

12. Incubate cultures in darkness at 25–27°C on a revolving drum at 10 rpm at an angle of elevation of 8° (Fig. 4d,) [17]

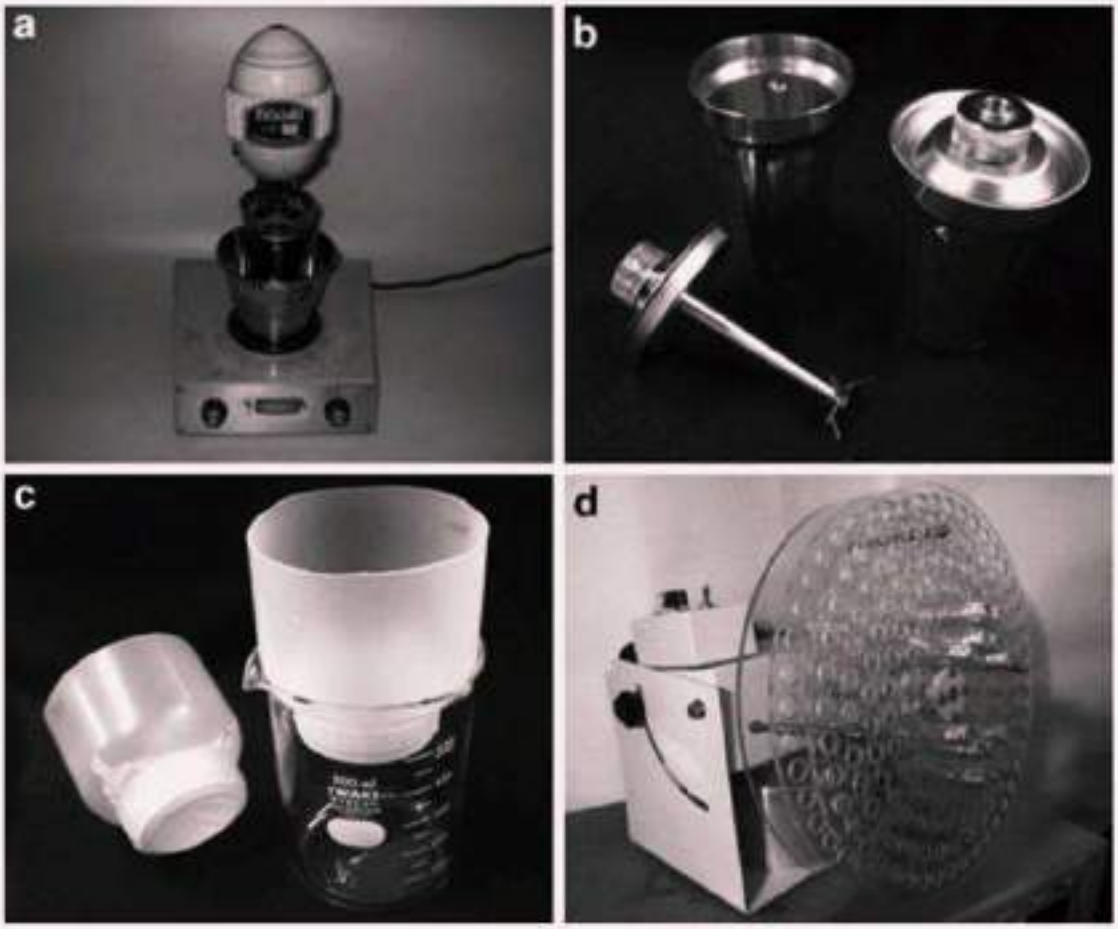


Fig. 4. Experimental apparatus used for the culture of *Zinnia* mesophyll cells. (**a**) A Waring-type blender (**b**) Two sets of the stainless-steel cup and a blade used with the blender (**c**) A nylon mesh is attached to a cylinder and set on a glass beaker for use (**d**) A revolving drum, which is placed in a temperature-controlled incubator or room.

5.5.3. Determination

Of Frequencies of TE

Differentiation and Cell

Division: 30-50% of cells synchronously develop into TEs after 72 hours of incubation. Characteristic patterns of secondary cell walls, visible even under a light microscope, can easily be detected (see Note 32). As a result, the number of TEs generated can be counted without any pre-treatment using a haemocytometer.The number of TEs per number of live cells plus TEs

is used to calculate the frequency of TE formation. Since all mesophyll cells start off as single cells, the number of septa can be used to estimate the frequency of cell division.

5.5.4. Observation

Of Zinnia Cells: The unusual cell wall thickenings visible under a light microscope identify TEs from other cells, as discussed above.TEs can also be recognised by using phloroglucinol-HCl or fluorochrome-conjugated wheat germ agglutinin to stain lignified secondary cell walls Isolated Z. elegans cells are suitable for Z. elegans isolated cells can be observed using a fluorescence microscope as well as a con-focal laser scanning microscope. Intracellular components, including nuclei, are lysed autonomously as TEs mature. The staining of nuclei with a DNA-specific fluorochrome, DAPI, can be used to monitor this stage of differentiation.

1. Fix the cells by adding glutaraldehyde to a final concentration of 2% (v/v).

2. Add 1/100 volume of 0.2 mg/mL DAPI and incubate briefly in dark. Observe the nuclei under ultraviolet light using a fluorescence microscope.

3. Add 1/1,000 volume of 1 mM SYTO16 to living TEs and incubate for 10 minutes to visualise the nuclei. Use a fluorescence microscope to find out. The fluorescence is detected at 515–545 nm when the dye is activated at 488 nm .[18,19]

6.NEED FOR LOW COST TISSUE CULTURE TECHNOLOGY

Tissue culture technology's commercial applicability is limited due to high production costs.[20,21]As a result, the most difficult part at the moment is to minimise manufacturing costs while increasing production efficiency.[22,23,24]Tissue culture at a low cost is particularly important not just for farmers, but also for large-scale commercial replication on a regular basis [26]Nutrients/media chemicals (plant growth hormones, vitamins and minerals nutrients), plant materials, equipment (culture containers, autoclave, laminar flow, instruments used for micropropagation, pH metre, etc.) and infrastructures (media preparation, inoculation, growth and hardening rooms) are among the various plant tissue culturing components.[27]

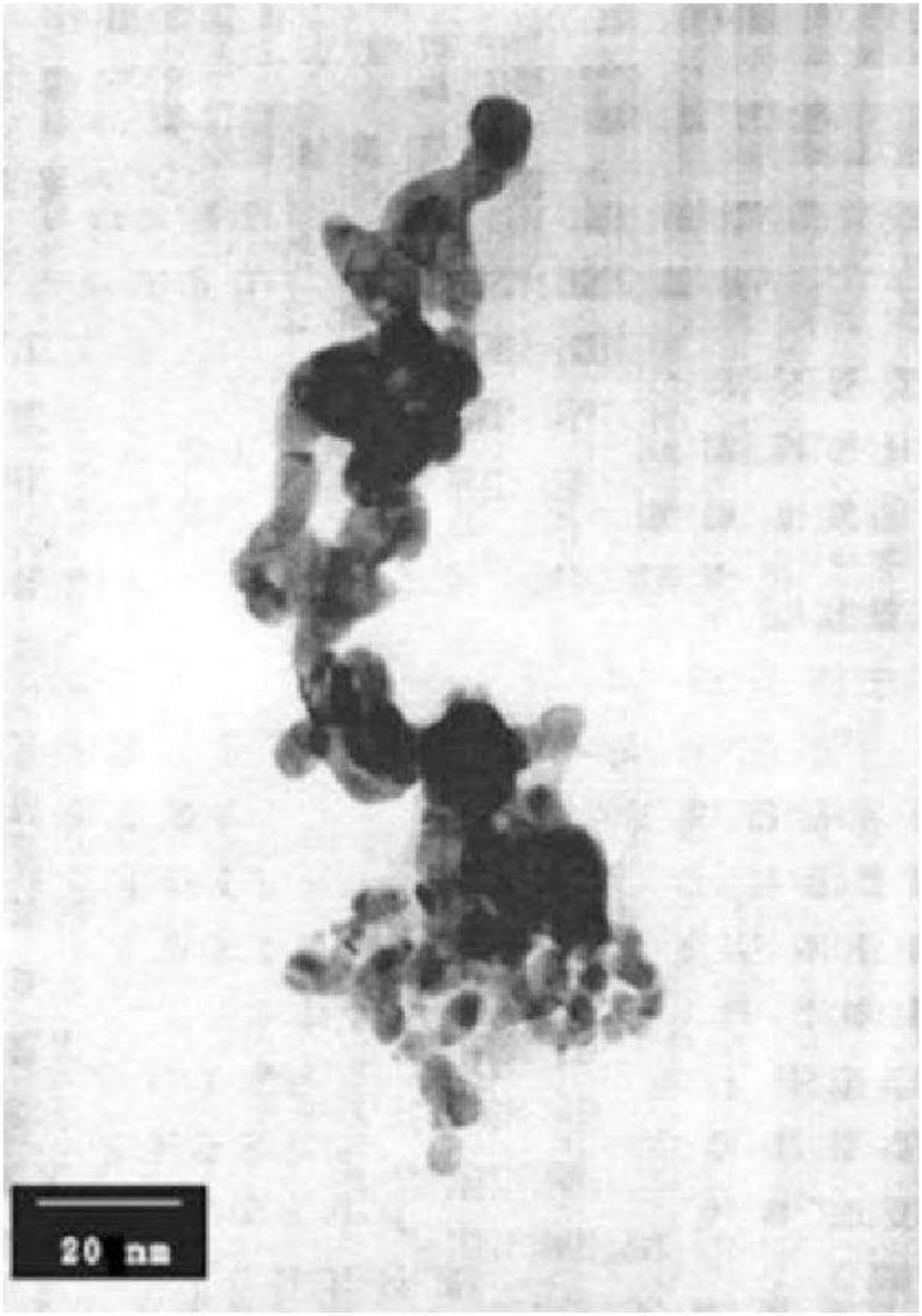
6.Application of nanomaterial in valerian tissue culture(nano silver)

Successful tissue culture of all plants depends on the removal of exogenous and endogenous contaminating microorganisms.[28]Fungi and bacteria are the most common microorganisms to be found on or in plant tissues. To eliminate bacterial contamination during in vitro propagation,different methods have been developed in the last few years.[29]Teixeira da Silva et al.(2003) reported a decrease in explant survival and biomass reduction, malformation of roots and inhibition of shoot formation in chrysanthemum, and also in tobacco endoreduplication by application of antibiotics in the media. Nano silver (NS) has shown to have antibacterial,antifungal and antiviral effects. [30] Studies have demonstrated that silver ions interact with sulfydryl (–SH) groups of proteins as well aswith the bases of DNA leading either to the inhibition of respiratory processes. [31]They interact with a wide range of molecular processes within microorganisms resulting in a range of effects from inhibition of growth, loss of infectivity through cell death.The mechanism depends on both the concentration of silver ions present and the sensitivity of the microbial species to silver.Contact time and temperature can have impact on both the concentration.[32]The effects of NS solution on growth, proliferation rate and rooting were studied and compared with non-NS treated materials.

6.1 Materials and methods

6.1.A nano silver preparation

The nano-particles used in this experiment were silver particles 35 nm (average) in size. Figure 1 show the transmission electron microscopy (TEM) micrograph of silver (Ag) nano-particles. The base working silver (Ag) nano-particles. The base working fluid was pure water. Ag nano-fluids were prepared using a two-step method. Ag nano-particles were prepared first. They were produced using a catalytic chemical vapour deposition method (Nanocid Company Method). The Ag nano-particles were then added to pure water. No surfactant was used in the Ag nano-fluid suspensions. The mixture was prepared using an ultrasonic homogenizer. Nano-fluid concentrations at 25, 50 and 100 mg l-1 were used in this study. Some atomic and physical properties of NS used in this study are presented in Table 1.



Plant materials and culture conditions

Greenhouse grown valerian (Valeriana officinalis L.) mother plants were used in this study. These plants were tested by culturing their stem explants in potato dextrose agar (PDA) medium for internal contamination assay. The explants were first surface sterilized with 70% ethanol for 1 min and 10% Clorox (containing 5.25% sodium hypochlorite) for 1 min and then rinsed four times with sterilized distilled water. The cause of internal contamination was identified with special laboratory methods, in the Department of Plant Pathology, Shiraz University, as Xanthomonas genus. After testing, the mother plants were divided into two groups: with internal contamination (group 1) and without internal contamination (group 2).

Table 1 Properties of NS used in this study

Physical properties		Atomic properties	
Density (near r.t.)[a]	10.49 g cm^{-3}	Oxidation states	1 (Amphoteric oxide)
Liquid density at melting point	9.320 g cm^{-3}	Electronegativity	1.93 (Pauling scale)
Melting point	1,234.93 K (961.78°C, 1,763.2°F)	Ionization energies	First: 731.0 kJ mol^{-1}, second: 2,070 kJ mol^{-1}, third: 3,361 kJ mol^{-1}
Boiling point	2,435 K (2,162°C, 3,924°F)	Atomic radius	160 pm
Heat of fusion	11.28 kJ mol^{-1}	Atomic radius (calc.)	165 pm
Heat of vaporization	258 kJ mol^{-1}	Covalent radius	153 pm
Heat capacity	(25°C) 25.350 J $(mol\ K)^{-1}$	Van der Waals radius	172 pm
Atomic mass	107.8682(2) g mol^{-1}	Crystal structure	Cubic face centred
Electron configuration	[Kr] $4d^{10}\ 5\ s^{1}$		
Electrons per shell	2, 8, 18, 18, 1		

[a] Reproduced with permission from Nanocid Company, Tehran, Iran

For group 1, a number of 20–25 cm stems were cut and transferred to the laboratory immediately. They were cut to the length of about 0.5–1 cm and prewashed in water supplemented with 10 drops of a weak household detergent solution for 10 min and then placed under running tap water for at least 30 min. Nano silver solution at different concentrations (25, 50 and 100 mg l-1) and exposure times (30, 60, 180, 300, 600 and 1,200 min) was used at two stages; before and after surface sterilization along with the control. Initial experiment showed that in high exposure times (300, 600 and 1,200 min) explants turned to bleach.Therefore, high exposure time's results were omitted.For the treatment without surface sterilization, after prewashing in water and keeping under running tap water,nodal segments were dipped at appropriate times and concentrations of NS solution. After this treatment, the explants were rinsed four times with sterilized distilled water. For the treatment before surface sterilization, after dipping explants in NS solution, the explants were surface sterilized (as

mentioned above). For the treatment after surface sterilization, the explants were rinsed with sterilized distilled water and dipped in NS solution with appropriate concentrations at different times. After recut,the sterilized explants were dipped in NS solution before being transferred to the culture vessels. After sterilization,about 1 cm single node explants were cultured on a modified MS (Murashige and Skoog 1962) medium containing salts, organic constituents, 30 g l-1 sucrose,8gl-1 agar and 5 mg l-1 Kin and 0.1 mg l-1 NAA Abdi 2006). The pH of media was adjusted to 5.8 by (0.1 N HCl before autoclaving for 15 min at 121°C and 1.5 kg cm-2 pressure. Cultures were kept under a 16 h photoperiod of 30 mm m-2 s -1 light intensity emitted by two cool white fluorescent lamps at 25 ± 3°C.Explants in group 2 were just prewashed, surface sterilized and cultured without NS treatment. All the cultural conditions in this group were similar to those in group 1.[33]

Data collecting

The percentages of infected explants were recorded 3 days after culture for without surface sterilization treatment. For estimation of size and growth of bacterial and fungal colonies the grades of 1 (lowest) to 5 (highest) contamination were given. For other treatments, the percentages infected explants were recorded 3 weeks after cultures. The experiment was conducted as a completely randomized design in a factorial arrangement with four replications and each replicate with ten explants. Means were compared using Duncan's new multiple range test (DNMRT) at 5% probability level. Impact of the NS on subsequent shoot formation and rooting was assessed in four subcultures with 4 week intervals.[34]

Results

Using NS solution without surface sterilization did not affect the contamination. In control, visible fungal contaminations were observed only 3–5 days after culture,while, the bacterial contaminations were observed 7–10 days after culture, in the other treatments, colony appearance was delayed by at least 6–8 and 12–18 days for fungal and bacterial contaminations, respectively. The size and growth of the colonies varied significantly among treatments. In the control treatment, growth of the colonies was quick; whereas, in the other treatments depending on exposure time and concentration of NS solution the influence on growth was negligible (Table 2).

Table 2 Effects of NS solution on appearance, colony size, growth and fungal and bacterial contamination percentages of *Valeriana officinalis* L. single node explants

Treatments (time + concentration)	Contaminations (%)		Appearance (day)		Size[b]		Growth[c]	
	Fungal	Bacterial	Fungal	Bacterial	Fungal	Bacterial	Fungal	Bacterial
Control	100a[a]	100a	3–5	8–10	5.00a	5.00a	5.00a	5.00a
30 min								
25 mg l^{-1}	97a	98a	12	5	4.50b	4.75ab	4.50b	4.75ab
50 mg l^{-1}	95a	97a	12	5	4.00bc	4.00c	4.00bc	4.00c
100 mg l^{-1}	91ab	97a	14	6	3.50c	4.00c	3.50c	4.00c
60 min								
25 mg l^{-1}	91ab	95a	13	5	3.50c	4.25bc	3.50c	4.25bc
50 mg l^{-1}	91ab	94a	13	7	3.00d	3.75cd	3.00d	3.75cd
100 mg l^{-1}	89ab	88ab	15	7	2.50e	3.00d	2.50e	3.00d
180 min								
25 mg l^{-1}	90ab	78c	14	7	2.75de	2.00e	2.75de	2.00e
50 mg l^{-1}	89ab	71c	17	8	2.00f	1.00f	2.00f	1.20f
100 mg l^{-1}	88ab	68d	18	8	1.00g	1.00f	1.00g	1.00f

[a] In each column, means followed by the same letters are not significantly different using DNMRT at 5% probability level

[b] Ranking from 1 (smallest colony) to 5 (largest colony)

[c] Ranking from 1 (lowest growth) to 5 (the highest growth)

Cultures subjected to NS solution treatment before surface sterilization showed low percentage of disinfected valerian explants (Table 3). In all the treatments, the percentage of fungal contamination was zero. Among the treatments, highest percentages of disinfection (32%) were observed when the explants were dipped in 100 mg l-1 NS solution for 180 min.

Using NS solution after surface sterilization was successful. Treatment with 100 mg l-1 of NS for 180 min after rinsing the explants in sterilized distilled water was the most successful disinfection treatment. This treatment had significant differences with other treatments. The 11% contamination left after this treatment was bacterial contamination (data not shown). Also, this treatment did not have any negative impact on measured characters in micropropagation of valerian in four subsequent subcultures.

Group 2

As mentioned above, explants in this group did not show any contamination during culture period. Comparing this group with explants obtained from NS solution treatment after surface sterilization did not show any significant differences in measured characters (proliferation rate,

leaf number, percentage of fresh weight, number of rooted explants, number of roots and root length) in micropropagation of valerian in four subsequent subcultures (data not shown)

Table 3 Comparisons among different treatments used for disinfection of *Valeriana officinalis* L. single node explants 3 weeks after culturing on the medium

Treatments (time + concentration)	Before surface sterilization, Contamination (%)	After surface sterilization, Contamination (%)
Control	98a[a]	99a
30 min		
25 mg l^{-1}	97b	90b
50 mg l^{-1}	97b	91b
100 mg l^{-1}	96b	82c
60 min		
25 mg l^{-1}	95b	80c
50 mg l^{-1}	94b	71d
100 mg l^{-1}	88bc	63e
180 min		
25 mg l^{-1}	78c	32f
50 mg l^{-1}	73c	28f
100 mg l^{-1}	68cd	11g

[a] In each column, means followed by the same letters are not significantly different using DNMRT at 5% probability level

Discussion

Novel approaches for controlling the contamination in plant tissue culture, screening the contaminants, disinfestation of explant methods using activated charcoal or diethylpyrocarbonate, elimination of microbial contaminants using density gradient centrifugation, flexible container system,repeated subculture, low free-water medium, acidification,egg white lysozyme and antibiotics are reviewed in the book edited by Herman.[7] Silver and its compounds have long been used as antimicrobial agents.[8,9] The most important silver compound currently in use is silver sulfadiazine (AgSD), although silver metal, silver acetate, silver nitrate and silver protein have antimicrobial effect too Using AgNO3 as silver compound against infection in tissue culture is common .[34]

Explant methods using activated charcoal or diethylpyrocarbonate, elimination of microbial contaminants using density gradient centrifugation, flexible container system,repeated subculture, low free-water medium, acidification,egg white lysozyme and antibiotics are reviewed in the book edited by Herman.[7] Silver and its compounds have long been used as antimicrobial agents.[35,36] The most important silver compound currently in use is silver sulfadiazine (AgSD), although silver metal, silver acetate, silver nitrate and silver protein have antimicrobial effect too Using AgNO3 as silver compound against infection in tissue culture is common .[34]

Our results showed that silver in nano size can similarly control the bacterial infection in tissue culture conditions. Also, subcultures indicated that bacterial contaminations were removed because the late appearance of contamination was not observed in subsequent subcultures. In general, using NS solution after surface sterilization had acceptable influence on the bacterial contaminants control without any adverse effects on growth characters in micropropagation of valerian. However, it was not effective in controlling the fungi in this experiment. The differences in the effects of NS treatment before and after surface sterilization may be due to the presence of NS in NS solution treatment after surface sterilization at the cut end of the explants inside the medium.After recut, the sterilized explants were dipped in NS solution before being transferred to the culture vessels.However, in NS solution treatment, before surface sterilization explants were washed with distilled water and then transferred to the medium. A method has been suggested by Salehi and Khosh-Khui (1997) for controlling bacterial contamination in miniature roses. They used gentamicin solution after surface sterilization. Using NS may be more convenient and less toxic than using antibiotics in the medium. Furthermore, using other methods for controlling the infection like first acidification of the medium and later regulation of pH to normal condition and price may be time consuming methods in tissue culture techniques.[37,38]Showing acceptable antibacterial activity in this investigation is in agreement with the results obtained by other investigators . [39,40]

Since, this is the first report of NS application in the tissue culture methods, further studies are needed on using this chemical in in vitro culture of other species.

References

1.E Evans, JOD Coleman and A Kearns, Plant Cell Culture, Bios Scientific Publishers, Taylor & Francis Group, London, p.1, 2003.

2.S S Bhojwani and M K Razdan, Plant Tissue Culture: Theory and Practice, A revised edition, Elsevier, New Delhi, p.3, 2004.

3.K. Keren , R. S. Berman , E. Buchstab , U. Sivan and E. Braun , Science, 2003, 302 , 1380 —1382

4.M. K. Sarmast and H. Salehi , Mol. Biotechnol., 2016, 58 , 441

5.B. Ruttkay-Nedecky , O. Krystofova , L. Nejdl and V. Adam , J. Nanobiotechnol., 2017

6.Bijalwan, P. (2021). Plant Tissue Culture-A New Tool for Vegetable Improvement (Indian scenario): A Review. Agricultural Reviews, 42(2).

7.Pierik, R.L.M. (1997) In vitro culture of higher plants. 4th ed. Kluwer, Dordrecht. 348 p.

1. Fukuda, H., and Komamine, A. (1980)

Establishment of an experimental system for the tracheary element differentiation from single cells isolated from the mesophyll of Zinnia elegans. Plant Physiol 65, 57–60.

9.Kärkönen, A., Koutaniemi, S., Mustonen, M.,

Syrjänen, K., Brunow, G., Kilpeläinen, I., Teeri, T.H., and Simola, L. K. (2002) Lignification related enzymes in Picea abies suspension cul-tures. Physiol Plant 114, 343–353.

10.Simola, L.K., and Santanen, A. (1990)

Improvement of nutrient medium for growth

And embryogenesis of megagametophyte and embryo callus lines of Picea abies Physiol. Plant

80, 27–35.

11.Bajaj, Y.P.S. (1996) Plant protoplasts and

Genetic engineering VII. Springer, Berlin. 317 p. ISBN 3-540-60876-1.

12. Murashige, T., and Skoog, F. (1962) A revised medium for rapid growth and bio assays with tobacco tissue cultures. Physiol Plant 15,473–497.

13. Gamborg, O. L., Miller, R. A., and Ojima, K. (1968) Nutrient requirements of suspension cultures of soybean root cells. Exp Cell Res50, 151–158.

14. Widholm, J. M. (1972) The use of fluorescein diacetate and phenosafranine for determining viability of cultured plant cells. Stain Technol47, 189–194.

15. Huang, C. N., Cornejo, M. J., Bush, D. S.,

And Jones, R. L. (1986) Estimating viability

Of plant protoplasts using double and single staining. Protoplasma 135, 80–87.

16. Ohdaira, Y., Kakegawa, K., Amino, S.,

Sugiyama, M., and Fukuda, H. (2002) Activity of cell-wall degradation associated with differentiation of isolated mesophyll cells of Zinnia elegans into tracheary elements. Planta 215,177–184.

17. Siegel, S. M. (1953) On the biosynthesis of lignin. Physiol Plant 6, 134–139.

18. Hogetsu, T. (1990) Detection of hemicelluloses specific to the cell wall of tracheary elements and phloem cells by fluorescein-conjugated lectins.Protoplasma 156, 67–73.

19. Obara, K., Kuriyama, H., and Fukuda, H.

(2001) Direct evidence of active and rapid

Nuclear degradation triggered by vacuole.

20..Kozai T, Kubota C, Jeong BR, et al. Plant cell tissue and organ culture. 1997;51:49–56.

21. Babbar SB, Jain R curr. Microbol.2006;52.287:292

22.Anderson WC, Meagher GW. Horticult. Sci. 1977;126:543-544

23. Sluis CJ, Walker KA. Newslett. Int. Ass. Plant Tissue Cult. 1985;47:2-12

24. Donnan A. Determining and minimizing production costs In: Tissue culture as a plant production system for horticultural crops Zimmerman RH, Grierbach RJ, Hammerschlag FA, Lawson RHeds, Martinus Nijhoff Publishers Boston. 1986; 163-173

25.Thro MA, Roca W, Restrepo J, Caballero H, Poats S, Escobar R, Mafla G, Hernández C, et al. Cell Dev. Biol. Plant. 1999;35:382-387.

26. Ganapathi TR, Mohan JSS, Suprasanna P, Bapat VA, Rao P S, et al current science 1995:68:640-665

27.Buckley PM, Reed BM (1994) Antibiotic susceptibility of plant associated bacteria. HortScience 29:434(Abst)

28.Constantine DR (1986) Micropropagation in the commercial environment. In: Withers L, Alderson PG (eds)Plant tissue culture and its agricultural applications. Butterworth,London, pp 175–186

29.Sondi I, Salopek-Sondi B (2004) Silver nano particles as antimicrobial agent: A case study on E. coli as a model for Gram-negative bacteria. J Colloid Interface Sci 275:177–182.

30.Nomiya K, Yoshizawa A, Tsukagoshi K, Kasuga NC, Hirakava S,Watanabe J (2004) Synthesis and structural characterization of silver (I), aluminium (III) and cobalt (II) complexes with 4-isopropyltropolone (hinokitiol) showing noteworthy biological activities. Action of silver (I)-oxygen bonding complexes on the antimicrobial activities. J Inorg Biochem 98:46–60.

31.Abdi, G., Salehi, H., & Khosh-Khui, M. (2008). Nano silver: a novel nanomaterial for removal of bacterial contaminants in valerian (Valeriana officinalis L.) tissue culture. Acta Physiologiae Plantarum, 30(5), 709-714.

32.Dibrov P, Dzioba J, Khoosheh K, Gosink K, Claudia C (2002)Chemiosmotic mechanism of antimicrobial activity of Ag+ inVibrio cholerae. Antimicrob Agents Chemother 46:2668–2670

33.Herman EB (ed) (1996) Microbial contamination of plant tissue cultures. Agritech Consultants Inc, Shrub Oak, USA, 84 p.

34.Brown MRW, Anderson RA (1968) The bactericidal effect of silver ions on Pseudomonas aeruginosa. J Pharm Pharmacol 20(Suppl):1S–3S

35.Russell AD, Hugo WB (1994) Antimicrobial activity and action of silver. Prog Med Chem 31:351–371.

36..Leifert C, Cammota H, Waites WM (1992) Effect of combinations of antibiotics on micropropagated Clematis, Delphinium, Hosta,Iris and Photinia. Plant Cell Tissue Organ Cult 29:153–160

37..Hussain S, Lane SD, Price DN (1994) A preliminary evaluation of the use of microbial culture filtrates for the control of contaminants in plant tissue culture systems. Plant Cell Tissue Organ Culture 36:45–51

38.Sondi I, Salopek-Sondi B (2004) Silver nano particles as antimicrobial agent: A case study on E. coli as a model for Gram-negative bacteria. J Colloid Interface Sci 275:177–182

39.Thurmann RB, Gerba CP (1989) The molecular mechanisms of copper and silver ion disinfection of bacteria and viruses. Critic Rev Environ Cont 18:295–315.

40.Smart DR, Ferro A, Ritchie K, Bugbee BG (1995) On the use of Antibiotics to reduce rhizoplane microbial populations in root physiology and ecology investigations. Physiol Plant 95:533–540