



**SUBMITTED BY:**

DEEPALI

**ID:** BTBTI20137

**ROLL NO.:** 2016736

**BRANCH:** B.TECH IT (A)

**BATCH:** 2

# *DATA STRUCTURES LAB RECORD*

**SUBMITTED TO:**

DR. VAIBHAV VYAS

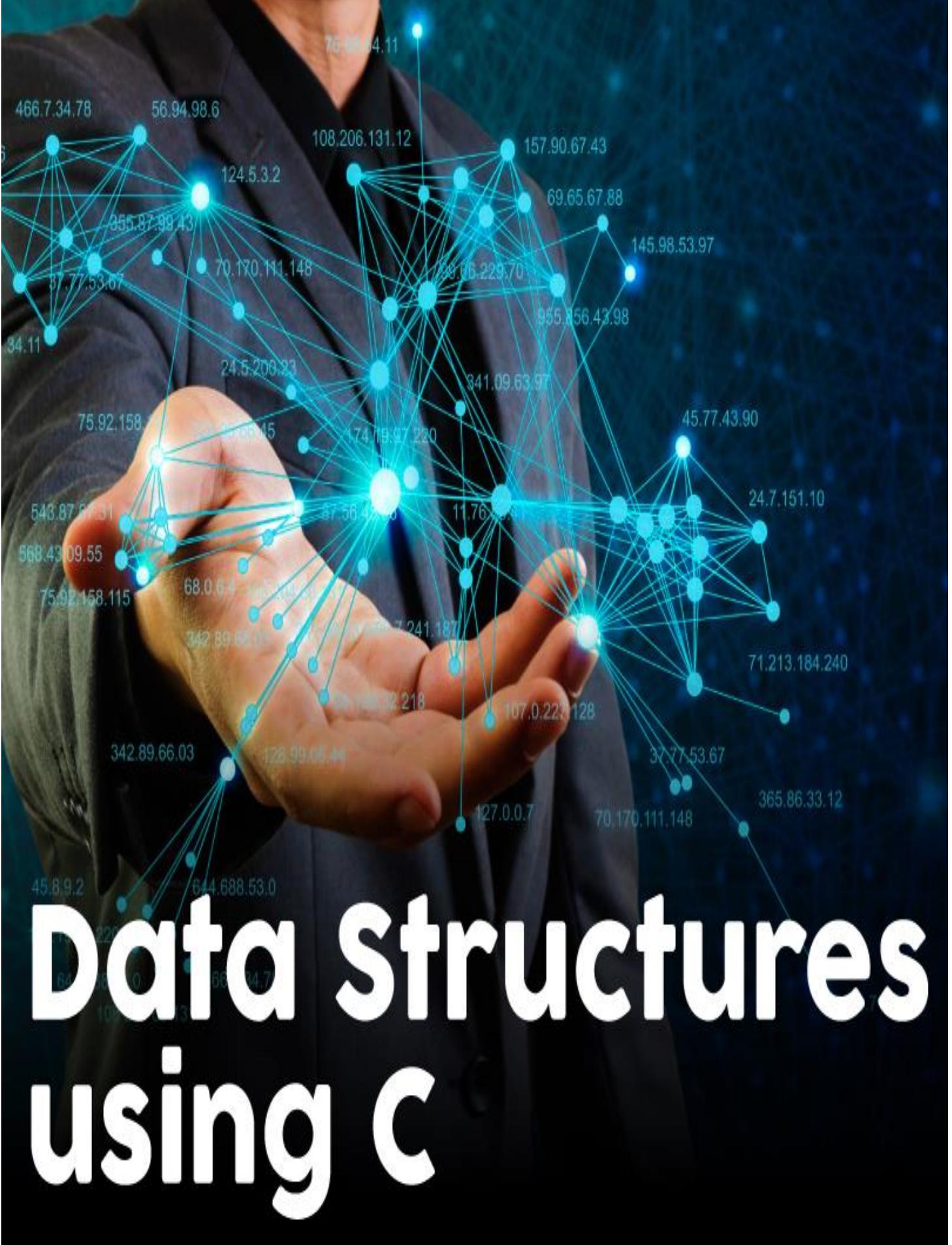
(ASSOCIATE PROFESSOR, COMPUTER SCIENCE)

**SUBJECT:** DS LAB

**SUBJECT CODE:** CS209L

**TOTAL PAGES SUBMITTED:** 160

**SUBMISSION DATE:** 07-12-2021



# Data Structures using C

# INDEX

S.NO.	TOPICS COVERRED	QUESTIONS COVERRED	PAGE NO.
1.	PROGRAMS BASED ON ARRAY	Double data type array Reversing array elements Menu driven array Unique array elements Addition of matrices Multiplication of matrices Transpose of matrices	6 6 - 7 7 - 11 11 - 12 12 - 13 13 - 15 15 - 16
2.	PROGRAMS BASED ON STRUCTURE	Student record Customer record	16 - 19 19 - 24
3.	PROGRAMS BASED ON POINTERS	Read and print string Fibonacci series	24 - 25 25 - 26
4.	PROGRAM BASED ON STACK	Linear stack Decimal to binary String reverse Parentheses check Infix to postfix Infix to prefix Postfix evaluation	26 - 30 30 - 31 31 - 32 32 - 33 33 - 35 35 - 37 37 - 38
5.	PROGRAM BASED ON QUEUE	Linear queue Circular queue Priority queue	38 - 42 42 - 46 46 - 50
6.	PROGRAM	Linear search	51 - 52

	<b>BASED ON SEARCHING</b>	Binary search	52-53
7.	<b>PROGRAM BASED ON SORTING</b>	Bubble sort	53-54
		Selection sort	54-56
		Insertion sort	56-57
		Quick sort	57-59
8.	<b>PROGRAM BASED ON LINKED LIST</b>	Singly linked list with all basic operations	59-67
		Create, reversal and display	67-69
		Create, search, sort and display	69-73
		Create, split and display	73-76
		Create, merge and display	76-80
		Create, duplicate removal, move node, swap node, insert sorted, split in odd and even and display	80-88
		Doubly linked list with all basic operations	88-95
		Circular linked list with all basic operations	96-102
		Header linked list with all basic operations	102-108
		Generic linked list with all basic operations	108-115
		Polynomial representation, operations and display	115-119
		Sparse matrix representation	119-123

		<i>and display</i> 1. Array 2. Linked List	
9.  <b>PROGRAM BASED ON BINARY SEARCH TREE</b>	<i>BST with all basic operations and non-recursive traversals</i>	<i>123-130</i>	
		<i>BST with all basic operations and recursive traversals</i>	<i>130-136</i>
	<i>BST create, traversal and other remaining operations</i> 1. Non- Recursive 2. Recursive	<i>136-157</i>	
		<i>Heap sort</i>	<i>157-159</i>

# PROGRAMS BASED ON ARRAY

## Q1. WAP for reading and writing of double datatype array.

Ans1.

### C PROGRAM –

```
#include<stdio.h>
int main()
{
    double a[20];
    int i,num;
    printf("\nEnter number of elements in the array : ");
    scanf("%d",&num);
    printf("\n*****Enter elements*****\n");
    for(i=0;i<num;i++)
        scanf("%lf",&a[i]);
    printf("\n*****Entered elements*****\n");
    for(i=0;i<num;i++)
        printf("%lf\t",a[i]);
    return 0;
}
```

### OUTPUT -

```
BT1 DOS LAB\1doublearr.exe
Enter number of elements in the array : 10
*****Enter elements*****
0.000000 1.000000 2.000000 3.000000 4.000000 5.000000 6.000000 7.000000 8.000000 9.000000 2.000000
*****Entered elements*****
0.000000 1.000000 2.000000 3.000000 4.000000 5.000000 6.000000 7.000000 8.000000 9.000000 2.000000
Process exited after 16.92 seconds with return value 0
Press any key to continue . . .
```

## Q2. WAP for reversing an array's element.

Ans2.

### C PROGRAM –

```
#include<stdio.h>
int main()
{
    int a[10],b[10],i,j,n;
    printf("\nEnter number of elements in array : ");
    scanf("%d",&n);
    printf("\nEnter elements : ");
    for (i=0;i<n;i++)
        scanf("%d",&a[i]);
    printf("\nOriginal array : \n");
```

```

for (i=0;i<n;i++)
    printf("%d\t",a[i]);
for (i=0,j=n-1;i<n;i++,j--)
    b[j]=a[i];
printf("\nReversed array : \n");
for(j=0;j<n;j++)
    printf("%d\t",b[j]);
return 0;
}

```

### OUTPUT-

```

W1 CDS LAB3Imrevese
Enter number of elements in array : 5
Enter elements :
3 5 6 2
Original array :
3      5      6      2
Reversed array :
6      5      3      1
Process exited after 13.8 seconds with return value 0
Press any key to continue . . .

```

### Q3.WAP to implement array as Data Structures with following operations:

(use switch case for menu driven program)

- (i) Creation
- (ii) Insert First, Last and Random
- (iii) Delete First, Last and Random
- (iv) Display
- (v) Searching
- (vi) Sort (using Insertion Sort Algorithm)
- Optional-
- (vii)Splitting (into 2 equal parts)
- (viii) Merging (two arrays into 1).

Ans3.

### C PROGRAM –

```

#include<stdio.h>
void main()
{
    int A[20],B[20],C[40],n=0,i,j,x,pos,ch,t,found=0,m;
    do
    {
        printf("\n1.Insert first, \n2.Insert last, \n3.Insert random");
        printf("\n4.Delete first, \n5.Delete last, \n6.Delete random");
        printf("\n7.Display, \n8.Exit, \n9.Search, \n10.Insertion sort");
        printf("\n11.Splitting, \n12.Direct Merging");
        printf("\nEnter your choice : ");
        scanf("%d",&ch);
        switch(ch)

```

```

{
    case 1:
        printf("\nWe are in case 1");
        printf("\nRead the value to insert : ");
        scanf("%d",&x);
        for(i=n;i>0;i--)
        {
            A[i]=A[i-1];
        }
        A[0]=x;
        n++;
        break;

    case 2:
        printf("\nWe are in case 2");
        printf("\nRead the value to insert : ");
        scanf("%d",&x);
        A[n++]=x;
        break;

    case 3:
        printf("\nWe are in case 3");
        printf("\nRead the value to insert : ");
        scanf("%d",&x);
        printf("\nRead the position to insert : ");
        scanf("%d",&pos);
        for(i=n;i>(pos-1);i--)
        {
            A[i]=A[i-1];
        }
        A[pos-1]=x;
        n++;
        break;

    case 4:
        printf("\nWe are in case 4");
        x=A[0];
        for(i=0;i<(n-1);i++)
        {
            A[i]=A[i+1];
        }
        printf("\ndeleted element is %d",x);
        n--;
        break;

    case 5:
        printf("\nWe are in case 5");
        x=A[n-1];
        printf("\ndeleted element is %d",x);
        n--;
        break;

    case 6:
        printf("\nWe are in case 6");
        printf("\nRead the position to delete : ");
        scanf("%d",&pos);
        for(i=pos-1;i<n-1;i++)
        {

```

```

        x=A[pos-1];
        A[i]=A[i+1];
    }
    printf("\ndeleted element is %d",x);
    n--;
    break;
case 7:
    printf("\nWe are in case 7\n");
    if(n==0)
        printf("\nNo element to display");
    else
        printf("Elements are : ");
    for(i=0;i<n;i++)
        printf("%d ",A[i]);
    printf("\n");
    break;
case 8:
    printf("\nWe are in case 8");
    break;
case 9:
    printf("\nWe are in case 9");
    printf("\nEnter element to search : ");
    scanf("%d",&x);
    for(i=0;i<n;i++)
    {
        if(A[i]==x)
        {
            printf("\nElement found at location %d",++i);
            found=1;
        }
    }
    if(found==0)
        printf("\nElement not found");
    break;
case 10:
    printf("\nWe are in case 10");
    for(i=1;i<n;i++)
    {
        t=A[i];
        j=i-1;
        while((j>=0)&&(A[j]>t))
        {
            A[j+1]=A[j];
            j--;
        }
        A[j+1]=t;
    }
    break;
case 11:

```

```

        printf("\nWe are in case 11");
        printf("\nEnter position to split : ");
        scanf("%d",&x);
        for(i=0;i<x;i++)
            B[i]=A[i];
        for(i=x;i<n;i++)
            C[i]=A[i];
        printf("\nAfter splitting : ");
        printf("\nArray 1 : ");
        for(i=0;i<x;i++)
            printf("%d ",B[i]);
        printf("\nArray 2 : ");
        for(i=x;i<n;i++)
            printf("%d ",C[i]);
        break;
    case 12:
        printf("\nWe are in case 12");
        printf("\nEnter size of array : ");
        scanf("%d",&m);
        printf("\nEnter elements : ");
        for(i=0;i<m;i++)
            scanf("%d",&B[i]);
        for(i=0;i<n;i++)
            C[i]=A[i];
        for(i=n,j=0;i<(m+n);i++,j++)
            C[i]=B[j];
        printf("\nMerged array becomes : ");
        for(i=0;i<(m+n);i++)
            printf("%d ",C[i]);
        break;
    default:
        printf("\nIt is a default case");
    }
}while(ch!=0);
}

```

## OUTPUT-

```

E:\FDS5 LAB\5menu.exe
1.Insert first,
2.Insert last,
3.Insert random,
4.Delete first,
5.Delete last,
6.Delete random,
7.Display,
8.Exit,
9.Search,
10.Insertion sort,
11.Splitting,
12.Direct Merging
Enter your choice : 1

We are in case 1
Read the value to insert : 5
1.Insert first,
2.Insert last,
3.Insert random,
4.Delete first,
5.Delete last,
6.Delete random,
7.Display,
8.Exit,
9.Search,
10.Insertion sort,
11.Splitting,
12.Direct Merging
Enter your choice : 2

We are in case 2
Read the value to insert : 6
1.Insert first,
2.Insert last,
3.Insert random,
4.Delete first,
5.Delete last,
6.Delete random,
7.Display,
8.Exit,
9.Search,
10.Insertion sort,
11.Splitting,
12.Direct Merging
Enter your choice : 3

We are in case 3

```

```

# EUS LABS\menu.exe
We are in case 3
Read the value to insert : 2
Read the position to insert : 2
1.Insert first,
2.Insert last,
3.Insert random
4.Delete first,
5.Delete last,
6.Delete random
7.Display,
8.Exit,
9.Search,
10.Insertion sort
11.Splitting,
12.Direct Merging
Enter your choice : 7

We are in case 7
Elements are : 5 2 6
1.Insert first,
2.Insert last,
3.Insert random
4.Delete first,
5.Delete last,
6.Delete random
7.Display,
8.Exit,
9.Search,
10.Insertion sort
11.Splitting,
12.Direct Merging
Enter your choice : 10

We are in case 10
Elements are : 5 2 6
1.Insert first,
2.Insert last,
3.Insert random
4.Delete first,
5.Delete last,
6.Delete random
7.Display,
8.Exit,
9.Search,
10.Insertion sort
11.Splitting,
12.Direct Merging
Enter your choice : 7

We are in case 7
Elements are : 5 2 6
1.Insert first,
2.Insert last,
3.Insert random
4.Delete first,
5.Delete last,
6.Delete random
7.Display,
8.Exit,
9.Search,
10.Insertion sort
11.Splitting,
12.Direct Merging
Enter your choice : 11

We are in case 11
Read the position to split : 2
After splitting :
Array Z : 9
Array Z : 6
1.Insert first,
2.Insert last,
3.Insert random
4.Delete first,
5.Delete last,
6.Delete random
7.Display,
8.Exit,
9.Search,
10.Insertion sort
11.Splitting,
12.Direct Merging
Enter your choice : 5

We are in case 5
Selected element is 6
1.Insert first,
2.Insert last,
3.Insert random
4.Delete first,
5.Delete last,
6.Delete random
7.Display,
8.Exit,
9.Search,
10.Insertion sort
11.Splitting,
12.Direct Merging
Enter your choice : 5

We are in case 5
Selected element is 6
1.Insert first,
2.Insert last,
3.Insert random
4.Delete first,
5.Delete last,
6.Delete random
7.Display,
8.Exit,
9.Search,
10.Insertion sort
11.Splitting,
12.Direct Merging
Enter your choice : 8

We are in case 8
Process exited after 57.97 seconds with return value 8
Press any key to continue . . .

```

#### Q4. WAP to read/print 10 unique array elements (no duplicate elements are allowed).

Ans4.

#### C PROGRAM –

```

#include<stdio.h>
int main()
{
    int i=1,j,a[20],n=10,c=0,flag=0;
    printf("\nEnter first element : ");
    scanf("%d",&a[0]);
    while(c<n-1)
    {

```

```

printf("\nEnter element : ");
scanf("%d",&a[i]);
for(j=i-1;j>=0;j--)
{
    if(a[i]==a[j])
    {
        flag=1;
        printf("\nENTER DIFFERENT ELEMENT");
        break;
    }
    else
        flag=0;
}
if(flag!=1)
{
    c++;
    i++;
}
printf("\nUnique 10 elements array : \n");
for(i=0;i<n;i++)
    printf("%d ",a[i]);
return 0;
}

```

### OUTPUT-

```

EDS LAB\uniquearray.exe
Enter first element : 3
Enter element : 3
ENTER DIFFERENT ELEMENT
Enter element : 1
Enter element : 5
Enter element : 1
ENTER DIFFERENT ELEMENT
Enter element : 7
Enter element : 2
ENTER DIFFERENT ELEMENT
Enter element : 9
Enter element : 0
Enter element : 8
Enter element : 11
Enter element : 8
ENTER DIFFERENT ELEMENT
Enter element : 22
Unique 10 elements array :
3 2 1 5 7 9 0 8 11 22
Process exited after 27.62 seconds with return value 0
Press any key to continue . . .

```

### Q5. WAP for addition of two matrices.

Ans5.

### C PROGRAM –

```

#include<stdio.h>
int main()
{
    int m,n,i,j,a[10][10],b[10][10],c[10][10];
    printf("\nEnter rows and columns of matrix : ");
    scanf("%d%d",&m,&n);
    printf("\nEnter matrix 1 elements : ");

```

```

for(i=0;i<m;i++)
    for(j=0;j<n;j++)
        scanf("%d",&a[i][j]);
    printf("\nMatrix 1 : \n");
for(i=0;i<m;i++)
{
    for(j=0;j<n;j++)
        printf("%d ",a[i][j]);
    printf("\n");
}
printf("\nEnter matrix 2 elements : ");
for(i=0;i<m;i++)
    for(j=0;j<n;j++)
        scanf("%d",&b[i][j]);
    printf("\nMatrix 2 : \n");
for(i=0;i<m;i++)
{
    for(j=0;j<n;j++)
        printf("%d ",b[i][j]);
    printf("\n");
}
for(i=0;i<m;i++)
    for(j=0;j<n;j++)
        c[i][j]=a[i][j]+b[i][j];
printf("\nSum of matrix 1 and matrix 2 : \n");
for(i=0;i<m;i++)
{
    for(j=0;j<n;j++)
        printf("%d ",c[i][j]);
    printf("\n");
}
return 0;
}

```

### OUTPUT-

```

E:\DS LAB\90\adimrme
Enter rows and columns of matrix 1 : 2
1
2
3
4
Matrix 1 :
1 2 3
0 3 2
Enter matrix 2 elements : 1
2
3
4
Matrix 2 :
1 7 8
1 1 2
Sum of matrix 1 and matrix 2 :
2 11 11
2 4 4
Process exited after 20.65 seconds with return value 0
Press any key to continue . .

```

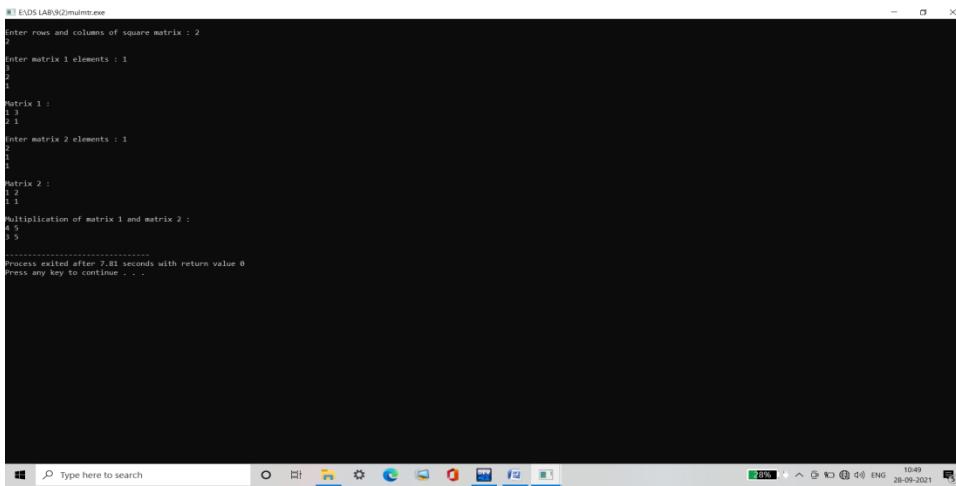
### Q6. WAP for multiplication of two matrices.

Ans6.

## C PROGRAM –

```
#include<stdio.h>
int main()
{
    int r,c,i,j,k,a[10][10],b[10][10],m[10][10];
    printf("\nEnter rows and columns of square matrix : ");
    scanf("%d%d",&r,&c);
    printf("\nEnter matrix 1 elements : ");
    for(i=0;i<r;i++)
        for(j=0;j<c;j++)
            scanf("%d",&a[i][j]);
    printf("\nMatrix 1 : \n");
    for(i=0;i<r;i++)
    {
        for(j=0;j<c;j++)
            printf("%d ",a[i][j]);
        printf("\n");
    }
    printf("\nEnter matrix 2 elements : ");
    for(i=0;i<r;i++)
        for(j=0;j<c;j++)
            scanf("%d",&b[i][j]);
    printf("\nMatrix 2 : \n");
    for(i=0;i<r;i++)
    {
        for(j=0;j<c;j++)
            printf("%d ",b[i][j]);
        printf("\n");
    }
    for(i=0;i<r;i++)
    {
        for(j=0;j<c;j++)
        {
            m[i][j]=0;
            for(k=0;k<c;k++)
                m[i][j]+=a[i][k]*b[k][j];
        }
    }
    printf("\nMultiplication of matrix 1 and matrix 2 : \n");
    for(i=0;i<r;i++)
    {
        for(j=0;j<c;j++)
            printf("%d ",m[i][j]);
        printf("\n");
    }
    return 0;
}
```

## OUTPUT-



```
#1 FUS LAB\Q2\multmatr.exe
enter rows and columns of square matrix : 2
|
enter matrix 1 elements :
|
|
Matrix 1 :
1 3
0 1
|
enter matrix 2 elements :
|
|
Matrix 2 :
2 1
1 1
|
Multiplication of matrix 1 and matrix 2 :
0 5
0 5
Process exited after 7.81 seconds with return value 0
Press any key to continue . . .
```

## Q7. WAP for transpose of a matrix.

Ans7.

### C PROGRAM –

```
#include<stdio.h>
int main()
{
    int a[5][5],trans[5][5],i,j,c,r,sor;
    printf("Enter number of row : ");
    scanf("%d",&r);
    printf("Enter number of column : ");
    scanf("%d",&c);
    printf("Enter the matrix element : \n");
    for(i=0;i<r;i++)
    {
        for(j=0;j<c;j++)
        {
            scanf("%d",&a[i][j]);
        }
    }
    for(i=0;i<r;i++)
    {
        for(j=0;j<c;j++)
        {
            printf("%d\t",a[i][j]);
        }
        printf("\n");
    }
    for(i=0;i<r;i++)
    {
        for(j=0;j<c;j++)
        {
            trans[j][i]=a[i][j];
        }
    }
    printf("Transpose of the matrix :\n");
```

```

for(i=0;i<c;i++)
{
    for(j=0;j<r;j++)
    {
        printf("%d\t",trans[i][j]);
    }
    printf("\n");
}
return 0;
}

```

### OUTPUT-

```

E:\DS LAB\10trans.c
Enter number of row : 2
Enter number of column : 3
Enter the matrix element :
A
2      3
5      6
Transpose of the matrix :
X
2      5
5      6
Process exited after 11.00 seconds with return value 0
Press any key to continue . . .

```

## PROGRAMS BASED ON STRUCTURE

---

**Q8. WAP using array of structure (holding studid, studname). This application should be capable of performing following operations (using functions).**

- 1. Read operation- Reads a single student's record.**
- 2. Print operation – Prints records of all the students.**
- 3. Sort operation - Sorts records on id/name as per user's choice.**

Ans8.

### C PROGRAM –

```

#include<stdio.h>
#include<stdlib.h>
#include<string.h>
struct studrecord
{
    int studid;
    char studname[20];
};
int Read(struct studrecord x[]);
void Print(struct studrecord y[],int n);
void Sort(struct studrecord y[],int n);
void main()
{
    struct studrecord s[100];

```

```

int ch,n=0;
static int i=0;
do
{
    printf("\n1.Read \n2.Print \n3.Sort \n4.Exit");
    printf("\nEnter your choice : ");
    scanf("%d",&ch);
    switch(ch)
    {
        case 1:
            n=Read(s);
            break;
        case 2:
            Print(s,n);
            break;
        case 3:
            Sort(s,n);
            break;
        case 4:
            printf("\nExiting.....!");
            break;
        default:
            printf("\nWrong choice entered!");
    }
}while(ch!=4);
}
int Read(struct studrecord x[])
{
    static int i=0;
    printf("\nRecord %d\n",i+1);
    printf("ID : ");
    scanf("%d",&x[i].studid);
    fflush(stdin);
    printf("NAME : ");
    gets(x[i].studname);
    return (++i);
}
void Print(struct studrecord y[],int n)
{
    int j;
    if(n==0)
    {
        printf("\n.....NO DETAIL FOUND..... \n");
    }
    else
    {
        printf("\n.....DETAILS.....\n");
        for(j=0;j<n;j++)
        {

```

```

    printf("Record no. %d\n",j+1);
    printf("ID : %d\n",y[j].studid);
    printf("NAME : %s\n",y[j].studname);
    printf("\n");
}
}
}
void Sort(struct studrecord y[],int n)
{
    struct studrecord t;
    int j,k,ch;
    if(n==0)
    {
        printf("\n.....NO DETAIL FOUND FOR SORTING..... \n");
    }
    else
    {
        do
        {
            printf("\n1.Sort on name \n2.Sort on id");
            printf("\nEnter your choice : ");
            scanf("%d",&ch);
            switch(ch)
            {
                case 1:
                    for(j=0;j<n;j++)
                        for(k=j+1;k<n;k++)
                            if (strcmp(y[j].studname,y[k].studname)>0)
                            {
                                t=y[j];
                                y[j]=y[k];
                                y[k]=t;
                            }
                    Print(y,n);
                    break;
                case 2:
                    for(j=0;j<n;j++)
                        for(k=j+1;k<n;k++)
                            if (y[j].studid>y[k].studid)
                            {
                                t=y[j];
                                y[j]=y[k];
                                y[k]=t;
                            }
                    Print(y,n);
                    break;
                default :
                    printf("\nWrong choice entered!");
            }
        }
    }
}

```

```

}while((ch!=1)&&(ch!=2));
}
}

```

### OUTPUT-

```

1.Read
2.Print
3.Sort
4.Exit
Enter your choice : 2
.....NO DETAIL FOUND.....
1.Read
2.Print
3.Sort
4.Exit
Enter your choice : 1
Record 1
ID : 02
NAME : Deepali Singh
1.Read
2.Print
3.Sort
4.Exit
Enter your choice : 1
Record 2
ID : 03
NAME : Anni Jindal
1.Read
2.Print
3.Sort
4.Exit
Enter your choice : 3
.....DETAILS.....
Record no. 1
ID : 02
NAME : Deepali Singh
Record no. 2
ID : 03
NAME : Anni Jindal
1.Read
2.Print
3.Sort
4.Exit
Enter your choice : 4
Exiting.....

```

### Q9. WAP structure Customer with following operations:

(Do with the help of functions only)

1. Hold record of 50 customers.
2. Print record of 50 customers.
3. Sort records on name / id / wallet as / des sort.
4. Search a particular customer in your records.
5. Update any customer's record.
6. Sum all customer's wallet amount.

Ans9.

### C PROGRAM –

```

#include<stdio.h>
#include<stdlib.h>
#include<string.h>
#define max 100
struct customer
{
    int id, walletamt;
    char name[20];
};
int read (struct customer c[]);
void display(struct customer c[], int);
void sort(struct customer c[], int);
int search(struct customer c[], int, char s[]);
void update(struct customer c[], int);
int sumwalamt(struct customer c[], int);
void main()
{
    struct customer c[max];
    int n=0,res,ans,ch,sum;
    char s[50],r[50];
    do

```

```

{
printf("\n 1. READ\n 2. DISPLAY\n 3. SORT\n 4. SEARCH\n 5. UPDATE\n 6. SUM OF WALLET AMOUNT\n 7.
EXIT\n");
printf("\nEnter your choice : ");
scanf("%d",&ch);
switch(ch)
{
    case 1: printf("\nWe are in case 1");
        n=read(c);
        break;
    case 2: printf("\nWe are in case 2");
        display(c,n);
        break;
    case 3: printf("\nWe are in case 3");
        sort(c,n);
        break;
    case 4: printf("\nWe are in case 4");
        if(n==0)
printf("\n*****NO DETAIL FOR SEARCHING*****");
        else
        {
            printf("\nEnter customer to search : ");
            fflush(stdin);
            gets(s);
            res=search(c,n,s);
            if(res==0)
                printf("\nThe customer to search is not found");
            else
                printf("\nThe customer is found at location %d",res);
            }
        break;
    case 5: printf("\nWe are in case 5");
        if(n==0)
printf("\n*****NO DETAIL FOR UPDATION*****");
        else
        {
            printf("\nEnter customer to update : ");
            fflush(stdin);
            gets(r);
            ans=search(c,n,r);
            if(ans==0)
                printf("\nThe customer to update is not found");
            else
                update(c,ans-1);
            }
        break;
    case 6: printf("\nWe are in case 6");
        if(n==0)
printf("\n*****NO DETAIL FOUND FOR SUM*****");
}

```

```

        else
        {
            sum=sumwalamt(c,n);
            if(sum==0)
                printf("\nThe sum of the wallet amount is 0");
            else
                printf("\nThe sum of the wallet amount is %d",sum);
            }
            break;
        case 7: printf("\nExiting.....!");
            break;
        default : printf("\nWrong choice entered!");
    }
}while(ch!=7);

int read(struct customer c[])
{
    int i,n;
    printf("\nEnter total number of customers : ");
    scanf("%d",&n);
    printf("\n*****ENTER DETAILS*****\n");
    for(i=0;i<n;i++)
    {
        printf("\nRecord %d : \n",i+1);
        printf("ID : ");
        scanf("%d",&c[i].id);
        printf("NAME : ");
        fflush(stdin);
        gets(c[i].name);
        printf("WALLET AMOUNT : ");
        scanf("%d",&c[i].walletamt);
    }
    return n;
}

void display(struct customer c[], int n)
{
    int i;
    if(n==0)
        printf("\n*****NO DETAIL FOUND*****\n");
    else
    {
        printf("\n*****ENTERED DETAILS*****\n");
        for(i=0;i<n;i++)
        {
            printf("\nRecord %d : ",i+1);
            printf("\nID : ");
            printf("%d",c[i].id);
            printf("\nNAME : ");
            puts(c[i].name);
        }
    }
}

```

```

        printf("WALLET AMOUNT : ");
        printf("%d\n",c[i].walletamt);
    }
}
}

void sort(struct customer c[], int n)
{
    struct customer key;
    int ch,i,j;
    if(n==0)
        printf("\n*****NO DETAIL FOR SORTING*****");
    else
        do
    {
        printf("\n 1. Sort on Id \n 2. Sort on Name \n 3. Sort on Wallet amount\n");
        printf("\nEnter your choice : ");
        scanf("%d",&ch);
        switch(ch)
        {
            case 1:
                printf("\nWe are in case 1");
                for(i=1;i<n;i++)
                {
                    key=c[i];
                    for(j=i-1;(j>-1) && (c[j].id>key.id);j--)
                        c[j+1]=c[j];
                    c[j+1]=key;
                }
                break;
            case 2:
                printf("\nWe are in case 2");
                for(i=1;i<n;i++)
                {
                    key=c[i];
                    for(j=i-1;(j>-1) && (strcmp(c[j].name,key.name)>0);j--)
                        c[j+1]=c[j];
                    c[j+1]=key;
                }
                break;
            case 3:
                printf("\nWe are in case 3");
                for(i=1;i<n;i++)
                {
                    key=c[i];
                    for(j=i-1;(j>-1) && (c[j].walletamt>key.walletamt);j--)
                        c[j+1]=c[j];
                    c[j+1]=key;
                }
                break;
        }
    }
}

```

```

default:
    printf("\nWrong choice entered\n");
}
}while((ch!=1)&&(ch!=2)&&(ch!=3));
}

int search(struct customer c[], int n, char s[])
{
    int i;
    for(i=0;i<n;i++)
    {
        if(strcmp(c[i].name,s)==0)
            return (i+1);
    }
    return 0;
}

void update(struct customer c[], int ans)
{
    printf("\nOld Id : ");
    printf("%d",c[ans].id);
    printf("\nNew Id : ");
    scanf("%d",&c[ans].id);
    printf("\nOld Name : ");
    puts(c[ans].name);
    printf("New Name : ");
    fflush(stdin);
    gets(c[ans].name);
    printf("\nOld Wallet Amount : ");
    printf("%d",c[ans].walletamt);
    printf("\nNew Wallet Amount : ");
    scanf("%d",&c[ans].walletamt);
}

int sumwalamt(struct customer c[], int n)
{
    int count=0,i;
    for(i=0;i<n;i++)
        count=count+c[i].walletamt;
    return count;
}

```

**OUTPUT-**

```

E:\DS lab\DS LAB 2\structure.exe
1. READ
2. DISPLAY
3. SORT
4. SEARCH
5. UPDATE
6. SUM OF WALLET AMOUNT
7. EXIT
Enter your choice : 2
We are in case 2
*****NO DETAIL FOUND*****
1. READ
2. DISPLAY
3. SORT
4. SEARCH
5. UPDATE
6. SUM OF WALLET AMOUNT
7. EXIT
Enter your choice : 1
We are in case 1
Enter total number of customers : 3
*****ENTER DETAILS*****
Record 1 :
ID : 11
NAME : Deepali Singh
WALLET AMOUNT : 35000
Record 2 :
ID : 11
NAME : Anubhav Singh
WALLET AMOUNT : 15000
Record 3 :
ID : 22
NAME : Sudhanshu Singh
WALLET AMOUNT : 50000
1. READ
2. DISPLAY
3. SORT
4. SEARCH
5. UPDATE
6. SUM OF WALLET AMOUNT
7. EXIT
Enter your choice : 3
We are in case 3
1. READ
2. DISPLAY
3. SORT
4. SEARCH
5. UPDATE
6. SUM OF WALLET AMOUNT
7. EXIT
Enter your choice : 6
We are in case 6
The sum of the wallet amount is 100000
1. READ
2. DISPLAY
3. SORT
4. SEARCH
5. UPDATE
6. SUM OF WALLET AMOUNT
7. EXIT
Enter your choice : 2
We are in case 2
*****ENTERED DETAILS*****
Record 1 :
ID : 11
NAME : Anubhav Singh
WALLET AMOUNT : 15000
Record 2 :
ID : 11
NAME : Deepali Singh
WALLET AMOUNT : 35000
Record 3 :
ID : 22
NAME : Sudhanshu Singh
WALLET AMOUNT : 50000
1. READ
2. DISPLAY
3. SORT
4. SEARCH
5. UPDATE
6. SUM OF WALLET AMOUNT
7. EXIT
Enter your choice : 5
We are in case 1
1. READ
2. DISPLAY
3. SORT
4. SEARCH
5. UPDATE
6. SUM OF WALLET AMOUNT
7. EXIT
Enter your choice : 5
We are in case 5
Enter customer to update : Anubhav Singh
Old ID : 11
New ID : 12
Old Name : Anubhav Singh
New Name : Anubhav
Old Wallet Amount : 15000
New Wallet Amount : 35000
1. READ
2. DISPLAY
3. SORT
4. SEARCH
5. UPDATE
6. SUM OF WALLET AMOUNT
7. EXIT
Enter your choice : 2
We are in case 2
*****ENTERED DETAILS*****
Record 1 :
ID : 11
NAME : Anubhav
WALLET AMOUNT : 35000
Record 2 :
ID : 11
NAME : Deepali Singh
WALLET AMOUNT : 35000
Record 3 :
ID : 22
NAME : Sudhanshu Singh
WALLET AMOUNT : 50000
1. READ
2. DISPLAY
3. SORT
4. SEARCH
5. UPDATE
6. SUM OF WALLET AMOUNT
7. EXIT
Enter your choice : 7
Execution...
Process exited after 133.3 seconds with return value 2
Press any key to continue . . .

```

## PROGRAMS BASED ON POINTER

---

**Q10. WAP to read name in function and print in main.**

**Ans10.**

**C PROGRAM –**

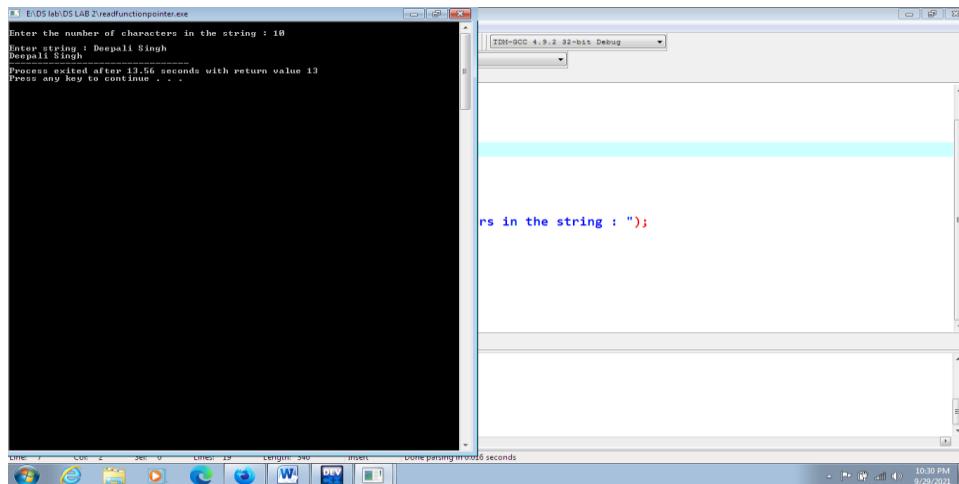
```
#include<stdio.h>
#include<stdlib.h>
char * Read();
```

```

void main()
{
    printf("%s",Read());
}
char * Read()
{
    char *p;
    int n;
    printf("\nEnter the number of characters in the string : ");
    scanf("%d",&n);
    fflush(stdin);
    p=(char *)malloc(sizeof(char)*n);
    printf("\nEnter string : ");
    gets(p);
    return p;
}

```

#### OUTPUT-



#### Q11. WAP of Fibonacci series using pointers.

Ans11.

#### C PROGRAM –

```

#include<stdio.h>
#include<malloc.h>
#include<stdlib.h>
int * Fibo(int n);
void main()
{
    int n,i;
    int *q;
    printf("\nEnter the number of elements : ");
    scanf("%d",&n);
    q=Fibo(n);
    printf("\nThe Fibonacci series is : \n");
    for(i=0;i<n;i++)
        printf("%d ",*(q+i));
}

```

```

int * Fibo(int n)
{
    int *p,i,A,B,C;
    p=(int *)malloc(sizeof(int)*n);
    for(i=0,A=0,B=1;i<n;i++)
    {
        *(p+i)=A;
        C=A+B;
        A=B;
        B=C;
    }
    return p;
}

```

### OUTPUT-

The screenshot shows a Windows desktop environment. On the left, a terminal window titled 'TDM-GCC 4.9.2 32-bit Debug' displays the following text:

```

E:\DS\Lab\DS LAB\2\fibpointer.c
Enter the number of elements : 10
The Fibonacci series is :
0 1 2 3 5 8 13 21 34
Process exited after 3.174 seconds with return value 10
Press any key to continue . . .

```

On the right, there is a code editor window showing the C code for the Fibo function. Below the desktop window, the taskbar shows various icons and the system tray indicates the date and time as 10:41 PM on 9/29/2021.

## PROGRAMS BASED ON STACK

---

### Q12. WAP to implement the basic operations of STACK.

Ans12.

#### C PROGRAM –

```

#include<stdio.h>
int top=-1;
#define max 20
int stk[max];
void init();
void push(int);
int pop();
void print();
int underflow();
int overflow();
int size();
int peek();
int main()

```

```

{
int ch,n,y;
do
{
printf("\n1.Push \n2.Pop \n3.Display \n4.Initialization \n5.Overflow \n6.Underflow \n7.Size \n8.Peek
\n9.Exit");
printf("\nEnter your choice : ");
scanf("%d",&ch);
switch(ch)
{
case 1:
    printf("\nEnter an element : ");
    scanf("%d",&n);
    push(n);
    break;
case 2:
    y=(pop());
    if(y!=-999)
        printf("\nThe deleted element is : %d",y);
    break;
case 3:
    print();
    break;
case 4:
    init();
    break;
case 5:
    if (overflow())
        printf("\nStack overflow");
    else
        printf("\nStack is not overflow");
    break;
case 6:
    if (underflow())
        printf("\nStack underflow");
    else
        printf("\nStack is not underflow");
    break;
case 7:
    printf("\nThe size of the stack is : %d",size());
    break;
case 8:
    if(peek())
        printf("\nThe current element in the stack is : %d",peek());
    else
        printf("\nThere is no element to display");
    break;
case 9:
    printf("\nExiting....");
}
}

```

```

        break;
    default:
        printf("\nWrong choice");
    }
}while(ch!=9);
return 0;
}
void init()
{
    top=-1;
}
void push(int x)
{
    if(overflow())
        printf("\nStack overflow");
    else
    {
        top++;
        stk[top]=x;
    }
}
int pop()
{
    int x;
    if(underflow())
    {
        printf("\nStack underflow");
        return -999;
    }
    else
    {
        x=stk[top];
        top--;
        return x;
    }
}
void print()
{
    int i;
    if(top===-1)
        printf("\nThere is no element to display");
    else
    {
        printf("\nThe elements of the stack are :\n");
        for(i=top;i>-1;i--)
        {
            printf("%d\t",stk[i]);
        }
    }
}

```

```

}

int overflow()
{
    if(top==max-1)
        return 1;
    else
        return 0;
}

int underflow()
{
    if(top==-1)
        return 1;
    else
        return 0;
}

int size()
{
    int y;
    y=top+1;
    return y;
}

int peek()
{
    if(top==-1)
        return 0;
    else
        return stk[top];
}

```

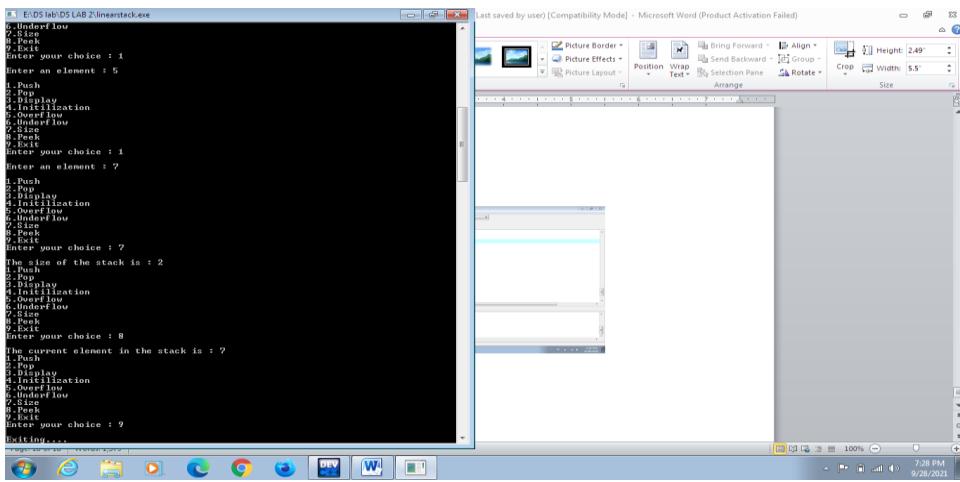
### OUTPUT-

The image shows a Windows desktop environment with two open windows. The left window is a command-line interface for a linear stack program. It displays a menu with options 1 through 9, followed by prompts for user input. The user has selected option 1 (push) multiple times, with the message 'The deleted element is : 5' appearing after each deletion. The right window is the IDM-GCC 4.9.2 32-bit Debug interface, showing assembly code and registers.

```

E:\DS lab\DS LAB 2\linearstack.exe
1.Push
2.Pop
3.Display
4.Initialization
5.Overflow
6.Underflow
7.Size
8.Peek
9.Exit
Enter your choice : 1
Enter an element : 5
1.Push
2.Pop
3.Display
4.Initialization
5.Overflow
6.Underflow
7.Size
8.Peek
9.Exit
Enter your choice : 2
The deleted element is : 5
1.Push
2.Pop
3.Display
4.Initialization
5.Overflow
6.Underflow
7.Size
8.Peek
9.Exit
Enter your choice : 5
Stack is not overflow
1.Push
2.Pop
3.Display
4.Initialization
5.Overflow
6.Underflow
7.Size
8.Peek
9.Exit
Enter your choice : 6
Stack underflow
1.Push
2.Pop
3.Display
4.Initialization
5.Overflow
6.Underflow
7.Size
8.Peek
9.Exit

```



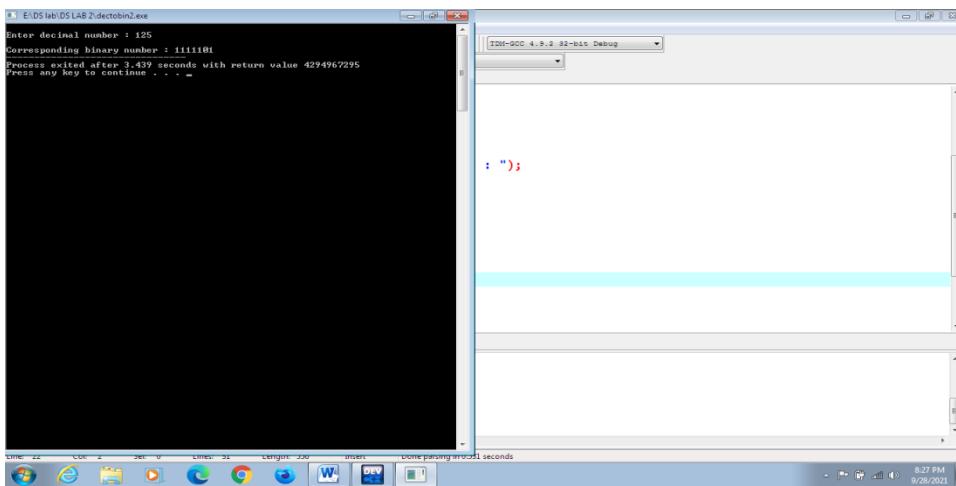
### Q13. WAP to convert a decimal number into binary using STACK.

Ans13.

#### C PROGRAM –

```
#include<stdio.h>
#define max 50
void push(int [],int *,int *);
int pop(int [],int *);
void main()
{
    int n,r;
    int stack[max];
    int top=-1;
    printf("\nEnter decimal number : ");
    scanf("%d",&n);
    while(n>0)
        push(stack,&top,&n);
    printf("\nCorresponding binary number : ");
    while(top>-1)
    {
        r=pop(stack,&top);
        printf("%d",r);
    }
}
void push(int stack[],int *top,int *a)
{
    stack[+(*top)]=*a%2;
    *a=*a/2;
}
int pop(int stack[],int *top)
{
    int x;
    x=stack[(*top)--];
    return x;
}
```

#### OUTPUT-



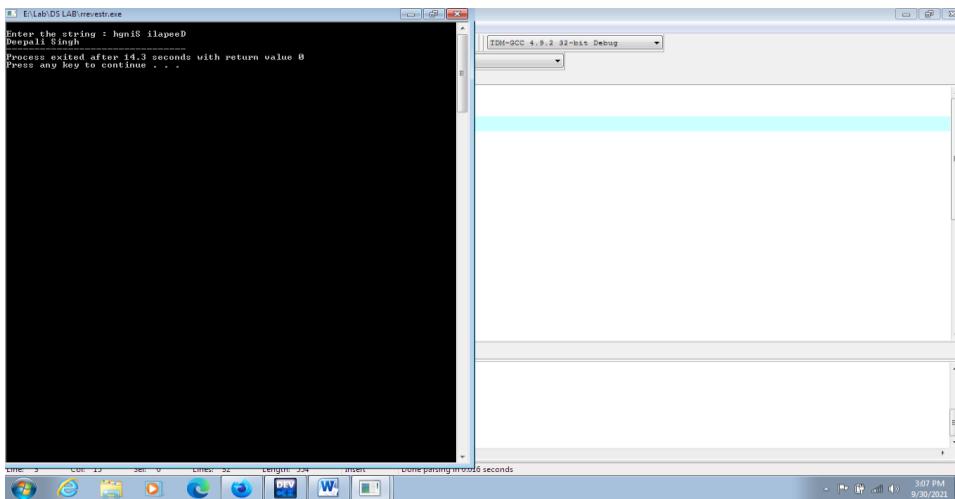
#### **Q14. WAP to reverse a string using STACK.**

Ans14.

#### **C PROGRAM –**

```
#include<stdio.h>
#include<string.h>
#define MAX 50
void push(char);
void pop();
int top;
char stack[MAX];
int main()
{
    char str[MAX];
    printf("\nEnter the string : ");
    gets(str);
    int len=strlen(str);
    int i;
    for(i=0;i<len;i++)
        push(str[i]);
    for(i=0;i<len;i++)
        pop();
    return 0;
}
void push(char x)
{
if(top==MAX-1)
    printf("\nStack overflow");
else
    stack[++top]=x;
}
void pop()
{
    printf("%c",stack[top--]);
}
```

#### **OUTPUT-**



### Q15. WAP to check whether the parentheses are balanced or not in a given expression using STACK.

Ans15.

#### C PROGRAM –

```
#include<stdio.h>
#include<ctype.h>
#define max 50
int stk[max];
int top=-1;
void push(char);
void pop();
void main()
{
    char exp[max];
    int i,opr=0,opd=0;
    char ch;
    printf("\nEnter the expression : ");
    gets(exp);
    for(i=0;exp[i]!='\0';i++)
    {
        ch=exp[i];
        printf("\nexp=%c",ch);
        if(ch=='(' || ch=='{' || ch=='[')
            push(ch);
        else if(isalpha(ch))
            opd++;
        else if(ch==')' || ch=='}' || ch==']')
        {
            if(top==-1)
                break;
            else
            {
                if(stk[top]=='(&&ch==')' || stk[top]=='['&&ch==']' || stk[top]=='{'&&ch=='}')
                    pop();
                else
                    break;
            }
        }
    }
}
```

```

    }
}
else
opr++;
}
printf("\nOperand = %d and operator = %d",opd,opr);
printf("\ntop=%d",top);
if((top== -1)&&(exp[i]=='0')&&(opd==opr+1))
printf("\nYour expression is valid...!");
else
printf("\nYour expression is not valid...!");
}
void push(char ch)
{
    stk[++top]=ch;
}
void pop()
{
    int x;
    x=stk[top--];
}

```

### OUTPUT-

The terminal window shows the following output:

```

Enter the expression : a+b-c*(d^e/f*g)
expr=a
expr=b
expr=c
expr=d
expr=e
expr=f
expr=g
expr=*
expr=/
expr=^
expr=-
expr=+
expr=()
expr=*
expr=/
expr=^
expr=-
expr=+
expr=()
Operand = 6 and operator = 5
Your expression is valid...!
Process exited after 25.32 seconds with return value 29
Press any key to continue . .

```

The IDE window shows the C source code with syntax highlighting. Red text highlights operators like '+', '\*', etc., and the closing brace '}' at the end of the if statement.

### Q16. WAP to convert an infix expression into postfix.

Ans16.

#### C PROGRAM –

```

#include<stdio.h>
#include<string.h>
#include<stdlib.h>
#define max 50
char stk[max];
int top=-1;
void push(char);
int pop();
int prec(char);
void main()

```

```

{
    char exp[max],post[max];
    int i,j=0,k;
    char ch;
    printf("\nEnter Infix expression : ");
    gets(exp);
    for(i=0;exp[i]!='\0';i++)
    {
        ch=exp[i];
        if(ch=='[' | ch=='{')
            ch='(';
        else if(ch==']' | ch=='}')
            ch=')';
        if(isalpha(ch))
            post[j++]=ch;
        else if(ch=='(')
            push(ch);
        else if(ch==')')
        {
            while(stk[top]!='(')
                post[j++]=pop();
            top--;
        }
        else
        {
            for(k=top;((k>-1) && (stk[k]!='(') && (prec(stk[top])>=prec(ch)));k--)
                post[j++]=pop();
            push(ch);
        }
    }
    while(top>-1)
        post[j++]=pop();
    post[j]='\0';
    printf("\nCorresponding postfix expression : ");
    puts(post);
}
void push(char ch)
{
    stk[++top]=ch;
}
int pop()
{
    int x;
    x=stk[top--];
    return x;
}
int prec(char x)
{
    if(x=='^')

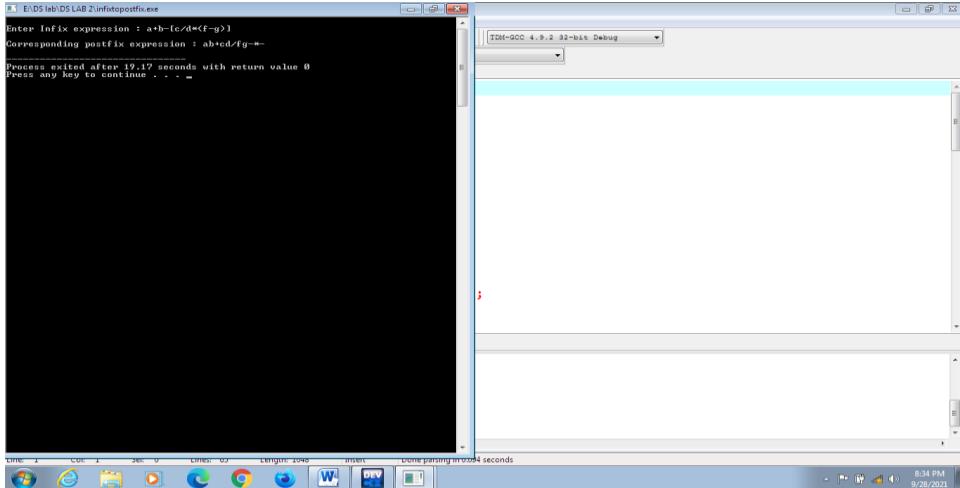
```

```

        return 9;
    else if(x=='*' || x=='/')
        return 7;
    else if(x=='+' || x=='-')
        return 5;
}

```

### OUTPUT-



### Q17. WAP to convert an infix expression into prefix.

Ans17.

#### C PROGRAM –

```

#include<stdio.h>
#include<string.h>
#include<stdlib.h>
#define max 50
char stk[max];
int top=-1;
void push(char);
int pop();
int prec(char);
void main()
{
    char exp[max],pre[max];
    int i,j=0,k;
    char ch;
    printf("\nEnter Infix expression : ");
    gets(exp);
    strrev(exp);
    for(i=0;exp[i]!='\0';i++)
    {
        ch=exp[i];
        if(ch=='[' || ch=='{')
            ch='(';
        else if(ch==']' || ch=='}')
            ch=')';
        if(isalpha(ch))

```

```

pre[j++]=ch;
else if(ch=='')
push(ch);
else if(ch=='(')
{
    while(stk[top]!=')')
    pre[j++]=pop();
    top--;
}
else
{
    for(k=top;((k>-1) && (stk[k]!='')) && (prec(stk[top])>prec(ch)));k--)
    pre[j++]=pop();
    push(ch);
}
}
while(top>-1)
pre[j++]=pop();
pre[j]='\0';
strrev(pre);
printf("\nCorresponding prefix expression : ");
puts(pre);
}

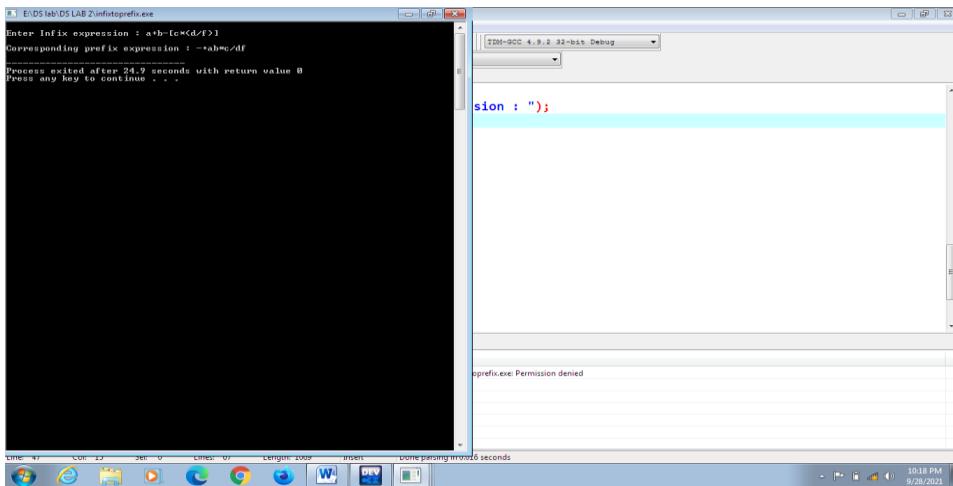
void push(char ch)
{
    stk[++top]=ch;
}

int pop()
{
    int x;
    x=stk[top--];
    return x;
}

int prec(char x)
{
    if(x=='^')
        return 9;
    else if(x=='*' | x=='/')
        return 7;
    else if(x=='+' | x=='-')
        return 5;
}

```

**OUTPUT-**



### Q18. WAP to evaluate a postfix expression.

Ans18.

#### C PROGRAM –

```
#include<stdio.h>
#include<ctype.h>
#include<math.h>
#define max 30
void push(double);
double pop();
double stk[max];
int top=-1;
void main()
{
    char post[max],ch;
    int i;
    double op1,op2,n;
    printf("\nEnter a valid postfix expression : ");
    gets(post);
    for(i=0;post[i]!='\0';i++)
    {
        ch=post[i];
        if(isalpha(ch))
        {
            printf("\nRead value of %c : ",ch);
            scanf("%lf",&n);
            push(n);
        }
        else
        {
            op1=pop();
            op2=pop();
            switch(ch)
            {
                case '+': push(op2+op1);
                break;
            }
        }
    }
}
```

```

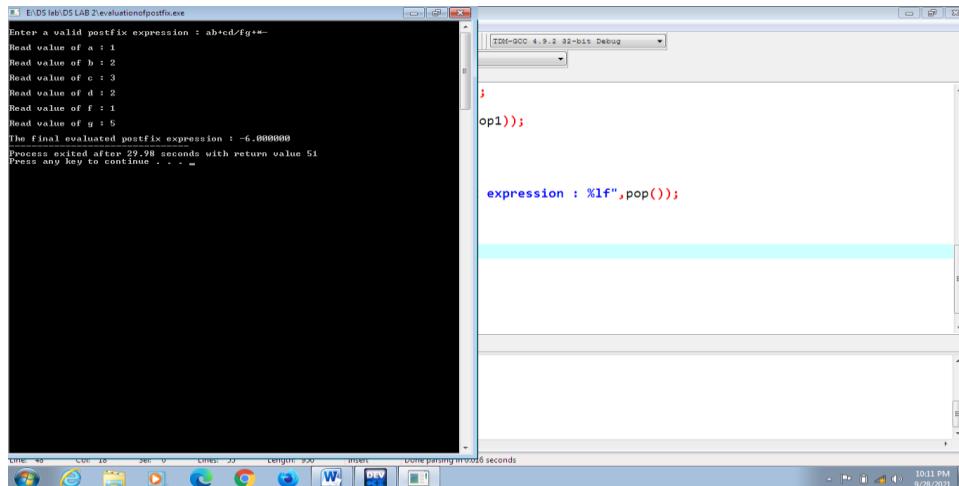
        case '-' : push(op2-op1);
                     break;
        case '*' : push(op2*op1);
                     break;
        case '/' : push(op2/op1);
                     break;
        case '^' : push(pow(op2,op1));
                     break;
    }
}
printf("\nThe final evaluated postfix expression : %lf",pop());
}

void push(double n)
{
    stk[++top]=n;
    printf("\n%lf",stk[top]);
}

double pop()
{
    double x;
    x=stk[top--];
    return x;
}

```

#### OUTPUT-



## PROGRAMS BASED ON QUEUE

---

### Q19. WAP to implement the basic operations of QUEUE.

Ans19.

#### C PROGRAM –

```
#include<stdio.h>
```

```

#define max 20
void init(int *, int *);
int underflow(int *, int *);
int underflow(int *, int *);
void display(int [],int *,int *);
int size(int *, int *);
int peek(int [],int *, int *);
void enqueue(int [],int *, int *, int);
int dequeue(int [],int *, int *);
void main()
{
    int ch,LQ[max],R=-1,F=-1,y,z,k;
    do
    {
        printf("\n1.Initialization \n2.Enqueue \n3.Dequeue \n4.Display");
        printf("\n5.Underflow \n6.Overflow \n7.Size \n8.Peek \n9.Exit");
        printf("\nEnter your choice : ");
        scanf("%d",&ch);
        switch(ch)
        {
            case 1: printf("\nWe are in case 1");
                init(&R,&F);
                break;
            case 2: printf("\nWe are in case 2");
                printf("\nEnter the element you want to insert : ");
                scanf("%d",&y);
                enqueue(LQ,&R,&F,y);
                break;
            case 3: printf("\nWe are in case 3");
                k=dequeue(LQ,&R,&F);
                if(k!=999)
                    printf("\nThe deleted element is %d",k);
                else
                    printf("\nThere is no element for deletion");
                break;
            case 4: printf("\nWe are in case 4");
                display(LQ,&R,&F);
                break;
            case 5: printf("\nWe are in case 5");
                if(underflow(&R,&F))
                    printf("\nThe linear queue is underflow");
                else
                    printf("\nThe linear queue is not underflow");
                break;
            case 6: printf("\nWe are in case 6");
                if(overflow(&R,&F))
                    printf("\nThe linear queue is overflow");
                else
                    printf("\nThe linear queue is not overflow");
        }
    }
}

```

```

        break;
    case 7: printf("\nWe are in case 7");
        if(size(&R,&F))
            printf("\nThe size of linear queue is %d",size(&R,&F));
        else
            printf("\nThe size of linear queue is 0");
        break;
    case 8: printf("\nWe are in case 8");
        if(peek(LQ,&R,&F)!=999)
            printf("\nThe peek of linear queue is %d",(peek(LQ,&R,&F)));
        else
            printf("\nThe linear queue is empty");
        break;
    case 9: printf("\nWe are in case 9");
        printf("\nExiting.....!");
        break;
    default:
        printf("\nWe are in default case");
    }
}while(ch!=9);
}

void init(int *R, int *F)
{
    *R=*F=-1;
}

int underflow(int *R, int *F)
{
    if(*F==-1)
        return 1;
    else
        return 0;
}

int overflow(int *R,int *F)
{
    if(*R==max-1)
        return 1;
    else
        return 0;
}

int size(int *R, int *F)
{
    if(*F==-1)
        return 0;
    else
        return (*R-*F+1);
}

int peek(int Q[], int *R, int *F)
{
    if(*F==-1)

```

```

        return 999;
    else
        return Q[*F];
}
void display(int Q[], int *R, int *F)
{
    int i;
    if(*F==-1)
        printf("\nThe linear queue is empty");
    else
    {
        printf("\nThe linear queue is : \n");
        for(i=*F;i<=*R;i++)
            printf("%d\t",Q[i]);
    }
}
void enqueue (int Q[],int *R,int *F,int x)
{
    if(*R==max-1)
        printf("\nThe linear queue is overflow");
    else
    {
        if(*R==-1)
            *F=*R=0;
        else
            (*R)++;
        Q[*R]=x;
    }
}
int dequeue(int Q[],int *R,int *F)
{
    int x;
    if(*F==-1)
        return 999;
    else
    {
        x=Q[*F];
        if(*F==*R)
            *F=*R=-1;
        else
            (*F)++;
        return x;
    }
}

```

**OUTPUT-**

```

E:\DS lab\DS LAB 2\linearqueue.exe
1.Initialization
2.Enqueue
3.Display
4.Underflow
5.Overflow
6.Size
7.Peek
8.Exit
Enter your choice : 2
We are in case 2
Enter the element you want to insert : 4
1.Initialization
2.Enqueue
3.Display
4.Underflow
5.Overflow
6.Size
7.Peek
8.Exit
Enter your choice : 2
We are in case 2
Enter the element you want to insert : 5
1.Initialization
2.Enqueue
3.Display
4.Underflow
5.Overflow
6.Size
7.Peek
8.Exit
Enter your choice : 2
We are in case 2
The linear queue is :
1 2 3 5
11:04 PM 9/28/2021

E:\DS lab\DS LAB 2\linearqueue.exe
1.Initialization
2.Enqueue
3.Display
4.Underflow
5.Overflow
6.Size
7.Peek
8.Exit
Enter your choice : 7
We are in case 7
The deleted element is 4
1.Initialization
2.Enqueue
3.Display
4.Underflow
5.Overflow
6.Size
7.Peek
8.Exit
Enter your choice : 6
We are in case 6
The linear queue is not underflow
1.Initialization
2.Enqueue
3.Display
4.Underflow
5.Overflow
6.Size
7.Peek
8.Exit
Enter your choice : 5
We are in case 5
The linear queue is not overflow
1.Initialization
2.Enqueue
3.Display
4.Underflow
5.Overflow
6.Size
7.Peek
8.Exit
Enter your choice : 6
We are in case 6
The linear queue is not overflow
1.Initialization
2.Enqueue
3.Display
4.Underflow
5.Overflow
6.Size
7.Peek
8.Exit
Enter your choice : 3
We are in case 3
The deleted element is 4
1.Initialization
2.Enqueue
3.Display
4.Underflow
5.Overflow
6.Size
7.Peek
8.Exit
Enter your choice : 9
We are in case 9
Exiting.....
Process exited after 56.24 seconds with return value 9
Press any key to continue . . .

```

## Q20. WAP to implement the basic operations of CIRCULAR QUEUE.

Ans20.

### C PROGRAM –

```
#include <stdio.h>
#define SIZE 5
int cq[SIZE];
int f=-1, r=-1;
void init();
int size();
int peek();
int isFull();
```

```

int isEmpty();
void enqueue(int) ;
int dequeue();
void display();
void main()
{
    int ch,y,z,k;
    do
    {
        printf("\n 1.Initialization \n 2.Enqueue \n 3.Dequeue \n 4.Display");
        printf("\n 5.Isfull \n 6.Isempy \n 7.Size \n 8.Peek \n 9.Exit");
        printf("\n Enter your choice : ");
        scanf("%d",&ch);
        switch(ch)
        {
            case 1: printf("\n We are in case 1");
                init();
                break;
            case 2: printf("\n We are in case 2");
                printf("\n Enter the element you want to insert : ");
                scanf("%d",&y);
                enqueue(y);
                break;
            case 3: printf("\n We are in case 3");
                k=dequeue();
                if(k!=999)
                    printf("\n The deleted element is %d",k);
                else
                    printf("\n There is no element for deletion");
                break;
            case 4: printf("\n We are in case 4");
                display();
                break;
            case 5: printf("\n We are in case 5");
                if(isFull())
                    printf("\n The circular queue is full");
                else
                    printf("\n The circular queue is not full");
                break;
            case 6: printf("\n We are in case 6");
                if(isEmpty())
                    printf("\n The circular queue is empty");
                else
                    printf("\n The circular queue is not empty");
                break;
            case 7: printf("\n We are in case 7");
                if(size())
                    printf("\n The size of circular queue is %d",size());
                else

```

```

        printf("\n The size of circular queue is 0");
        break;
    case 8: printf("\n We are in case 8");
        if(peek()!=999)
            printf("\n The peek of circular queue is %d",(peek()));
        else
            printf("\n The circular queue is empty");
        break;
    case 9: printf("\n We are in case 9");
        printf("\n Exiting.....!");
        break;
    default:
        printf("\n We are in default case");
    }
}while(ch!=9);
}
void init()
{
    r=f=-1;
}
int size()
{
    if(f==-1)
        return 0;
    else if(r>=f)
        return (r-f+1);
    else
        return (SIZE-f+r+1);
}
int peek()
{
    if(f==-1)
        return 999;
    else
        return cq[f];
}
int isFull()
{
    if((f==r+1) || (f==0 && r==SIZE-1))
        return 1;
    else
        return 0;
}
int isEmpty()
{
    if (f==-1)
        return 1;
    else
        return 0;
}

```

```

}

void enqueue(int m)
{
    if (isFull())
        printf("\n Queue is full!! \n");
    else
    {
        if (f==-1)
            f=0;
        r=(r+1)%SIZE;
        cq[r]=m;
    }
}

int dequeue()
{
    int m;
    if (isEmpty())
        return 999;
    else
    {
        m=cq[f];
        if(f==r)
        {
            f=-1;
            r=-1;
        }
        else
        {
            f=(f+1)%SIZE;
        }
        return(m);
    }
}

void display()
{
    int i;
    if (isEmpty())
        printf(" \n The circular queue is empty \n");
    else
    {
        printf("\n Front -> %d ", f);
        printf("\n Circular queue -> ");
        for(i=f; i!=r; i=(i+1)%SIZE)
        {
            printf("%d ", cq[i]);
        }
        printf("%d ", cq[i]);
        printf("\n Rear -> %d \n", r);
    }
}

```

}

## OUTPUT-

```
1. Initialization
2. Enqueue
3. Dequeue
4. Display
5. IsFull
6. IsEmpty
7. Size
8. Peek
9. Exit
Enter your choice : 2
We are in case 2
Enter the element you want to insert : 3
1. Initialization
2. Enqueue
3. Dequeue
4. Display
5. IsFull
6. IsEmpty
7. Size
8. Peek
9. Exit
Enter your choice : 2
We are in case 2
Enter the element you want to insert : 4
1. Initialization
2. Enqueue
3. Dequeue
4. Display
5. IsFull
6. IsEmpty
7. Size
8. Peek
9. Exit
Enter your choice : 4
We are in case 4
Front -> 3
Queue -> 3 4
Rear -> 1
1. Initialization
2. Enqueue
3. Dequeue
4. Display
5. IsFull
6. IsEmpty
7. Size
8. Peek
9. Exit
Enter your choice : 7
We are in case 7
1. Initialization
2. Enqueue
3. Dequeue
4. Display
5. IsFull
6. IsEmpty
7. Size
8. Peek
9. Exit
Enter your choice : 8
We are in case 8
The size of circular queue is 2
1. Initialization
2. Enqueue
3. Dequeue
4. Display
5. IsFull
6. IsEmpty
7. Size
8. Peek
9. Exit
Enter your choice : 3
We are in case 3
The front element is 3
1. Initialization
2. Enqueue
3. Dequeue
4. Display
5. IsFull
6. IsEmpty
7. Size
8. Peek
9. Exit
Enter your choice : 5
We are in case 5
The circular queue is not full
1. Initialization
2. Enqueue
3. Dequeue
4. Display
5. IsFull
6. IsEmpty
7. Size
8. Peek
9. Exit
Enter your choice : 9
We are in case 9
Exiting.....
Process exited after 79.02 seconds with return value 9
Press any key to continue . . .

```

## Q21. WAP to implement the basic operations of a PRIORITY QUEUE.

Ans21.

### C PROGRAM –

```
#include<stdio.h>
#define N 5
int Q[N],P[N];
int r=-1,f=-1;
void init();
void display();
void enqueue(int,int);
void dequeue();
int isempty();
int isfull();
int size();
int peek();
int main()
{
    int ch,data,p,i,n;
    do
    {
        printf("\n 1. Initializtion \n 2. Enqueue \n 3. Display \n 4. Dequeue");

```

```

printf("\n 5. Size \n 6. Peek \n 7. Isempty \n 8. Isfull \n 9. Exit");
printf("\n Enter Your Choice : ");
scanf("%d",&ch);
switch(ch)
{
    case 1:
        init();
        break;
    case 2:
        printf("\n Enter the number of data : ");
        scanf("%d",&n);
        printf("\n Enter your data and priority of data : ");
        i=0;
        while(i<n)
        {
            printf("\n Insert %d : ",i+1);
            scanf("%d %d",&data,&p);
            enqueue(data,p);
            i++;
        }
        break;
    case 3:
        display();
        break;
    case 4:
        dequeue();
        break;
    case 5:
        if(size())
            printf("\nThe size of priority queue is %d",size());
        else
            printf("\nThe size of priority queue is 0");
        break;
    case 6:
        if(peek()!=999)
            printf("\nThe peek of priority queue is %d",peek());
        else
            printf("\nThe priority queue is empty");
        break;
    case 7:
        if(isempty())
            printf("\nThe priority queue is empty");
        else
            printf("\nThe priority queue is not empty");
        break;
    case 8:
        if(isfull())
            printf("\nThe priority queue is full");
        else

```

```

        printf("\nThe priority queue is not full");
        break;
    case 9:
        printf("\nWe are exiting...!");
        break;
    default:
        printf("\nIncorrect Choice");
    }
}while(ch!=9);
return 0;
}
void init()
{
    r=f=-1;
}
int isempty()
{
    if(f==-1)
    return 1;
    else
    return 0;
}
int isfull()
{
    if(r==N-1)
    return 1;
    else
    return 0;
}
int size()
{
    if(f==-1)
    return 0;
    else
    return (r-f+1);
}
int peek()
{
    if(f==-1)
    return 999;
    else
    return Q[f];
}
void enqueue(int d,int p)
{
    int i;
    if((f==0)&&(r==N-1))
    printf("\nQueue is full");
    else

```

```

{
    if(f== -1)
    {
        f=r=0;
        Q[r]=d;
        P[r]=p;
    }
    else if(r==N-1)
    {
        for(i=f;i<=r;i++)
        {
            Q[i-f]=Q[i];
            P[i-f]=P[i];
            r=r-f;
            f=0;
            for(i=r;i>f;i--)
            {
                if(p>P[i])
                {
                    Q[i+1]=Q[i];
                    P[i+1]=P[i];
                }
                else
                break;
                Q[i+1]=d;
                P[i+1]=p;
                r++;
            }
        }
    }
    else
    {
        for(i=r;i>=f;i--)
        {
            if(p>P[i])
            {
                Q[i+1]=Q[i];
                P[i+1]=P[i];
            }
            else
            break;
        }
        Q[i+1]=d;
        P[i+1]=p;
        r++;
    }
}
void display()

```

```

{
int i;
if(f==-1)
    printf("\nQueue is Empty");
else
    for(i=f;i<=r;i++)
        printf("\nElement = %d\tPriority = %d",Q[i],P[i]);
}
void dequeue()
{
    if(f==-1)
        printf("\nQueue is Empty");
    else
    {
        printf("\nDeleted Element = %d\tPriority = %d",Q[f],P[f]);
        if(f==r)
            f=r=-1;
        else
            f++;
    }
}

```

## OUTPUT-

```

E:\Lab\OS LAB>pq.cxx
1. Initialization
2. Enqueue
3. Display
4. Dequeue
5. Size
6. Peek
7. IsEmpty
8. IsFull
9. Exit
Enter Your Choice : 2
Enter the number of data : 3
Enter your data and priority of data :
Insert 1 : 1 3
Insert 2 : 2 1
Insert 3 : 3 2
1. Initialization
2. Enqueue
3. Display
4. Dequeue
5. Size
6. Peek
7. IsEmpty
8. IsFull
9. Exit
Enter Your Choice : 3
Element = 2 Priority = 2
Element = 1 Priority = 1
1. Initialization
2. Enqueue
3. Display
4. Dequeue
5. Size
6. Peek
7. IsEmpty
8. IsFull
9. Exit
Enter Your Choice : 4
Deleted Element = 1 Priority = 3
1. Initialization
2. Enqueue
3. Display
4. Dequeue
5. Size
6. Peek
7. IsEmpty
8. IsFull
9. Exit
Enter Your Choice : 5

```

```

E:\Lab\OS LAB>pq.cxx
Enter Your Choice : 5
The peek of priority queue is 2
1. Initialization
2. Enqueue
3. Display
4. Dequeue
5. Size
6. Peek
7. IsEmpty
8. IsFull
9. Exit
Enter Your Choice : 6
The peek of priority queue is 3
1. Initialization
2. Enqueue
3. Display
4. Dequeue
5. Size
6. Peek
7. IsEmpty
8. IsFull
9. Exit
Enter Your Choice : 9
We are exiting...
Process exited after 66.06 seconds with return value 0
Press any key to continue . . .

```

## PROGRAMS BASED ON SEARCHING

---

**Q22. WAP for linear search.**

Ans22.

**C PROGRAM –**

```
#include<stdio.h>
int linsearch(int [],int,int);
void print(int [],int);
int main()
{
    int a[100],i,j,n,x,r;
    printf("\nEnter the number of elements in array : ");
    scanf("%d",&n);
    printf("\nEnter array : \n");
    for(i=0;i<n;i++)
        scanf("%d",&a[i]);
    printf("\nThe original array is : \n");
    print(a,n);
    printf("\nRead element to search : ");
    scanf("%d",&x);
    r=linsearch(a,n,x);
    if(r!=-999)
        printf("\n%d found %d times",x,r);
    else
        printf("\n%d not found",x);
    return 0;
}
int linsearch(int a[], int n, int x)
{
    int i,c=0;
    for(i=0;i<n;i++)
    {
        if(a[i]==x)
        {
            printf("\n%d found at location %d",x,i+1);
            c++;
        }
        else
            continue;
    }
    if(c==0)
        return -999;
    else
        return c;
}
void print(int a[],int n)
{
    int i;
```

```

        for(i=0;i<n;i++)
            printf("%d\t",a[i]);
    }
}

```

### OUTPUT-

```

E:\DS lab\DS LAB 2\linsearch.exe
Enter array :
2 3 1 5 6 7 1
The original array is : 2 3 1 5 6 7 1
Read element to search : 1
1 found at location 1
1 found at location 5
1 found at location 10
1 found at location 14
Process completed after 14.28 seconds with return value 0
Press any key to continue . .

```

### Q23. WAP for binary search.

Ans23.

#### C PROGRAM –

```

#include<stdio.h>
int binsearch(int [],int,int);
void print(int [],int);
int main()
{
    int a[100],i,j,n,x,r;
    printf("\nEnter the number of elements in array : ");
    scanf("%d",&n);
    printf("\nEnter array : \n");
    for(i=0;i<n;i++)
        scanf("%d",&a[i]);
    printf("\nThe original array is : \n");
    print(a,n);
    printf("\nRead element to search : ");
    scanf("%d",&x);
    r=binsearch(a,n,x);
    if(r!=999)
        printf("\n%d found at location %d",x,r+1);
    else
        printf("\n%d not found",x);
    return 0;
}
int binsearch(int a[], int n, int x)
{
    int l=0,h=n-1,m;
    do
    {
        m=(l+h)/2;

```

```

if(x<a[m])
    h=m-1;
else if(x>a[m])
    l=m+1;
else
    return m;
}while(l<=h);
return -999;
}

void print(int a[],int n)
{
int i;
for(i=0;i<n;i++)
{
    printf("%d\t",a[i]);
}
}

```

#### OUTPUT-

```

E:\DS lab\DS LAB 2\bimsearch.exe
Enter the number of elements in array : 10
Enter array :
5
4
6
-2
6
2
8
9
2
The original array is :
5      4      6      -2      6      2      8      9      2
Read element to search : -2
-2 found at location 5
Process exited after 14.49 seconds with return value 0
Press any key to continue . .

```

## PROGRAMS BASED ON SORTING

---

#### Q24. WAP for Bubble sort.

Ans24.

#### C PROGRAM –

```

#include<stdio.h>
void bsort(int [],int);
void print(int [],int);
int main()
{
    int a[100],i,j,n,x,r;
    printf("\nEnter the number of elements in array : ");
    scanf("%d",&n);
    printf("\nEnter array : \n");

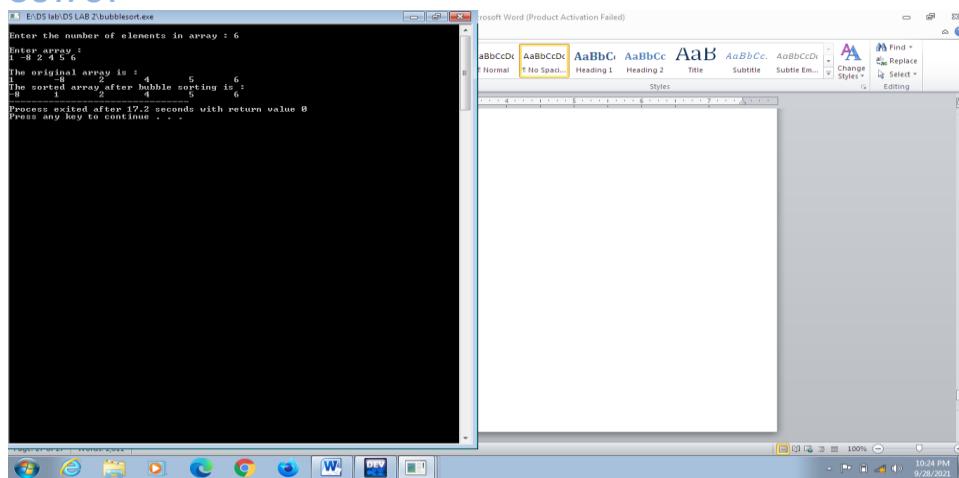
```

```

for(i=0;i<n;i++)
    scanf("%d",&a[i]);
printf("\nThe original array is : \n");
print(a,n);
bsort(a,n);
printf("\nThe sorted array after bubble sorting is : \n");
print(a,n);
return 0;
}
void bsort(int a[], int n)
{
    int i,j,t;
    for(i=1;i<n;i++)
        for(j=0;j<n-i;j++)
        {
            if(a[j]>a[j+1])
            {
                t=a[j];
                a[j]=a[j+1];
                a[j+1]=t;
            }
        }
}
void print(int a[],int n)
{
    int i;
    for(i=0;i<n;i++)
    {
        printf("%d\t",a[i]);
    }
}

```

### OUTPUT-



### Q25. WAP for Selection sort.

Ans25.

#### C PROGRAM –

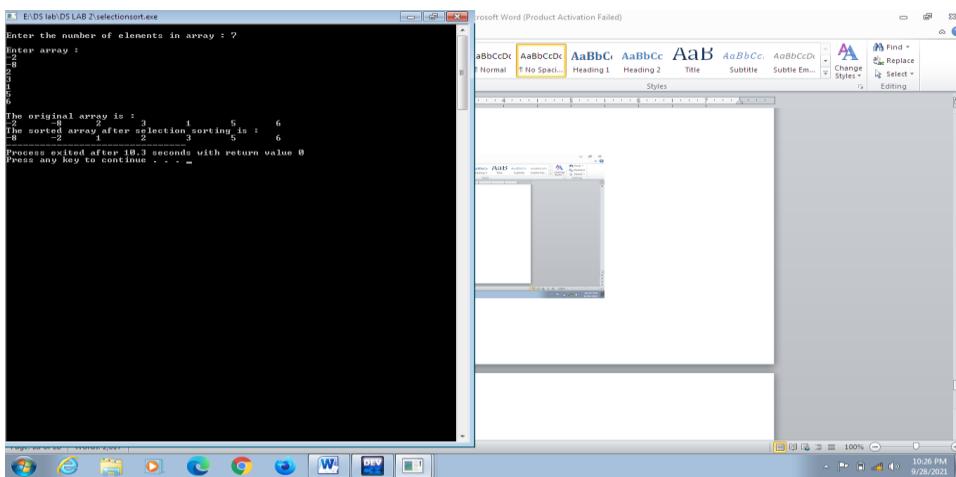
```
#include<stdio.h>
```

```

void ssort(int [],int);
int main()
{
    int a[10],i,j,n;
    printf("\nEnter the number of elements in array : ");
    scanf("%d",&n);
    printf("\nEnter array : \n");
    for(i=0;i<n;i++)
    {
        scanf("%d",&a[i]);
    }
    printf("\nThe original array is : \n");
    for(i=0;i<n;i++)
    {
        printf("%d\t",a[i]);
    }
    ssort(a,n);
    printf("\nThe sorted array after selection sorting is : \n");
    for(i=0;i<n;i++)
    {
        printf("%d\t",a[i]);
    }
    return 0;
}
void ssort(int a[],int n)
{
    int i,j,min,t;
    for(i=0;i<n-1;i++)
    {
        min=i;
        for(j=i+1;j<n;j++)
        {
            if(a[min]>a[j])
                min=j;
        }
        t=a[i];
        a[i]=a[min];
        a[min]=t;
    }
}

```

**OUTPUT-**



## Q26. WAP for Insertion sort.

Ans26.

### C PROGRAM –

```
#include<stdio.h>
void isort(int [],int);
void print(int [],int);
int main()
{
    int a[10],i,j,n;
    printf("\nEnter the number of elements in array : ");
    scanf("%d",&n);
    printf("\nEnter array : \n");
    for(i=0;i<n;i++)
    {
        scanf("%d",&a[i]);
    }
    printf("\nThe original array is : \n");
    for(i=0;i<n;i++)
    {
        printf("%d\t",a[i]);
    }
    isort(a,n);
    print(a,n);
    return 0;
}
void isort(int a[],int n)
{
    int i,j,key;
    for(i=1;i<n;i++)
    {
        key=a[i];
        for(j=i-1;(j>-1)&&(a[j]>key);j--)
            a[j+1]=a[j];
        a[j+1]=key;
    }
}
```

```

void print(int a[],int n)
{
    int i;
    printf("\nThe sorted array after insertion sorting is : \n");
    for(i=0;i<n;i++)
    {
        printf("%d\t",a[i]);
    }
}

```

### OUTPUT-

```

E:\DS\Lab\DS LAB 2\insertionsort.exe
Enter the number of elements in array : 10
Enter array :
2
4
5
7
8
9
0
-1
4
7
the original array is : 2 4 5 7 8 9 0 -1 4 7
The sorted array after insertion sorting is : 1 2 4 5 7 8 9
Process exited after 16.93 seconds with return value 0
Press any key to continue . . .

```

### Q27. WAP for Quick sort.

Ans27.

#### C PROGRAM –

```

#include<stdio.h>
void quicksort(int [], int, int);
int partition(int [], int, int);
void swap(int *, int *);
void read(int[],int);
void print(int[],int);
void main()
{
    int n;
    int a[50];
    printf("\nEnter number of elements in the array : ");
    scanf("%d",&n);
    printf("\nEnter the elements : \n");
    read(a,n);
    printf("\nThe unsorted array : \n");
    print(a,n);
    quicksort(a,0,n-1);
    printf("\nThe sorted array : \n");
    print(a,n);
}
void quicksort(int b[], int low, int high)
{

```

```

int pi,n=7;
if(low<high)
{
    pi=partition(b,low,high);
    quicksort(b,low,pi-1);
    quicksort(b,pi+1,high);
}
int partition(int c[], int low, int high)
{
    int pivot,i,j;
    pivot=c[high];
    i=low-1;
    for(j=low;j<high;j++)
    {
        if(c[j]<pivot)
        {
            i++;
            if(i!=j)
                swap(&c[i],&c[j]);
        }
    }
    swap(&c[i+1],&c[high]);
    return (i+1);
}
void swap(int *a, int *b)
{
    int temp;
    temp=*a;
    *a=*b;
    *b=temp;
}
void read(int d[],int n)
{
    int i;
    for(i=0;i<n;i++)
        scanf("%d",&d[i]);
}
void print(int d[],int n)
{
    int i;
    for(i=0;i<n;i++)
        printf("%d\t",d[i]);
    printf("\n");
}

```

**OUTPUT-**

The screenshot shows a Windows desktop environment. In the foreground, a terminal window titled 'E:\DS lab\DS LAB 2\quicksort.exe' displays the output of a quicksort program. The program prompts for the number of elements, takes input, prints the original array, sorts it, and prints the sorted array. It also shows execution time and a prompt for continuation. In the background, an IDE window titled 'IDM-GCC 4.9.2 32-bit Debug' is visible, though its content is mostly blank.

```

E:\DS lab\DS LAB 2\quicksort.exe
Enter number of elements in the array : 10
Enter the elements :
22
33
44
55
66
11
44
222
33
22
56
The unsorted array :
22      33      44      55      66     -134     222      0      22      56
The sorted array :
0      22      22      33      44      55      56      66     222
Process exited after 18.86 seconds with return value 10
Press any key to continue . .

```

## PROGRAMS BASED ON LINKED LIST

---

**Q28. WAP for implementing Linked List with all basic operations - All insertions Deletions etc.**

Ans28.

**C PROGRAM –**

```

#include<stdio.h>
#include<malloc.h>
struct node
{
    int data;
    struct node *next;
};
typedef struct node Node;
Node *start=NULL;
void init();
Node *createnode(int);
void insertf(int);
void insertl(int);
void insertr(int,int);
void insertm(int);
void display();
int deletef();
int deletel();
int deleter(int);
int deletem();
void delspdata(int);
int size();
void lsearch(int);
void main()
{
    int ch,a,loc,key,u;
    do
    {

```

```

    printf("\n 1. Initialization \n 2. Insert First \n 3. Insert Random \n 4. Insert Middle \n 5. Insert
Last \n 6. Display");
    printf("\n 7. Delete First \n 8. Delete Random \n 9. Delete Middle \n 10. Delete Last \n 11. Delete
specific data");
    printf("\n 12. Size \n 13. Exit");
    printf("\n Enter your choice : ");
    scanf("%d",&ch);
    switch(ch)
    {
        case 1:
            init();
            break;
        case 2:
            printf("\n Enter element to insert : ");
            scanf("%d",&a);
            insertf(a);
            break;
        case 3:
            printf("\n Enter element to insert : ");
            scanf("%d",&a);
            printf("\n Enter position to insert : ");
            scanf("%d",&loc);
            insertr(a,loc);
            break;
        case 4:
            printf("\n Enter element to insert : ");
            scanf("%d",&a);
            insertm(a);
            break;
        case 5:
            printf("\n Enter element to insert : ");
            scanf("%d",&a);
            insertl(a);
            break;
        case 6:
            display();
            break;
        case 7:
            u=deletef();
            if(u==-999)
                printf("\n Linked list is empty");
            else
                printf("\n Deleted element : %d",u);
            break;
        case 8:
            printf("\n Enter position to delete : ");
            scanf("%d",&loc);
            u=deleter(loc);
            if(u==-999)

```

```

printf("\n Linked list is empty");
else if(u== -9999)
printf("\n Deletion not possible");
else
printf("\n Deleted element : %d",u);
break;
case 9:
u=deletem();
if(u== -999)
printf("\n Linked list is empty");
else
printf("\n Deleted element : %d",u);
break;
case 10:
u=deletel();
if(u== -999)
printf("\n Linked list is empty");
else
printf("\n Deleted element : %d",u);
break;
case 11:
printf("\n Enter data to be deleted : ");
scanf("%d",&a);
delspdata(a);
break;
case 12:
if(size())
printf("\n Size of Linked list : %d",size());
else
printf("\n Size of Linked list : 0");
break;
case 13:
printf("\n We are exiting..Good Bye..!");
break;
default:
printf("\n Wrong choice entered..!");
}
}while(ch!=13);
}
void init()
{
start=NULL;
}
Node * createnode(int x)
{
Node *temp;
temp=(Node *)malloc(sizeof(Node));
temp->data=x;
temp->next=NULL;

```

```

        return temp;
    }
void insertf(int x)
{
    Node *temp;
    temp=createnode(x);
    if(start==NULL)
        start=temp;
    else
    {
        temp->next=start;
        start=temp;
    }
}
void insertl(int x)
{
    Node *temp,*p;
    temp=createnode(x);
    if(start==NULL)
        start=temp;
    else
    {
        for(p=start;p->next!=NULL;p=p->next);
        p->next=temp;
    }
}
void insertr(int x,int pos)
{
    Node *temp,*p;
    int i,l=size();
    if(start==NULL)
        printf("\n Linked list is empty");
    else if(pos<1 || pos>(l+1))
    {
        printf("\n Insertion not possible");
        return;
    }
    else if(pos==1)
    {
        insertf(x);
        return;
    }
    else if(pos==(l+1))
    {
        insertl(x);
        return;
    }
    else
    {

```

```

        temp=createnode(x);
        for(p=start,i=1;i<pos-1;i++,p=p->next);
        temp->next=p->next;
        p->next=temp;
        return;
    }
}

void insertm(int x)
{
    Node *temp,*p;
    int i;
    if(start==NULL)
    {
        temp=createnode(x);
        start=temp;
    }
    else
    {
        temp=createnode(x);
        for(p=start,i=1;i<(size()/2);i++,p=p->next);
        temp->next=p->next;
        p->next=temp;
    }
}
int size()
{
    Node *p;
    int c;
    if(start==NULL)
        return 0;
    else
    {
        for(p=start,c=0;p!=NULL;c++,p=p->next);
    }
    return c;
}
int deletef()
{
    Node *p;
    int x;
    if(start==NULL)
        return -999;
    else
    {
        p=start;
        start=p->next;
        x=p->data;
        free(p);
        return x;
    }
}

```

```

    }
}

int deletel()
{
    Node *p,*t;
    int i,x;
    if(start==NULL)
        return -999;
    else if(size()==1)
    {
        deletef();
        return;
    }
    else
        for(p=start,i=1;i<(size()-1);i++,p=p->next);
        t=p->next;
        p->next=NULL;
        x=t->data;
        free(t);
        return x;
}
int deleter(int pos)
{
    Node *p,*t;
    int l=size(),x,i;
    if(start==NULL)
        return -999;
    else
    {
        if(pos<1 || pos>l)
            return -9999;
        else if(pos==1)
        {
            deletef();
            return;
        }
        else if(pos==l)
        {
            deletel();
            return;
        }
        else
        {
            for(p=start,i=1;i<(pos-1);i++,p=p->next);
            t=p->next;
            p->next=t->next;
            x=t->data;
            free(t);
            return x;
        }
    }
}

```

```

        }
    }
}

int deletem()
{
    Node *p,*t;
    int x,i;
    if(start==NULL)
        return -999;
    else if(size()==1)
    {
        t=start;
        start=NULL;
        x=t->data;
        free(t);
        return x;
    }
    else
    {
        for(p=start,i=1;i<(size()/2);i++,p=p->next);
        t=p->next;
        p->next=t->next;
        x=t->data;
        free(t);
        return x;
    }
}
void delspdata(int x)
{
    Node *p=start,*t;
    int flag=0;
    if(start==NULL)
    {
        printf("\n Linked list is empty");
        return;
    }
    else
    {
        while(p!=NULL)
        {
            if(p==start)
            {
                if(start->data==x)
                {
                    flag++;
                    start=start->next;
                    p=start;
                }
            }
        }
    }
}

```

```

        {
            t=p;
            p=p->next;
        }
    }
    else if(p->data==x)
    {
        flag++;
        t->next=p->next;
        p=t->next;
    }
    else
    {
        t=p;
        p=p->next;
    }
}
if(flag==0)
    printf("\n Element not found");
}
void display()
{
    Node *p,*t;
    if(start==NULL)
    {
        printf("\n Linked list is empty");
    }
    else
    {
        printf("\n Linked list : \n");
        for(p=start;p!=NULL;p=p->next)
        printf("%d\t",p->data);
    }
}
}

```

### OUTPUT-

```

$ EDS LAB 01
1. Initialization
2. Insert First
3. Insert Random
4. Insert Middle
5. Insert Last
6. Display
7. Delete First
8. Delete Random
9. Delete Middle
10. Delete Last
11. Delete specific data
12. First
13. Exit
Enter your choice : 1
1. Initialization
2. Insert First
3. Insert Random
4. Insert Middle
5. Insert Last
6. Display
7. Delete First
8. Delete Random
9. Delete Middle
10. Delete Last
11. Delete specific data
12. First
13. Exit
Enter your choice : 2
Enter element to insert : 3
1. Initialization
2. Insert First
3. Insert Random
4. Insert Middle
5. Insert Last
6. Display
7. Delete First
8. Delete Random
9. Delete Middle
10. Delete Last
11. Delete specific data
12. First
13. Exit
Enter your choice : 5
Enter element to insert : 2

```

```

# E:\C\LAB\Review
Enter element to insert : 2
1. Initialization
2. Insert First
3. Insert Random
4. Insert Middle
5. Insert Last
6. Display
7. Delete First
8. Delete Random
9. Delete Middle
10. Delete Last
11. Delete specific data
12. Size
13. Exit
Enter your choice : 6
Linked list :
1. Initialization
2. Insert First
3. Insert Random
4. Insert Middle
5. Insert Last
6. Display
7. Delete First
8. Delete Random
9. Delete Middle
10. Delete Last
11. Delete specific data
12. Size
13. Exit
Enter your choice : 12
Size of Linked list : 2
1. Initialization
2. Insert First
3. Insert Random
4. Insert Middle
5. Insert Last
6. Display
7. Delete First
8. Delete Random
9. Delete Middle
10. Delete Last
11. Delete specific data
12. Size
13. Exit
Enter your choice : 10
Deleted element : 2
1. Initialization
2. Insert First
3. Insert Random
4. Insert Middle
5. Insert Last
6. Display
7. Delete First
8. Delete Random
9. Delete Middle
10. Delete Last
11. Delete specific data
12. Size
13. Exit
Enter your choice : 3
Enter element to insert : 2
Enter position to insert : 0
Insertion not possible
1. Initialization
2. Insert First
3. Insert Random
4. Insert Middle
5. Insert Last
6. Display
7. Delete First
8. Delete Random
9. Delete Middle
10. Delete Last
11. Delete specific data
12. Size
13. Exit
Enter your choice : 13
We are exiting...Good Bye...
Process exited after 02.29 seconds with return value 13
Press any key to continue . . .

```

## Q29. WAP for LL Create, Reversal and Display.

Ans29.

### C PROGRAM –

```

#include<stdio.h>
#include<stdlib.h>
struct node
{
    int info;
    struct node *next;
};
typedef struct node node;
node *createnode(int);
node *create(node *);
void display(node *);
node *reversll(node *);
void main()
{
    node *s=NULL;
    int ch;
    do
    {
        printf("\n 1. CREATE \n 2. DISPLAY \n 3. REVERSE \n 0. EXIT");
        printf("\n Enter your choice : ");

```

```

scanf("%d",&ch);
switch(ch)
{
    case 1:
        s=create(s);
        break;
    case 2:
        display(s);
        break;
    case 3:
        s=reversll(s);
        break;
    case 0:
        printf("\n Exiting");
        break;
    default:
        printf("\n Wrong choice");
}
}while(ch!=0);
}

node * createnode(int x)
{
    node *temp;
    temp=(node *)malloc(sizeof(node));
    temp->info=x;
    temp->next=NULL;
    return temp;
}

node * create(node *s)
{
    node *temp,*p;
    int n,x,i;
    printf("\n Enter number of nodes in linked list : ");
    scanf("%d",&n);
    for(i=0;i<n;i++)
    {
        printf("\n Enter data : ");
        scanf("%d",&x);
        temp=createnode(x);
        if(s==NULL)
            s=temp;
        else
        {
            for(p=s;p->next!=NULL;p=p->next);
            p->next=temp;
        }
    }
    return s;
}

```

```

void display(node *s)
{
    node *p;
    if(s==NULL)
        printf("\n Linked list is empty");
    else
    {
        printf("\n LINKED LIST :\n");
        for(p=s;p!=NULL;p=p->next)
            printf(" %d ",p->info);
    }
}

node * reversll(node *s)
{
    node *pn=NULL,*cn=s,*nn=s;
    if(s==NULL || s->next==NULL)
        return s;
    else
    {
        while(nn!=NULL)
        {
            nn=nn->next;
            cn->next=pn;
            pn=cn;
            cn=nn;
        }
        s=pn;
        return s;
    }
}

```

## OUTPUT-

```

E:\EDS LAB\versalotfile.exe
1. CREATE
2. DISPLAY
3. REVERSE
0. EXIT
Enter your choice : 1
Enter number of nodes in linked list : 5
Enter data : 1
Enter data : 3
Enter data : 2
Enter data : 5
Enter data : 6
1. CREATE
2. DISPLAY
3. REVERSE
0. EXIT
Enter your choice : 2
LINKED LIST :
1. CREATE
2. DISPLAY
3. REVERSE
0. EXIT
Enter your choice : 3
LINKED LIST :
6 5 2 3 1
1. CREATE
2. DISPLAY
3. REVERSE
0. EXIT
Enter your choice :

```

## Q30. WAP for LL Create, Search, Sort and Display.

Ans30

### C PROGRAM –

```
#include<stdio.h>
```

```

#include<stdlib.h>
struct node
{
    int info;
    struct node *next;
};
typedef struct node node;
node * createnode(int);
node * create(node *);
void display(node *);
node * sort(node *);
void lsearch(node *,int);
void main()
{
    node *s=NULL;
    int ch,key;
    do
    {
        printf("\n 1. CREATE \n 2. DISPLAY \n 3. SEARCH \n 4. SORT \n 0. EXIT");
        printf("\n Enter your choice : ");
        scanf("%d",&ch);
        switch(ch)
        {
            case 1:
                s=create(s);
                break;
            case 2:
                display(s);
                break;
            case 3:
                printf("\n Enter element to search : ");
                scanf("%d",&key);
                lsearch(s,key);
                break;
            case 4:
                s=sort(s);
                break;
            case 0:
                printf("\n Exiting");
                break;
            default:
                printf("\n Wrong choice");
        }
    }while(ch!=0);
}
node * createnode(int x)
{
    node *temp;
    temp=(node *)malloc(sizeof(node));

```

```

temp->info=x;
temp->next=NULL;
return temp;
}
node * create(node *s)
{
node *temp,*p;
int n,x,i;
printf("\n Enter number of nodes in linked list : ");
scanf("%d",&n);
for(i=0;i<n;i++)
{
    printf("\n Enter data : ");
    scanf("%d",&x);
    temp=createnode(x);
    if(s==NULL)
        s=temp;
    else
    {
        for(p=s;p->next!=NULL;p=p->next);
        p->next=temp;
    }
}
return s;
}
void display(node *s)
{
node *p;
if(s==NULL)
printf("\n Linked list is empty");
else
{
    printf("\n LINKED LIST :\n");
    for(p=s;p!=NULL;p=p->next)
        printf(" %d ",p->info);
}
}
node * sort(node *s)
{
node *c=s,*n=NULL;
int temp;
if(s==NULL || s->next==NULL)
return s;
else
{
    while(c!=NULL)
    {
        n=c->next;
        while(n!=NULL)

```

```

    {
        if(c->info>n->info)
        {
            temp=c->info;
            c->info=n->info;
            n->info=temp;
        }
        n=n->next;
    }
    c=c->next;
}
return s;
}

void lsearch(node *s,int y)
{
node *p;
int c=0,i;
if(s==NULL)
printf("\n Linked list is empty");
else
{
    for(p=s,i=1;p!=NULL;i++,p=p->next)
    {
        if(y==p->info)
        {
            printf("\n %d found at %d position",y,i);
            c++;
        }
    }
    if(c==0)
    printf("\n %d not found",y);
    else
    printf("\n %d found %d times",y,c);
}
}

```

**OUTPUT-**

```

E:\CDS LAB\sortll.exe
1. CREATE
2. DISPLAY
3. SEARCH
4. SORT
0. EXIT
Enter your choice : 1
Enter number of nodes in linked list : 6
Enter data : 1
Enter data : 3
Enter data : 2
Enter data : 7
Enter data : 0
Enter data : 9
1. CREATE
2. DISPLAY
3. SEARCH
4. SORT
0. EXIT
Enter your choice : 2
LINKED LIST :
1 2 7 0 9
1. CREATE
2. DISPLAY
3. SEARCH
4. SORT
0. EXIT
Enter your choice : 4
1. CREATE
2. DISPLAY
3. SEARCH
4. SORT
0. EXIT
Enter your choice : 2
LINKED LIST :
0 1 2 3 7 9
1. CREATE
2. DISPLAY
3. SEARCH
4. SORT
0. EXIT
Enter your choice : 3
Enter element to search : 8
8 not found
1. CREATE
2. DISPLAY
3. SEARCH
4. SORT
0. EXIT
Enter your choice : 3
Enter element to search : 3
3 found at 4 position
3 found 1 times
1. CREATE
2. DISPLAY
3. SEARCH
4. SORT
0. EXIT
Enter your choice : 0
Exiting
Process exited after 48.51 seconds with return value 0
Press any key to continue . . .

```

### Q31. WAP for LL Create, Split (both) and Display.

Ans31.

#### C PROGRAM –

```

#include<stdio.h>
#include<stdlib.h>
struct node
{
    int info;
    struct node *next;
};
typedef struct node node;
node *createnode(int);
node *create(node *);
void display(node *);
void split(node *,int);
int size(node *s);
void main()
{
    node *s=NULL;
    int ch,pos;
    do
    {
        printf("\n 1. CREATE \n 2. DISPLAY \n 3. SPLIT \n 0. EXIT");

```

```

printf("\n Enter your choice : ");
scanf("%d",&ch);
switch(ch)
{
    case 1:
        s=create(s);
        break;
    case 2:
        display(s);
        break;
    case 3:
        printf("\n Enter position to split from : ");
        scanf("%d",&pos);
        split(s,pos);
        break;
    case 0:
        printf("\n Exiting");
        break;
    default:
        printf("\n Wrong choice");
}
}while(ch!=0);
}

node * createnode(int x)
{
    node *temp;
    temp=(node *)malloc(sizeof(node));
    temp->info=x;
    temp->next=NULL;
    return temp;
}

node * create(node *s)
{
    node *temp,*p;
    int n,x,i;
    printf("\n Enter number of nodes in linked list : ");
    scanf("%d",&n);
    for(i=0;i<n;i++)
    {
        printf("\n Enter data : ");
        scanf("%d",&x);
        temp=createnode(x);
        if(s==NULL)
            s=temp;
        else
        {
            for(p=s;p->next!=NULL;p=p->next);
            p->next=temp;
        }
    }
}

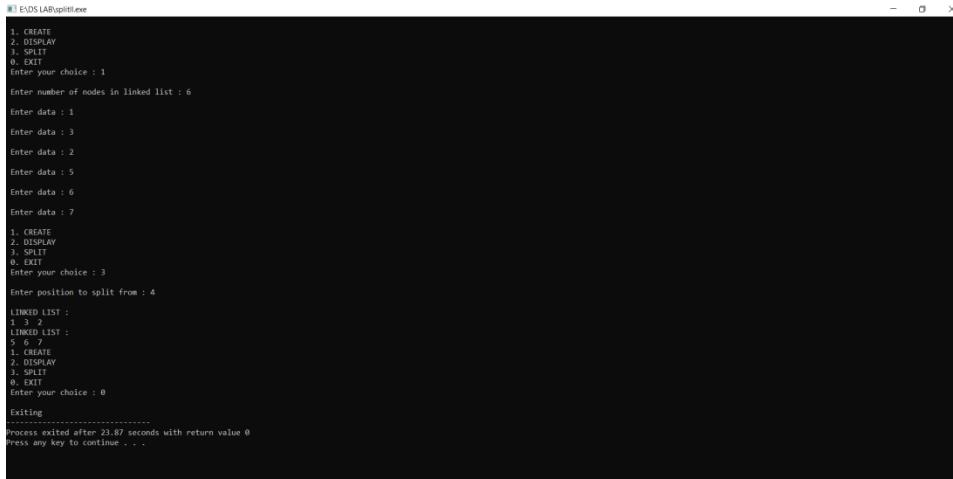
```

```

    }
    return s;
}
void display(node *s)
{
    node *p;
    if(s==NULL)
        printf("\n Linked list is empty");
    else
    {
        printf("\n LINKED LIST :\n");
        for(p=s;p!=NULL;p=p->next)
            printf(" %d ",p->info);
    }
}
int size(node *s)
{
    node *p;
    int i=0;
    if(s==NULL)
        return i;
    else
    {
        for(p=s,i=1;p->next!=NULL;i++,p=p->next);
        return i;
    }
}
void split(node *s,int pos)
{
    node *p=s,*q=NULL;
    int l=size(s),i;
    if(l==0)
        return ;
    else if(pos<2 || pos>l)
    {
        printf("\n Invalid position for splitting");
        return ;
    }
    else
    {
        for(p=s,i=1;i<(pos-1);i++,p=p->next);
        q=p->next;
        p->next=NULL;
        p=s;
        display(p);
        display(q);
        return;
    }
}

```

## OUTPUT-



```
W:\ESDS LAB\split1.exe
1. CREATE
2. DISPLAY
3. SPLIT
0. EXIT
Enter your choice : 1
Enter number of nodes in linked list : 6
Enter data : 1
Enter data : 2
Enter data : 3
Enter data : 4
Enter data : 5
Enter data : 6
Enter data : 7
1. CREATE
2. DISPLAY
3. SPLIT
0. EXIT
Enter your choice : 3
Enter position to split from : 4
LINKED LIST :
1 3 2
LINKED LIST :
5 6 7
1. CREATE
2. DISPLAY
3. SPLIT
0. EXIT
Enter your choice : 0
Exiting
Process exited after 23.07 seconds with return value 0
Press any key to continue . . .
```

## Q32. WAP for LL Create, Merge (both) and Display.

Ans32.

### C PROGRAM –

```
#include<stdio.h>
#include<stdlib.h>
struct node
{
    int info;
    struct node *next;
};
typedef struct node node;
node *createnode(int);
node *create(node *);
void display(node *);
void sortedmerge(node *,node *);
node * sort(node *);
node * insertl(node *,int);
void main()
{
    node *s=NULL,*f=NULL,*p=NULL;
    int ch;
    do
    {
        printf("\n 1. CREATE \n 2. DISPLAY \n 3. SORTED MERGE \n 0. EXIT");
        printf("\n Enter your choice : ");
        scanf("%d",&ch);
        switch(ch)
        {
            case 1:
                printf("\n ***** LINKED LIST 1 *****\n");
                s=create(s);
                printf("\n ***** LINKED LIST 2 *****\n");
                f=create(f);
                break;
```

```

        case 2:
            printf("\n ***** LINKED LIST 1 *****\n");
            display(s);
            printf("\n ***** LINKED LIST 2 *****\n");
            display(f);
            break;
        case 3:
            sortedmerge(s,f);
            break;
        case 0:
            printf("\n Exiting");
            break;
        default:
            printf("\n Wrong choice");
    }
}
}while(ch!=0);
}

node * createnode(int x)
{
    node *temp;
    temp=(node *)malloc(sizeof(node));
    temp->info=x;
    temp->next=NULL;
    return temp;
}

node * create(node *s)
{
    node *temp,*p;
    int n,x,i;
    printf("\n Enter number of nodes in linked list : ");
    scanf("%d",&n);
    for(i=0;i<n;i++)
    {
        printf("\n Enter data : ");
        scanf("%d",&x);
        temp=createnode(x);
        if(s==NULL)
            s=temp;
        else
        {
            for(p=s;p->next!=NULL;p=p->next);
            p->next=temp;
        }
    }
    return s;
}

void display(node *s)
{
    node *p;

```

```

if(s==NULL)
printf("\n Linked list is empty");
else
{
    printf("\n LINKED LIST :\n");
    for(p=s;p!=NULL;p=p->next)
        printf(" %d ",p->info);
}
}

void sortedmerge(node *s,node *f)
{
    node *p=NULL,*t,*u;
    if(s==NULL && f==NULL)
    {
        printf("\n Linked lists are empty");
        return;
    }
    else if(s!=NULL && f==NULL)
    {
        s=sort(s);
        display(s);
    }
    else if(s==NULL && f!=NULL)
    {
        f=sort(f);
        display(f);
    }
    else
    {
        s=sort(s);
        f=sort(f);
        t=s;
        u=f;
        while(t!=NULL && u!=NULL)
        {
            if(t->info<u->info)
            {
                p=insertl(p,t->info);
                t=t->next;
            }
            else if(u->info<t->info)
            {
                p=insertl(p,u->info);
                u=u->next;
            }
            else
            {
                p=insertl(p,t->info);
                t=t->next;
            }
        }
    }
}

```

```

        p=insertl(p,u->info);
        u=u->next;
    }
}
while(t!=NULL)
{
    p=insertl(p,t->info);
    t=t->next;
}
while(u!=NULL)
{
    p=insertl(p,u->info);
    u=u->next;
}
display(p);
}

node * sort(node *s)
{
    node *c=s,*n=NULL;
    int temp;
    if(s==NULL || s->next==NULL)
        return s;
    else
    {
        while(c!=NULL)
        {
            n=c->next;
            while(n!=NULL)
            {
                if(c->info>n->info)
                {
                    temp=c->info;
                    c->info=n->info;
                    n->info=temp;
                }
                n=n->next;
            }
            c=c->next;
        }
        return s;
    }
}
node * insertl(node *s,int x)
{
    node *temp,*p;
    temp=(node *)malloc(sizeof(node));
    temp->info=x;
    temp->next=NULL;
}

```

```

    if(s==NULL)
        s=temp;
    else
    {
        for(p=s;p->next!=NULL;p=p->next);
        p->next=temp;
    }
    return s;
}

```

### OUTPUT-

```

E:\DS LAB\sortmerge.c
1. CREATE
2. DISPLAY
3. SORTED MERGE
0. EXIT
Enter your choice : 1
***** LINKED LIST 1 *****
Enter number of nodes in linked list : 3
Enter data : 2
Enter data : 1
Enter data : 3
***** LINKED LIST 2 *****
Enter number of nodes in linked list : 4
Enter data : 1
Enter data : 2
Enter data : 5
Enter data : 4
1. CREATE
2. DISPLAY
3. SORTED MERGE
0. EXIT
Enter your choice : 3
LINKED LIST :
0 1 2 3 5
1. CREATE
2. DISPLAY
3. SORTED MERGE
0. EXIT
Enter your choice :

```

**Q33. WAP for LL Create, Other remaining operations like Duplicate Removal, movement of node etc. and Display.**

Ans33.

### C PROGRAM –

```

#include<stdio.h>
#include<stdlib.h>
struct node
{
    int info;
    struct node *next;
};
typedef struct node node;
node * createnode(int);
node * create(node *);
void dupremoval(node *);
void display(node *);
node * move(node *,int,int);
int size(node *s);
int search(node *,int);
node * swap(node *,int,int);
node * insertsorted(node *,int);
node * sort(node *);
void split(node *);
node * insertl(node *,int);
void main()

```

```

{
node *s=NULL;
int ch,p,q,x,y,pos,i,j;
do
{
    printf("\n 1. CREATE \n 2. DISPLAY \n 3. DUPLICATE REMOVAL \n 4. MOVE NODE \n 5. SWAP NODE");
    printf("\n 6. INSERT SORTED \n 7. SPLIT IN ODD AND EVEN \n 0. EXIT");
    printf("\n Enter your choice : ");
    scanf("%d",&ch);
    switch(ch)
    {
        case 1:
            s=create(s);
            break;
        case 2:
            display(s);
            break;
        case 3:
            if(s!=NULL)
                dupremoval(s);
            else
                printf("\n LINKED LIST IS EMPTY");
            break;
        case 4:
            if(s!=NULL)
            {
                do
                {
                    printf("\n Enter position of node to move : ");
                    scanf("%d",&p);
                    printf("\n Enter position to move node to : ");
                    scanf("%d",&q);
                    }while((p>size(s) || p<1) || (q>size(s) || q<1));
                    s=move(s,p,q);
                }
            else
                printf("\n LINKED LIST IS EMPTY");
            break;
        case 5:
            if(s!=NULL)
            {
                printf("\n Enter element to search : ");
                scanf("%d",&x);
                i=search(s,x);
                printf("\n Enter element to search : ");
                scanf("%d",&y);
                j=search(s,y);
                s=swap(s,i,j);
            }
}

```

```

        else
            printf("\n LINKED LIST IS EMPTY");
        break;
    case 6:
        printf("\n Enter element to insert : ");
        scanf("%d",&x);
        s=insertsorted(s,x);
        break;
    case 7:
        if(s!=NULL)
            split(s);
        else
            printf("\n LINKED LIST IS EMPTY");
        break;
    case 0:
        printf("\n Exiting");
        break;
    default:
        printf("\n Wrong choice");
    }
}
}while(ch!=0);
}

node * createnode(int x)
{
    node *temp;
    temp=(node *)malloc(sizeof(node));
    temp->info=x;
    temp->next=NULL;
    return temp;
}

node * create(node *s)
{
    node *temp,*p;
    int n,x,i;
    printf("\n Enter number of nodes in linked list : ");
    scanf("%d",&n);
    for(i=0;i<n;i++)
    {
        printf("\n Enter data : ");
        scanf("%d",&x);
        temp=createnode(x);
        if(s==NULL)
            s=temp;
        else
        {
            for(p=s;p->next!=NULL;p=p->next);
            p->next=temp;
        }
    }
}

```

```

    return s;
}
void display(node *s)
{
    node *p;
    if(s==NULL)
        printf("\n LINKED LIST IS EMPTY");
    else
    {
        printf("\n LINKED LIST :\n");
        for(p=s;p!=NULL;p=p->next)
            printf(" %d ",p->info);
    }
}
void dupremoval(node *s)
{
    node *p,*q,*r,*t;
    for(p=s;p!=NULL;p=p->next)
    {
        r=p;
        q=p->next;
        while(q!=NULL)
        {
            if(p->info==q->info)
            {
                r->next=q->next;
                t=q;
                q=q->next;
                free(t);
            }
            else
            {
                r=q;
                q=q->next;
            }
        }
    }
}
int search(node *s,int x)
{
    node *p;
    int flag=0,i;
    for(p=s,i=1;p!=NULL;i++,p=p->next)
    {
        if(p->info==x)
        {
            flag=1;
            return i;
        }
    }
}

```

```

        else
            continue;
    }
    if(p==NULL && flag==0)
        return 0;
}
node * move(node *s,int x,int y)
{
    int pos;
    node *p,*t,*q;
    if(x==1)
    {
        t=s;
        s=s->next;
    }
    else
    {
        for(pos=1,p=s;pos<(x-1);pos++,p=p->next);
        t=p->next;
        p->next=t->next;
    }
    if(y==1)
    {
        t->next=s;
        s=t;
    }
    else
    {
        for(pos=1,p=s;pos<(y-1);pos++,p=p->next);
        q=p->next;
        t->next=q;
        p->next=t;
    }
    return s;
}
int size(node *s)
{
    int i;
    node *p;
    if(s==NULL)
        return 0;
    else
    {
        for(i=1,p=s;p->next!=NULL;i++,p=p->next);
        return i;
    }
}
node * swap(node *s,int i,int j)
{

```

```

int pos;
node *temp,*p,*t=NULL,*prevx,*px,*prevy,*py;
for(p=s, pos=1; pos<i; pos++, t=p, p=p->next);
    prevx=t;
    px=p;
    for(p=s, pos=1; pos<j; pos++, t=p, p=p->next);
        prevy=t;
        py=p;
    //This is main swap function
    temp=py->next;
    py->next=px->next;
    px->next=temp;
    if(prevx==NULL)
    {
        s=py;
        prevy->next=px;
    }
    if(prevy==NULL)
    {
        s=px;
        prevx->next=py;
    }
    if(prevx!=NULL && prevy!=NULL)
    {
        prevx->next=py;
        prevy->next=px;
    }
    return s;
}
node * sort(node *s)
{
    node *c=s,*n=NULL;
    int temp;
    if(s==NULL || s->next==NULL)
        return s;
    else
    {
        while(c!=NULL)
        {
            n=c->next;
            while(n!=NULL)
            {
                if(c->info>n->info)
                {
                    temp=c->info;
                    c->info=n->info;
                    n->info=temp;
                }
                n=n->next;
            }
        }
    }
}

```

```

        }
        c=c->next;
    }
    return s;
}
}

node * insertsorted(node *s,int x)
{
    node *p,*temp,*r=NULL;
    temp=createnode(x);
    if(s==NULL)
    {
        s=temp;
        return s;
    }
    else
    {
        s=sort(s);
    }
    for(p=s;p!=NULL;r=p,p=p->next)
    {
        if(p->info>temp->info || p->info==temp->info)
        {
            if(p==s)
            {
                temp->next=s;
                s=temp;
                return s;
            }
            else
            {
                temp->next=r->next;
                r->next=temp;
                return s;
            }
        }
        else
        continue;
    }
    for(p=s;p->next!=NULL;p=p->next);
    p->next=temp;
    return s;
}
}

void split(node *s)
{
    node *p=NULL,*q=NULL,*t;
    for(t=s;t!=NULL;t=t->next)
    {
        if(t->info%2==0)

```

```

    p=insertl(p,t->info);
    else
        q=insertl(q,t->info);
}
display(p);
display(q);
}

node * insertl(node *s,int x)
{
    node *temp,*p;
    temp=(node *)malloc(sizeof(node));
    temp->info=x;
    temp->next=NULL;
    if(s==NULL)
        s=temp;
    else
    {
        for(p=s;p->next!=NULL;p=p->next);
        p->next=temp;
    }
    return s;
}

```

**OUTPUT-**

```

E:\DVS LAB\6.exe
1. CREATE
2. DISPLAY
3. DUPLICATE REMOVAL
4. MOVE NODE
5. SWAP NODE
6. INSERT SORTED
7. SPLIT IN ODD AND EVEN
0. EXIT
Enter your choice : 1
Enter number of nodes in linked list : 5
Enter data : 11
Enter data : 22
Enter data : 22
Enter data : 77
Enter data : 22
1. CREATE
2. DISPLAY
3. DUPLICATE REMOVAL
4. MOVE NODE
5. SWAP NODE
6. INSERT SORTED
7. SPLIT IN ODD AND EVEN
0. EXIT
Enter your choice : 2
LINKED LIST :
11 22 22 77 22
1. CREATE
2. DISPLAY
3. DUPLICATE REMOVAL
4. MOVE NODE
5. SWAP NODE
6. INSERT SORTED
7. SPLIT IN ODD AND EVEN
0. EXIT
Enter your choice : 3
4. MOVE NODE
5. SWAP NODE
6. INSERT SORTED
7. SPLIT IN ODD AND EVEN
0. EXIT
Enter your choice : 2
LINKED LIST :
11 22 77
1. CREATE
2. DISPLAY
3. DUPLICATE REMOVAL
4. MOVE NODE
5. SWAP NODE
6. INSERT SORTED
7. SPLIT IN ODD AND EVEN
0. EXIT
Enter your choice : 5
Enter element to search : 11
Enter element to search : 77
1. CREATE
2. DISPLAY
3. DUPLICATE REMOVAL
4. MOVE NODE
5. SWAP NODE
6. INSERT SORTED
7. SPLIT IN ODD AND EVEN
0. EXIT
Enter your choice : 2
LINKED LIST :
11 22 77
1. CREATE
2. DISPLAY
3. DUPLICATE REMOVAL
4. MOVE NODE
5. SWAP NODE
6. INSERT SORTED
7. SPLIT IN ODD AND EVEN
0. EXIT
Enter your choice : 6
Enter element to insert : 55
1. CREATE
2. DISPLAY
3. DUPLICATE REMOVAL
4. MOVE NODE
5. SWAP NODE
6. INSERT SORTED
7. SPLIT IN ODD AND EVEN
0. EXIT
Enter your choice : 2
LINKED LIST :
11 22 55 77
1. CREATE
2. DISPLAY
3. DUPLICATE REMOVAL
4. MOVE NODE
5. SWAP NODE
6. INSERT SORTED
7. SPLIT IN ODD AND EVEN
0. EXIT
Enter your choice : 7
LINKED LIST :
11 55 77
1. CREATE
2. DISPLAY
3. DUPLICATE REMOVAL
4. MOVE NODE
5. SWAP NODE
6. INSERT SORTED
7. SPLIT IN ODD AND EVEN
0. EXIT
Enter your choice : 0
Exiting
Process exited after 66.88 seconds with return value 0
Press any key to continue . . .

```

### Q34. WAP for implementing Doubly Linked List with all operations.

Ans34.

#### C PROGRAM –

```

#include<stdio.h>
#include<malloc.h>
struct dnode
{
    int data;
    struct dnode *next;
    struct dnode *prev;
};

```

```

typedef struct dnode dnode;
dnode * createnode(int);
dnode * init(dnode *);
dnode * insertf(dnode *,int);
dnode * insertl(dnode *,int);
dnode * insertp(dnode *,int,int);
dnode * deletef(dnode *);
dnode * deletel(dnode *);
dnode * deletep(dnode *,int);
void fdisplay(dnode *);
void bdisplay(dnode *);
int size(dnode *);
dnode * reversdll(dnode *);
void main()
{
    dnode *start=NULL;
    int ch,a,b,c,pos,p;
    do
    {
        printf("\n 1. Initialization \n 2. Insert first \n 3. Insert by position \n 4. Insert last \n 5. Forward
display");
        printf("\n 6. Backward display \n 7. Delete first \n 8. Delete by position \n 9. Delete last \n 10.
Size \n 11. Reverse\n 12. Exit");
        printf("\n Enter your choice : ");
        scanf("%d",&ch);
        switch(ch)
        {
            case 1:
                start=init(start);
                break;
            case 2:
                printf("\n Enter element to insert : ");
                scanf("%d",&a);
                start=insertf(start,a);
                break;
            case 3:
                printf("\n Enter element to insert : ");
                scanf("%d",&b);
                printf("\n Enter position to insert : ");
                scanf("%d",&p);
                start=insertp(start,b,p);
                break;
            case 4:
                printf("\n Enter element to insert : ");
                scanf("%d",&c);
                start=insertl(start,c);
                break;
            case 5:
                fdisplay(start);
        }
    }
}

```

```

        break;
    case 6:
        bdisplay(start);
        break;
    case 7:
        start=deletef(start);
        break;
    case 8:
        printf("\n Enter position to delete : ");
        scanf("%d",&pos);
        start=deletep(start,pos);
        break;
    case 9:
        start=deletel(start);
        break;
    case 10:
        if(size(start)==0)
            printf("\n Size of doubly linked list : 0");
        else
            printf("\n Size of doubly linked list : %d",size(start));
        break;
    case 11:
        start=reversdll(start);
        break;
    case 12:
        printf("\n Exiting...!");
        break;
    default:
        printf("\n Wrong choice entered");
    }
}while(ch!=12);
}
dnode * createnode(int x)
{
    dnode *temp;
    temp=(dnode *)malloc(sizeof(dnode));
    temp->data=x;
    temp->prev=NULL;
    temp->next=NULL;
    return temp;
}
dnode * init(dnode *start)
{
    start=NULL;
    return start;
}
dnode * insertf(dnode *start,int x)
{
    dnode *temp;

```

```

temp=createnode(x);
if(start==NULL)
{
    start=temp;
    return start;
}
else
{
    temp->next=start;
    start->prev=temp;
    start=temp;
    return start;
}
}

dnode * insertl(dnode *start,int x)
{
    dnode *temp,*p;
    temp=createnode(x);
    if(start==NULL)
    {
        start=insertf(start,x);
        return start;
    }
    else
    {
        for(p=start;p->next!=NULL;p=p->next);
        {
            p->next=temp;
            temp->prev=p;
            return start;
        }
    }
}
dnode * insertp(dnode *start,int x, int pos)
{
    int s=size(start),i;
    dnode *temp,*p=start,*t;
    if(pos<1 || pos>(s+1))
    {
        printf("\n Insertion not possible due to invalid position");
        return start;
    }
    else if(pos==1)
    {
        start=insertf(start,x);
        return start;
    }
    else if(pos==(s+1))
    {

```

```

        start=insertl(start,x);
        return start;
    }
else
{
    temp=createnode(x);
    for(i=1,p=start;i<pos;i++,p=p->next);
    t=p->prev;
    t->next=temp;
    temp->prev=t;
    temp->next=p;
    p->prev=temp;
    return start;
}
}

dnode * deletef(dnode *start)
{
    dnode *p;
    if(start==NULL)
    {
        printf("\n Deletion not possible as doubly linked list is empty");
        return start;
    }
    else if(start->next==NULL)
    {
        printf("\n %d deleted",start->data);
        free(start);
        start=NULL;
        return start;
    }
    else
    {
        p=start;
        start=start->next;
        start->prev=NULL;
        printf("\n %d deleted",p->data);
        free(p);
        return start;
    }
}

dnode * deletel(dnode *start)
{
    dnode *t,*p;
    if(start==NULL)
    {
        printf("\n Deletion not possible as doubly linked list is empty");
        return start;
    }
    else if(start->next==NULL)

```

```

{
    printf("\n %d deleted",start->data);
    free(start);
    start=NULL;
    return start;
}
else
{
    for(p=start;p->next!=NULL;p=p->next);
    {
        t=p->prev;
        t->next=NULL;
        printf("\n %d deleted",p->data);
        free(p);
        return start;
    }
}
dnode * deletep(dnode *start,int pos)
{
    int s=size(start),i;
    dnode *p=start,*t,*temp;
    if(start==NULL)
    {
        printf("\n Deletion not possible as doubly linked list is empty");
        return start;
    }
    else if(pos<1 || pos>s)
    {
        printf("\n Deletion not possible due to invalid position");
        return start;
    }
    else if(pos==1)
    {
        start=deletef(start);
        return start;
    }
    else if(pos==s)
    {
        start=deletel(start);
        return start;
    }
    else
    {
        for(i=1,p=start;i<pos;i++,p=p->next);
        t=p->prev;
        temp=p->next;
        t->next=temp;
        temp->prev=t;
    }
}

```

```

        printf("\n %d deleted",p->data);
        free(p);
        return start;
    }
}

int size(dnode *start)
{
    int i;
    dnode *p;
    if(start==NULL)
        return 0;
    else
    {
        for(i=1,p=start;p->next!=NULL;i++,p=p->next);
        return i;
    }
}

void fdisplay(dnode *start)
{
    dnode *p;
    if(start==NULL)
        printf("\n Doubly linked list is empty");
    else
    {
        printf("\n Doubly linked list : \n");
        for(p=start;p!=NULL;p=p->next)
            printf("%d\t",p->data);
    }
}

void bdisplay(dnode *start)
{
    dnode *p;
    if(start==NULL)
        printf("\n Doubly linked list is empty");
    else
    {
        printf("\n Doubly linked list : \n");
        for(p=start;p->next!=NULL;p=p->next);
            for( ;p!=NULL;p=p->prev)
                printf("%d\t",p->data);
    }
}

dnode * reversdll(dnode *start)
{
    dnode *cn=start,*nn;
    if(start==NULL || start->next==NULL)
        return start;
    else
    {

```

```

while(cn!=NULL)
{
    start=cn;
    nn=cn->next;
    cn->next=cn->prev;
    cn->prev=nn;
    cn=nn;
}
return start;
}
}

```

## OUTPUT-

```

E:\DUS\LAB\Buline
1. Initialization
2. Insert first
3. Insert by position
4. Insert last
5. Forward display
6. Backward display
7. Delete first
8. Delete by position
9. Delete last
10. Size
11. Reverse
12. Exit
Enter your choice : 2
Enter element to insert : 1
1. Initialization
2. Insert first
3. Insert by position
4. Insert last
5. Forward display
6. Backward display
7. Delete first
8. Delete by position
9. Delete last
10. Size
11. Reverse
12. Exit
Enter your choice : 2
Enter element to insert : 3
1. Initialization
2. Insert first
3. Insert by position
4. Insert last
5. Forward display
6. Backward display
7. Delete first
8. Delete by position
9. Delete last
10. Size
11. Reverse
12. Exit
Enter your choice : 2
Enter element to insert : 5
1. Initialization
2. Insert first
3. Insert by position
4. Insert last
5. Forward display
6. Backward display
7. Delete first
8. Delete by position
9. Delete last
10. Size
11. Reverse
12. Exit
Enter your choice : 5
Enter element to insert : 5
1. Initialization
2. Insert first
3. Insert by position
4. Insert last
5. Forward display
6. Backward display
7. Delete first
8. Delete by position
9. Delete last
10. Size
11. Reverse
12. Exit
Enter your choice : 5
Doubly linked list :
1. Initialization
2. Insert first
3. Insert by position
4. Insert last
5. Forward display
6. Backward display
7. Delete first
8. Delete by position
9. Delete last
10. Size
11. Reverse
12. Exit
Enter your choice : 6
Doubly linked list :
1. Initialization
2. Insert first
3. Insert by position
4. Insert last
5. Forward display
6. Backward display
7. Delete first
8. Delete by position
9. Delete last
10. Size
11. Reverse
12. Exit
Enter your choice : 10
Size of doubly linked list : 3
1. Initialization
2. Insert first
3. Insert by position
4. Insert last
5. Forward display
6. Backward display
7. Delete first
8. Delete by position
9. Delete last
10. Size
11. Reverse
12. Exit
Enter your choice : 11
Doubly linked list :
1. Initialization
2. Insert first
3. Insert by position
4. Insert last
5. Forward display
6. Backward display
7. Delete first
8. Delete by position
9. Delete last
10. Size
11. Reverse
12. Exit
Enter your choice : 5
Doubly linked list :
1. Initialization
2. Insert first
3. Insert by position
4. Insert last
5. Forward display
6. Backward display
7. Delete first
8. Delete by position
9. Delete last
10. Size
11. Reverse
12. Exit
Enter your choice : 12
Exiting...
Process exited after 52.6 seconds with return value 32
Press any key to continue . .

```

### **Q35. WAP for implementing Circular Linked List with all operations.**

Ans35.

#### **C PROGRAM –**

```
#include<stdio.h>
#include<stdlib.h>
struct node
{
    int data;
    struct node *next;
};
typedef struct node cnode;
cnode * init(cnode *);
cnode * createnode(int);
int size(cnode *);
cnode * insertf(cnode *,int);
cnode * insertl(cnode *,int);
cnode * insertr(cnode *,int,int);
cnode * deletef(cnode *);
cnode * deletel(cnode *);
cnode * deleter(cnode *,int);
void display(cnode *);
cnode * revercll(cnode *s);
void main()
{
    cnode *start=NULL;
    int ch,a,b,c,p,l,s;
    do
    {
        printf("\n1. Initialization \n2. Insert first \n3. Insert random \n4. Insert last \n5. Display");
        printf("\n6. Delete first \n7. Delete random \n8. Delete last \n9. Size \n10. Reverse \n11. Exit");
        printf("\n Enter your choice : ");
        scanf("%d",&ch);
        switch(ch)
        {
            case 1:
                start=init(start);
                break;
            case 2:
                printf("\n Enter element to insert : ");
                scanf("%d",&a);
                start=insertf(start,a);
                break;
            case 3:
                printf("\n Enter element to insert : ");
                scanf("%d",&b);
                printf("\n Enter position to insert : ");
                scanf("%d",&p);
                start=insertr(start,b,p);
        }
    }
}
```

```

        break;
    case 4:
        printf("\n Enter element to insert : ");
        scanf("%d",&c);
        start=insertl(start,c);
        break;
    case 5:
        display(start);
        break;
    case 6:
        start=deletef(start);
        break;
    case 7:
        printf("\n Enter position to delete : ");
        scanf("%d",&l);
        start=deleter(start,l);
        break;
    case 8:
        start=deletel(start);
        break;
    case 9:
        s=size(start);
        if(s!=0)
            printf("\n Size of circular linked list : %d",s);
        else
            printf("\n Size of circular linked list : 0");
        break;
    case 10:
        start=revercll(start);
        break;
    case 11:
        printf("\n Exiting");
        break;
    default:
        printf("\n Wrong choice entered");
    }
}while(ch!=11);
}

cnode * init(cnode *start)
{
    start=NULL;
    return start;
}
cnode * createnode(int x)
{
    cnode *temp;
    temp=(cnode *)malloc(sizeof(cnode));
    temp->data=x;
    return temp;
}

```

```

}

int size(cnode *start)
{
    cnode *p;
    int i;
    if(start==NULL)
        return 0;
    else
    {
        for(p=start,i=1;p->next!=start;i++,p=p->next);
        return i;
    }
}

cnode * insertf(cnode *start,int x)
{
    cnode *temp,*p;
    temp=createnode(x);
    if(start==NULL)
    {
        start=temp;
        temp->next=start;
        return start;
    }
    else
    {
        for(p=start;p->next!=start;p=p->next);
        p->next=temp;
        temp->next=start;
        start=temp;
        return start;
    }
}

cnode * insertl(cnode *start,int x)
{
    cnode *temp,*p;
    temp=createnode(x);
    if(start==NULL)
    {
        start=temp;
        temp->next=start;
        return start;
    }
    else
    {
        for(p=start;p->next!=start;p=p->next);
        p->next=temp;
        temp->next=start;
        return start;
    }
}

```

```

}

cnode * insertr(cnode *start,int x,int pos)
{
    cnode *temp,*p;
    int s=size(start),i;
    if(pos<1 || pos>s+1)
    {
        printf("\n Position not justified");
        return start;
    }
    else if(pos==1)
    {
        start=insertf(start,x);
        return start;
    }
    else if(pos==s+1)
    {
        start=insertl(start,x);
        return start;
    }
    else if(pos>1 && pos<=s)
    {
        temp=createnode(x);
        for(p=start,i=1;i<(pos-1);i++,p=p->next);
        temp->next=p->next;
        p->next=temp;
        return start;
    }
}
void display(cnode *start)
{
    cnode *p;
    if(start==NULL)
        printf("\n Circular linked list is empty");
    else
    {
        printf("\n Circular linked list : \n");
        for(p=start;p->next!=start;p=p->next)
        printf("%d -> ",p->data);
        printf("%d",p->data);
    }
}
cnode * deletef(cnode *start)
{
    cnode *p,*temp;
    if(start==NULL)
        printf("\n Circular linked list is empty");
    else
    {

```

```

        temp=start;
if(start->next==start)
    start=NULL;
else
{
    for(p=start;p->next!=start;p=p->next);
    start=temp->next;
    p->next=start;
}
printf("\n Deleted element : %d",temp->data);
free(temp);
}
return start;
}
cnode * deletel(cnode *start)
{
cnode *p,*t;
if(start==NULL)
printf("\n Circular linked list is empty");
else
{
    p=start;
    if(start->next==start)
        start=NULL;
    else
    {
        for(p=start;p->next!=start;t=p,p=p->next);
        t->next=start;
    }
    printf("\n Deleted element : %d",p->data);
    free(p);
}
return start;
}
cnode * deleter(cnode *start,int pos)
{
cnode *temp,*p,*t;
int s=size(start),i;
if(start==NULL)
printf("\n Circular linked list is empty");
else if(pos<1 || pos>s)
printf("\n Position not justified");
else if(pos==1)
start=deletef(start);
else if(pos==s)
start=deletel(start);
else if(pos>1 && pos<s)
{
    for(p=start,i=1;i<(pos-1);i++,p=p->next);

```

```

t=p->next;
p->next=t->next;
printf("\n Deleted element : %d",t->data);
free(t);
}
return start;
}
cnode * revercll(cnode *s)
{
cnode *c=s,*p=NULL,*n=NULL,*temp;
if(s==NULL || s->next==s)
return s;
else
{
for(c=s;c->next!=s;c=c->next);
temp=c;
n=s->next;
c=S;
while(n!=s)
{
if(c==s)
{
c->next=temp;
p=c;
c=n;
}
else
{
n=n->next;
c->next=p;
p=c;
c=n;
}
}
s=p;
return s;
}
}

```

**OUTPUT-**

```

E:\DS\LAB\c\cse
1. Initialization
2. Insert first
3. Insert random
4. Insert last
5. Display
6. Delete first
7. Delete random
8. Delete last
9. Size
10. Reverse
11. Exit
Enter your choice : 2
Enter element to insert : 1
1. Initialization
2. Insert first
3. Insert random
4. Insert last
5. Display
6. Delete first
7. Delete random
8. Delete last
9. Size
10. Reverse
11. Exit
Enter your choice : 2
Enter element to insert : 3
1. Initialization
2. Insert first
3. Insert random
4. Insert last
5. Display
6. Delete first
7. Delete random
8. Delete last
9. Size
10. Reverse
11. Exit
Enter your choice : 4
Enter element to insert : 7
E:\DS\LAB\c\cse
1. Initialization
2. Insert first
3. Insert random
4. Insert last
5. Display
6. Delete first
7. Delete random
8. Delete last
9. Size
10. Reverse
11. Exit
Enter your choice : 9
Size of circular linked list : 3
1. Initialization
2. Insert first
3. Insert random
4. Insert last
5. Display
6. Delete first
7. Delete random
8. Delete last
9. Size
10. Reverse
11. Exit
Enter your choice : 5
Circular linked list :
7 3 1
1. Initialization
2. Insert first
3. Insert random
4. Insert last
5. Display
6. Delete first
7. Delete random
8. Delete last
9. Size
10. Reverse
11. Exit
Enter your choice : 2
Enter position to delete : 2
Deleted element : 3
1. Initialization
2. Insert first
3. Insert random
4. Insert last
5. Display
6. Delete first
7. Delete random
8. Delete last
9. Size
10. Reverse
11. Exit
Enter your choice : 5
Circular linked list :
7 1
1. Initialization
2. Insert first
3. Insert random
4. Insert last
5. Display
6. Delete first
7. Delete random
8. Delete last
9. Size
10. Reverse
11. Exit
Enter your choice : 10
1. Initialization
2. Insert first
3. Insert random
4. Insert last
5. Display
6. Delete first
7. Delete random
8. Delete last
9. Size
10. Reverse
11. Exit
Enter your choice : 5
Circular linked list :
7 3 1
1. Initialization
2. Insert first
3. Insert random
4. Insert last
5. Display
6. Delete first
7. Delete random
8. Delete last
9. Size
10. Reverse
11. Exit
Enter your choice : 11
Exiting

```

### Q36. WAP for Header L.L. with Insertion, Delete and Display.

Ans36.

#### C PROGRAM –

```

#include<stdio.h>
#include<malloc.h>
#include<stdlib.h>
struct node
{
    int data;
    struct node * next;
};

```

```

struct head
{
    int hdata;
    struct node *next;
};

typedef struct node node;
typedef struct head head;
head * init(head *);
node * createnode(int);
int size(head *);
head * insertf(head *,int);
head * insertl(head *,int);
head * insertr(head *,int,int);
head * deletef(head *);
head * deletel(head *);
head * deleter(head *,int);
void display(head *);
void main()
{
    head *h;
    int ch,a,b,c,p,l;
    h=(head *)malloc(sizeof(head));
    h->hdata=0;
    h->next=NULL;
    do
    {
        printf("\n 1. Initialization \n 2. Insert first \n 3. Insert random \n 4. Insert last \n 5. Display");
        printf("\n 6. Delete first \n 7. Delete random \n 8. Delete last \n 9. Size \n 10. Exit");
        printf("\n Enter choice : ");
        scanf("%d",&ch);
        switch(ch)
        {
            case 1:
                h=init(h);
                break;
            case 2:
                printf("\n Enter element to insert : ");
                scanf("%d",&a);
                h=insertf(h,a);
                break;
            case 3:
                printf("\n Enter element to insert : ");
                scanf("%d",&b);
                printf("\n Enter position to insert at : ");
                scanf("%d",&p);
                h=insertr(h,b,p);
                break;
            case 4:
                printf("\n Enter element to insert : ");

```

```

        scanf("%d",&c);
        h=insertl(h,c);
        break;
    case 5:
        display(h);
        break;
    case 6:
        h=deletef(h);
        break;
    case 7:
        printf("\n Enter position to delete from : ");
        scanf("%d",&l);
        h=deleter(h,l);
        break;
    case 8:
        h=deletel(h);
        break;
    case 9:
        printf("\n Size of linked list : %d",h->hdata);
        break;
    case 10:
        printf("\n Exiting");
        break;
    default:
        printf("\n Wrong choice");

    }
}while(ch!=10);
}

head * init(head * h)
{
    h->next=NULL;
    h->hdata=0;
    return h;
}

node * createnode(int x)
{
    node *temp;
    temp=(node *)malloc(sizeof(node));
    temp->data=x;
    temp->next=NULL;
    return temp;
}

head * insertf(head *h,int x)
{
    node *temp;
    temp=createnode(x);
    h->hdata++;
    if(h->next==NULL)

```

```

h->next=temp;
else
{
    temp->next=h->next;
    h->next=temp;
}
return h;
}

head * insertl(head *h,int x)
{
    node *temp,*p;
    temp=createnode(x);
    h->hdata++;
    if(h->next==NULL)
        h->next=temp;
    else
    {
        for(p=h->next;p->next!=NULL;p=p->next);
        p->next=temp;
    }
    return h;
}

head * insertr(head *h,int x,int pos)
{
    int s=h->hdata;
    node *temp,*p;
    int i;
    if(pos<1 || pos>(s+1))
    {
        printf("\n Invalid position");
        return h;
    }
    else if(pos==1)
    {
        h=insertf(h,x);
        return h;
    }
    else if(pos==(s+1))
    {
        h=insertl(h,x);
        return h;
    }
    else
    {
        h->hdata++;
        temp=createnode(x);
        for(i=1,p=h->next;i<(pos-1);i++,p=p->next);
        temp->next=p->next;
        p->next=temp;
    }
}

```

```

        return h;
    }
}

head * deletef(head *h)
{
    node *p;
    if(h->next==NULL)
    {
        printf("\n Linked list is empty");
        return h;
    }
    else
    {
        h->hdata--;
        p=h->next;
        h->next=p->next;
        printf("\n Deleted element = %d",p->data);
        free(p);
        return h;
    }
}
head * deletel(head *h)
{
    int s=h->hdata;
    node *p,*t;
    if(h->next==NULL)
    {
        printf("\n Linked list is empty");
        return h;
    }
    else if(s==1)
    {
        h=deletef(h);
        return h;
    }
    else
    {
        h->hdata--;
        for(p=h->next;p->next!=NULL;t=p,p=p->next);
        t->next=NULL;
        printf("\n Deleted element = %d",p->data);
        free(p);
        return h;
    }
}
head * deleter(head *h,int pos)
{
    int s=h->hdata;
    node *t,*p;

```

```

int i;
if(h->next==NULL)
{
    printf("\n Linked list is empty");
    return h;
}
else if(pos<1 || pos>s)
{
    printf("\n Invalid position");
    return h;
}
else if(pos==1)
{
    h=deletef(h);
    return h;
}
else if(pos==s)
{
    h=deletel(h);
    return h;
}
else
{
    h->hdata--;
    for(p=h->next,i=1;i<(pos-1);i++,p=p->next);
    t=p->next;
    p->next=t->next;
    printf("\n Deleted element = %d",t->data);
    free(t);
    return h;
}
}

void display(head *h)
{
    node *p;
    if(h->next==NULL)
        printf("\n Linked list is empty");
    else
    {
        printf("\n Linked list : \n");
        for(p=h->next;p!=NULL;p=p->next)
            printf(" %d ",p->data);
    }
}

```

**OUTPUT-**

```

# EDS LABView
1. Initialization
2. Insert first
3. Insert random
4. Insert last
5. Display
6. Delete first
7. Delete random
8. Delete last
9. Size
10. Exit
Enter choice : 2
Enter element to insert : 1
1. Initialization
2. Insert first
3. Insert random
4. Insert last
5. Display
6. Delete first
7. Delete random
8. Delete last
9. Size
10. Exit
Enter choice : 4
Enter element to insert : 5
1. Initialization
2. Insert first
3. Insert random
4. Insert last
5. Display
6. Delete first
7. Delete random
8. Delete last
9. Size
10. Exit
Enter choice : 3
Enter element to insert : 3
Enter position to insert at : 3
1. Initialization
2. Insert first
3. Insert random
4. Insert last
5. Display
6. Delete first
7. Delete random
8. Delete last
9. Size
10. Exit
Enter choice : 5
Enter choice : 5
1. Initialization
2. Insert first
3. Insert random
4. Insert last
5. Display
6. Delete first
7. Delete random
8. Delete last
9. Size
10. Exit
Enter choice : 9
Size of linked list : 3
1. Initialization
2. Insert first
3. Insert random
4. Insert last
5. Display
6. Delete first
7. Delete random
8. Delete last
9. Size
10. Exit
Enter choice : 10
Exiting
Process exited after 54.28 seconds with return value 10
Press any key to continue . .

```

### Q37. WAP for Generate L.L. with Insertion, Delete and Display.

Ans37.

#### C PROGRAM –

```

#include<stdio.h>
#include<stdlib.h>
struct node
{
    void *data;
    struct node *next;
    int type;
};
typedef struct node node;
node *s=NULL;
void init();
void insertf();
void insertl();
void insertr(int);
void deletef();
void deletel();
void deleter(int);
void display();
int size();
void main()

```

```

{
int ch,p,l;
do
{
printf("\n 1. Initialization \n 2. Insert first \n 3. Insert last \n 4. Insert random \n 5. Delete first");
printf("\n 6. Delete last \n 7. Delete random \n 8. Display \n 9. Size \n 10. Exit");
printf("\n Enter choice : ");
scanf("%d",&ch);
switch(ch)
{
    case 1:
        init();
        break;
    case 2:
        insertf();
        break;
    case 3:
        insertl();
        break;
    case 4:
        printf("\n Enter position to insert : ");
        scanf("%d",&p);
        insertr(p);
        break;
    case 5:
        deletef();
        break;
    case 6:
        deletel();
        break;
    case 7:
        printf("\n Enter position to delete : ");
        scanf("%d",&l);
        deleter(l);
        break;
    case 8:
        display();
        break;
    case 9:
        if(size())
            printf("\n Size of generic linked list : %d",size());
        else
            printf("\n Size of generic linked list : 0");
        break;
    case 10:
        printf("\n Exiting");
        break;
default:
    printf("\n Wrong choice");
}
}

```

```

        }
    }while(ch!=10);
}

void init()
{
    s=NULL;
}

int size()
{
    node *p=s,*temp;
    int i=0;
    if(s==NULL)
        return i;
    else
    {
        while(p!=NULL)
        {
            i++;
            p=p->next;
        }
        return i;
    }
}

void insertf()
{
    int type,x;
    char y;
    float z;
    node *temp;
    printf("\n 1. Integer \n 2. Character \n 3. Float");
    printf("\n Choose type of data you want to insert : ");
    scanf("%d",&type);
    temp=(node *)malloc(sizeof(node));
    temp->type=type;
    temp->next=NULL;
    switch(type)
    {
        case 1:
            scanf("%d",&x);
            temp->data=malloc(sizeof(int));
            *(int *)temp->data=x;
            break;
        case 2:
            fflush(stdin);
            scanf("%c",&y);
            temp->data=malloc(sizeof(char));
            *(char *)temp->data=y;
            break;
        case 3:
}

```

```

        scanf("%f",&z);
        temp->data=malloc(sizeof(float));
        *(float *)temp->data=z;
        break;
    default:
        printf("\n Wrong choice");
    }
    if(s==NULL)
    s=temp;
    else
    {
        temp->next=s;
        s=temp;
    }
}
void insertl()
{
    int type,x;
    char y;
    float z;
    node *temp,*p;
    printf("\n 1. Integer \n 2. Character \n 3. Float");
    printf("\n Choose type of data you want to insert : ");
    scanf("%d",&type);
    temp=(node *)malloc(sizeof(node));
    temp->type=type;
    temp->next=NULL;
    switch(type)
    {
        case 1:
            scanf("%d",&x);
            temp->data=malloc(sizeof(int));
            *(int *)temp->data=x;
            break;
        case 2:
            fflush(stdin);
            scanf("%c",&y);
            temp->data=malloc(sizeof(char));
            *(char *)temp->data=y;
            break;
        case 3:
            scanf("%f",&z);
            temp->data=malloc(sizeof(float));
            *(float *)temp->data=z;
            break;
    default:
        printf("\n Wrong choice");
    }
    if(s==NULL)

```

```

s=temp;
else
{
    for(p=s;p->next!=NULL;p=p->next);
    p->next=temp;
}
}

void insertr(int pos)
{
    int type,x,i,l=size();
    char y;
    float z;
    node *temp,*p;
    if(pos<1 || pos>(l+1))
        printf("\n Invalid position");
    if(s==NULL)
        s=temp;
    else if(pos==1)
    {
        insertf();
        return;
    }
    else if(pos==l+1)
    {
        insertl();
        return;
    }
    else
    {
        printf("\n 1. Integer \n 2. Character \n 3. Float");
        printf("\n Choose type of data you want to insert : ");
        scanf("%d",&type);
        temp=(node *)malloc(sizeof(node));
        temp->type=type;
        temp->next=NULL;
        switch(type)
        {
            case 1:
                scanf("%d",&x);
                temp->data=malloc(sizeof(int));
                *(int *)temp->data=x;
                break;
            case 2:
                fflush(stdin);
                scanf("%c",&y);
                temp->data=malloc(sizeof(char));
                *(char *)temp->data=y;
                break;
            case 3:

```

```

        scanf("%f",&z);
        temp->data=malloc(sizeof(float));
        *(float *)temp->data=z;
        break;
    default:
        printf("\n Wrong choice");
    }
    for(p=s,i=1;i<(pos-1);i++,p=p->next);
    temp->next=p->next;
    p->next=temp;
}
}

void display()
{
    node *p;
    if(s==NULL)
        printf("\n Generic linked list is empty");
    else
    {
        for(p=s;p!=NULL;p=p->next)
        {
            if(p->type==1)
                printf(" %d ",*(int *)p->data);
            else if(p->type==2)
                printf(" %c ",*(char *)p->data);
            else
                printf(" %f ",*(float *)p->data);
        }
    }
}

void deletef()
{
    node *p=s;
    if(s==NULL)
        printf("\n Generic linked list is empty");
    else
    {
        s=s->next;
        if(p->type==1)
            printf("\n Deleted element : %d",*(int *)p->data);
        else if(p->type==2)
            printf("\n Deleted element : %c",*(char *)p->data);
        else
            printf("\n Deleted element : %f",*(float *)p->data);
        free(p);
    }
}

void deletel()
{

```

```

node *p=s,*t;
if(s==NULL)
printf("\n Generic linked list is empty");
else if(size()==1)
{
    deletef();
    return;
}
else
{
    for(p=s;p->next!=NULL;t=p,p=p->next);
    t->next=NULL;
    if(p->type==1)
        printf("\n Deleted element : %d",*(int *)p->data);
    else if(p->type==2)
        printf("\n Deleted element : %c",*(char *)p->data);
    else
        printf("\n Deleted element : %f",*(float *)p->data);
    free(p);
}
}

void deleter(int pos)
{
    node *p=s,*t;
    int l=size(),i;
    if(s==NULL)
        printf("\n Generic linked list is empty");
    else if(pos<1 || pos>l)
        printf("\n Invalid position");
    else if(pos==1)
    {
        deletef();
        return;
    }
    else if(pos==l)
    {
        deletel();
        return;
    }
    else
    {
        for(p=s,i=1;i<(pos-1);i++,p=p->next);
        t=p->next;
        p->next=t->next;
        if(t->type==1)
            printf("\n Deleted element : %d",*(int *)t->data);
        else if(t->type==2)
            printf("\n Deleted element : %c",*(char *)t->data);
    }
}

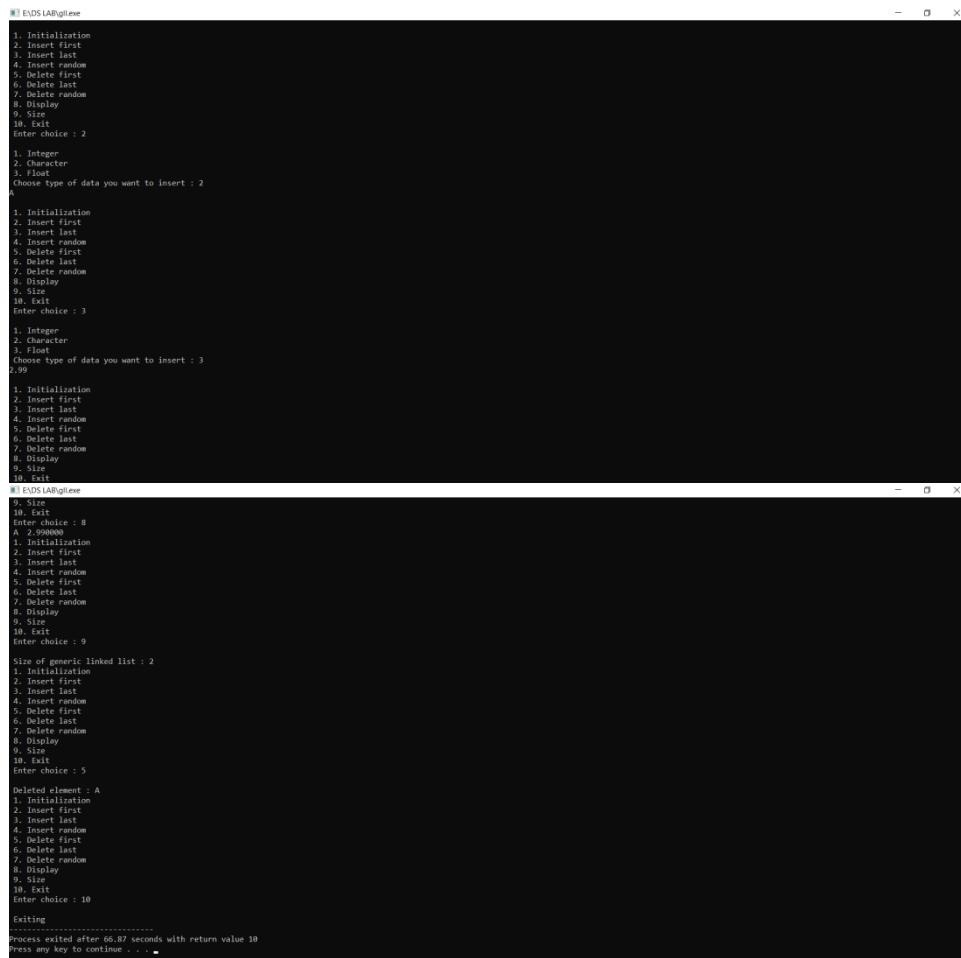
```

```

        printf("\n Deleted element : %f",*(float *)t->data);
        free(t);
    }
}

```

## OUTPUT-



```

E:\DS\LAB\q1.c
1. Initialization
2. Insert first
3. Insert last
4. Insert random
5. Delete first
6. Delete last
7. Delete random
8. Display
9. Size
10. Exit
Enter choice : 2

1. Integer
2. Character
3. Float
Choose type of data you want to insert : 2

1. Initialization
2. Insert first
3. Insert last
4. Insert random
5. Delete first
6. Delete last
7. Delete random
8. Display
9. Size
10. Exit
Enter choice : 3

1. Integer
2. Character
3. Float
Choose type of data you want to insert : 3
2.99

1. Initialization
2. Insert first
3. Insert last
4. Insert random
5. Delete first
6. Delete last
7. Delete random
8. Display
9. Size
10. Exit
Enter choice : 9

1. Size
10. Exit
Enter choice : 8
A
1. Insertion
2. Insert first
3. Insert last
4. Insert random
5. Delete first
6. Delete last
7. Delete random
8. Display
9. Size
10. Exit
Enter choice : 5

Deleted element : A
1. Initialization
2. Insert first
3. Insert last
4. Insert random
5. Delete first
6. Delete last
7. Delete random
8. Display
9. Size
10. Exit
Enter choice : 10

Exiting -----
Success exited after 06.87 seconds with return value 10
Press any key to continue . . .

```

## Q38. WAP for ,Polynomial representation, Polynomial Operations and Display.

Ans38.

### C PROGRAM –

```

#include<stdio.h>
#include<stdlib.h>

struct poly
{
    int co;
    int ex;
    struct poly *next;
};

typedef struct poly poly;
poly * create(poly *);
poly * addnode(poly *,int,int);
poly * addpoly(poly *,poly *, poly *);
poly * subpoly(poly *,poly *, poly *);
poly * mulpoly(poly *,poly *, poly *);
void display(poly *);

```

```

void main()
{
    int ch;
    poly *p1=NULL, *p2=NULL, *add=NULL, *sub=NULL, *mul=NULL;
    do
    {
        printf("\n 1. Creation \n 2. Addition \n 3. Subtraction \n 4. Multiplication \n 5. Exit");
        printf("\n Enter your choice : ");
        scanf("%d",&ch);
        switch(ch)
        {
            case 1:
                printf("\n ***** CREATING POLYNOMIAL 1 ***** : \n");
                p1=create(p1);
                display(p1);
                printf("\n ***** CREATING POLYNOMIAL 2 ***** : \n");
                p2=create(p2);
                display(p2);
                break;
            case 2:
                add=addpoly(add,p1,p2);
                display(add);
                break;
            case 3:
                sub=subpoly(sub,p1,p2);
                display(sub);
                break;
            case 4:
                mul=mulpoly(mul,p1,p2);
                display(mul);
                break;
            case 5:
                printf("\n Exiting");
                break;
            default:
                printf("\n Wrong choice");
        }
    }while(ch!=5);
}

poly * addnode(poly *p, int co, int ex)
{
    poly *temp,*q;
    temp=(poly *)malloc(sizeof(poly));
    temp->co=co;
    temp->ex=ex;
    temp->next=NULL;
    if(p==NULL)
        p=temp;
    else

```

```

{
    for(q=p;q->next!=NULL;q=q->next);
        q->next=temp;
}
return p;
}

poly * create(poly *p)
{
    int ex,co;
    do
    {
        printf("\n Enter coeffecient and exponent of polynomial : ");
        scanf("%d %d",&co,&ex);
        p=addnode(p,co,ex);
    }while(ex!=0);
    return p;
}

poly * addpoly(poly *add, poly *p1, poly *p2)
{
    poly *p,*q;
    p=p1;
    q=p2;
    while(p!=NULL && q!=NULL)
    {
        if(p->ex > q->ex)
        {
            add=addnode(add,p->co,p->ex);
            p=p->next;
        }
        else if(q->ex > p->ex)
        {
            add=addnode(add,q->co,q->ex);
            q=q->next;
        }
        else
        {
            add=addnode(add,(p->co + q->co),p->ex);
            p=p->next;
            q=q->next;
        }
    }
    while(p!=NULL)
    {
        add=addnode(add,p->co,p->ex);
        p=p->next;
    }
    while(q!=NULL)
    {
        add=addnode(add,q->co,q->ex);
    }
}

```

```

        q=q->next;
    }
    return add;
}

poly * subpoly(poly *sub, poly *p1, poly *p2)
{
    poly *p,*q;
    p=p1;
    q=p2;
    while(p!=NULL && q!=NULL)
    {
        if(p->ex > q->ex)
        {
            sub=addnode(sub,p->co,p->ex);
            p=p->next;
        }
        else if(q->ex > p->ex)
        {
            sub=addnode(sub,-(q->co),q->ex);
            q=q->next;
        }
        else
        {
            sub=addnode(sub,(p->co - q->co),p->ex);
            p=p->next;
            q=q->next;
        }
    }
    while(p!=NULL)
    {
        sub=addnode(sub,p->co,p->ex);
        p=p->next;
    }
    while(q!=NULL)
    {
        sub=addnode(sub,q->co,q->ex);
        q=q->next;
    }
    return sub;
}

poly * mulpoly(poly *mul, poly *p1, poly *p2)
{
    poly *p,*q;
    for(p=p1;p!=NULL;p=p->next)
    for(q=p2;q!=NULL;q=q->next)
        mul=addnode(mul,(p->co * q->co),(p->ex + q->ex));
    return mul;
}

void display(poly *p)

```

```

{
poly *t;
if(p==NULL)
printf("\n Polynomial is empty");
else
{
    printf("\n Required Polynomial : \n");
    for(t=p;t->next!=NULL;t=t->next)
        printf(" (%d x^%d) + ",t->co,t->ex);
    printf(" (%d x^%d) ",t->co,t->ex);
}
}

```

### OUTPUT-

```

E:\DS LAB\poly1.exe
1. Creation
2. Addition
3. Subtraction
4. Multiplication
5. Exit
Enter your choice : 1

***** CREATING POLYNOMIAL 1 *****
Enter coefficient and exponent of polynomial : 4
2
Enter coefficient and exponent of polynomial : -5
1
Enter coefficient and exponent of polynomial : 1
0
Required Polynomial :
(4 x^2) + (-5 x^1) + (1 x^0)

***** CREATING POLYNOMIAL 2 *****
Enter coefficient and exponent of polynomial : -2
3
Enter coefficient and exponent of polynomial : 1
2
Enter coefficient and exponent of polynomial : -7
1
Enter coefficient and exponent of polynomial : 5
0
Required Polynomial :
(-2 x^3) + (1 x^2) + (-7 x^1) + (5 x^0)

E:\DS LAB\poly1.exe
0
Required Polynomial :
(-2 x^3) + (1 x^2) + (-7 x^1) + (5 x^0)
1. Creation
2. Addition
3. Subtraction
4. Multiplication
5. Exit
Enter your choice : 2

Required Polynomial :
(2 x^3) + (5 x^2) + (-12 x^1) + (6 x^0)
1. Creation
2. Addition
3. Subtraction
4. Multiplication
5. Exit
Enter your choice : 3

Required Polynomial :
(2 x^3) + (3 x^2) + (2 x^1) + (-4 x^0)
1. Creation
2. Addition
3. Subtraction
4. Multiplication
5. Exit
Enter your choice : 4

Required Polynomial :
(8 x^5) + (4 x^4) + (-28 x^3) + (10 x^2) + (-5 x^3) + (35 x^2) + (-25 x^1) + (-2 x^3) + (1 x^2) + (-7 x^1) + (5 x^0)
1. Creation
2. Addition
3. Subtraction
4. Multiplication
5. Exit
Enter your choice : 5

Exiting
Process exited after 07.2 seconds with return value 5
Press any key to continue . . .

```

### Q39. WAP for Sparse Matrix representation and Display.

Ans39.

#### C PROGRAM –

#### ARRAY REPRESENTATION –

```

#include<stdio.h>
void main()
{
    int i,j,m[10][10],r,c,count=0,size=0;
    printf("\n Enter number of rows in matrix : ");
    scanf("%d",&r);

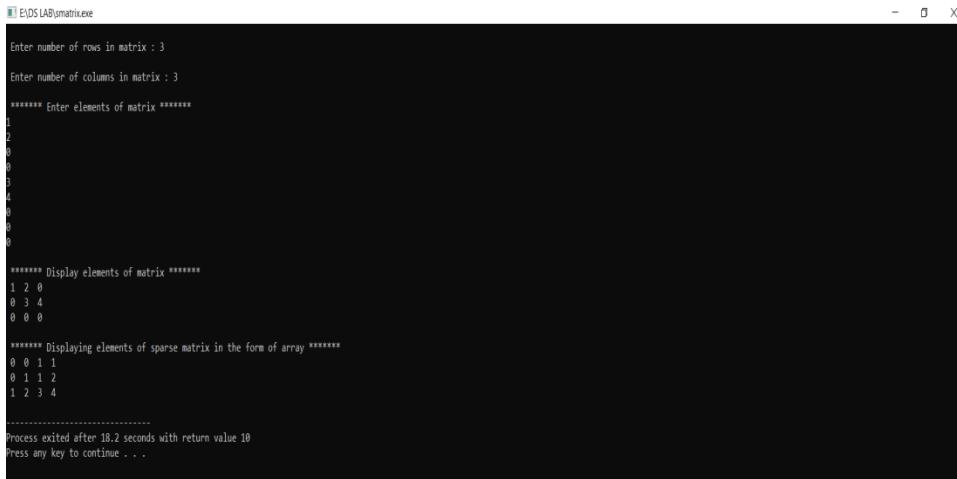
```

```

printf("\n Enter number of columns in matrix : ");
scanf("%d",&c);
do
{
    count=0;
    printf("\n ***** Enter elements of matrix ***** \n");
for(i=0;i<r;i++)
    for(j=0;j<c;j++)
        scanf("%d",&m[i][j]);
    for(i=0;i<r;i++)
        for(j=0;j<c;j++)
    {
        if(m[i][j]==0)
            count++;
    }
}while(count<=((r*c))/2);
printf("\n ***** Display elements of matrix ***** \n");
for(i=0;i<r;i++)
{
    for(j=0;j<c;j++)
        printf(" %d ",m[i][j]);
    printf("\n");
}
for(i=0;i<r;i++)
for(j=0;j<c;j++)
{
    if(m[i][j]!=0)
        size++;
}
int k=0,sm[3][size];
for(i=0;i<r;i++)
for(j=0;j<c;j++)
{
    if(m[i][j]!=0)
    {
        sm[0][k]=i;
        sm[1][k]=j;
        sm[2][k]=m[i][j];
        k++;
    }
}
printf("\n ***** Displaying elements of sparse matrix in the form of array ***** \n");
for(i=0;i<3;i++)
{
    for(j=0;j<size;j++)
        printf(" %d ",sm[i][j]);
    printf("\n");
}
}

```

## OUTPUT-



```
E:\DS LAB\matrix.exe
Enter number of rows in matrix : 3
Enter number of columns in matrix : 3
***** Enter elements of matrix *****
1
2
0
0
3
0
0
0
0

***** Display elements of matrix *****
1 2 0
0 3 4
0 0 0

***** Displaying elements of sparse matrix in the form of array *****
0 0 1 1
0 1 1 2
1 2 3 4

Process exited after 18.2 seconds with return value 10
Press any key to continue . . .
```

## LINKED LIST REPRESENTATION –

```
#include<stdio.h>
#include<stdlib.h>
#include<malloc.h>
struct smatrix
{
    int r;
    int c;
    int v;
    struct smatrix *next;
};

typedef struct smatrix smatrix;
smatrix * addnode(smatrix *,int,int,int);
void display(smatrix *);
void main()
{
    smatrix *sm=NULL;
    int i,j,m[10][10],r,c,count=0;
    printf("\n Enter number of rows in matrix : ");
    scanf("%d",&r);
    printf("\n Enter number of columns in matrix : ");
    scanf("%d",&c);
    do
    {
        count=0;
        printf("\n ***** Enter elements of matrix ***** \n");
        for(i=0;i<r;i++)
            for(j=0;j<c;j++)
                scanf("%d",&m[i][j]);
        for(i=0;i<r;i++)
            for(j=0;j<c;j++)
            {
                if(m[i][j]==0)
                    count++;
            }
    }
```

```

}while(count<=((r*c))/2);
printf("\n ***** Display elements of matrix ***** \n");
for(i=0;i<r;i++)
{
    for(j=0;j<c;j++)
        printf(" %d ",m[i][j]);
    printf("\n");
}
for(i=0;i<r;i++)
    for(j=0;j<c;j++)
    {
        if(m[i][j]!=0)
            sm=addnode(sm,i,j,m[i][j]);
    }
    display(sm);
}
smatrix * addnode(smatrix *sm,int x,int y,int z)
{
    smatrix *temp;
    smatrix *p;
    temp=(smatrix *)malloc(sizeof(smatrix));
    temp->r=x;
    temp->c=y;
    temp->v=z;
    temp->next=NULL;
    if(sm==NULL)
        sm=temp;
    else
    {
        for(p=sm;p->next!=NULL;p=p->next);
        p->next=temp;
    }
    return sm;
}
void display(smatrix *sm)
{
    smatrix *p;
    if(sm==NULL)
        printf("\n ***** Sparse matrix is empty *****");
    else
    {
        printf("\n ***** Sparse matrix in linked list form *****\n In the format row no. col no. value -> is
given below\n");
        for(p=sm;p->next!=NULL;p=p->next)
            printf(" %d %d %d ->",p->r,p->c,p->v);
            printf(" %d %d %d ",p->r,p->c,p->v);
    }
}

```

### OUTPUT-

```
FILEDS LAB\matrixll.exe
Enter number of rows in matrix : 3
Enter number of columns in matrix : 3
***** Enter elements of matrix *****
1
2
3
0
0
1
0
0
0
***** Display elements of matrix *****
1 2 3
0 0 4
0 0 0
***** Sparse matrix in linked list form *****
In the format row no. col no. value -> is given below
0 1 -> 0 1 2 -> 0 2 3 -> 1 2 4
-----
Process exited after 15.9 seconds with return value 7
Press any key to continue . . .
```

## PROGRAMS BASED ON BINARY SEARCH TREE

---

**Q40. WAP for implementing Binary Search with all basic operations and all no recursive Traversals.**

Ans40.

**C PROGRAM –**

```
#include<stdio.h>
#include<malloc.h>
#define max 100
struct bst
{
    int data;
    struct bst *lc;
    struct bst *rc;
};
typedef struct bst BST;
BST * init(BST *);
BST * insert(BST *,int);
void search(BST *,int);
void nrinorder(BST *);
void nrpreorder(BST *);
void nrpostorder(BST *);
void nrlevelorder(BST *);
BST * nrdelete(BST *,int);
void main()
{
    BST *R=NULL;
    int n,i=0,x,ch,op;
    do
    {
        printf("\n ***** NON-RECURSIVE PROGRAM *****\n");
        printf("\n 1. INITIALIZATION \n 2. INSERT \n 3. DISPLAY \n 4. DELETE \n 5. SEARCH \n 0. EXIT");
        printf("\n Enter your choice : ");
        scanf("%d",&ch);
```

```

switch(ch)
{
    case 1:
        R=init(R);
        break;
    case 2:
        printf("\n Enter total no. of elements in the binary tree : ");
        scanf("%d",&n);
        i=0;
        while(i<n)
        {
            i++;
            printf("\n Enter element to insert : ");
            scanf("%d",&x);
            R=insert(R,x);
        }
        break;
    case 3:
        do
        {
            printf("\n 1. INORDER DISPLAY \n 2. PREORDER DISPLAY \n 3.
POSTORDER DISPLAY \n 4. LEVELORDER DISPLAY \n 0. EXIT");
            printf("\n Enter your choice : ");
            scanf("%d",&op);
            switch(op)
            {
                case 1:
                    if(R==NULL)
                        printf("\n Binary tree is empty");
                    else
                        nrinorder(R);
                    break;
                case 2:
                    if(R==NULL)
                        printf("\n Binary tree is empty");
                    else
                        nrpreorder(R);
                    break;
                case 3:
                    if(R==NULL)
                        printf("\n Binary tree is empty");
                    else
                        nrpostorder(R);
                    break;
                case 4:
                    if(R==NULL)
                        printf("\n Binary tree is empty");
                    else
                        nrlevelorder(R);
            }
        }
    }
}

```

```

        break;
    case 0:
        break;
    default:
        printf("\n Wrong choice");
    }
}while(op!=0);
break;

case 4:
if(R==NULL)
    printf("\n Deletion not possible as binary tree is empty");
else
{
    printf("\n Enter element to be deleted : ");
    scanf("%d",&x);
    R=nrdelete(R,x);
}
break;

case 5:
if(R==NULL)
    printf("\n Search not possible as binary tree is empty");
else
{
    printf("\n Enter element to search : ");
    scanf("%d",&x);
    search(R,x);
}
break;

case 0:
printf("\n Exiting");
break;
default:
printf("\n Wrong choice");
}

}while(ch!=0);
}

BST * init(BST *R)
{
    R=NULL;
    return R;
}

BST * insert(BST *R,int x)
{
    BST *temp,*p,*t;
    temp=(BST *)malloc(sizeof(BST));
    temp->data=x;
    temp->lc=temp->rc=NULL;
    if(R==NULL)
        R=temp;
    else
    {
        p=R;
        while(p->rc!=NULL)
            p=p->rc;
        p->rc=temp;
    }
}

```

```

    else
    {
        t=R;
        while(t!=NULL)
        {
            p=t;
            if(x<t->data)
                t=t->lc;
            else if(x>t->data)
                t=t->rc;
            else
            {
                printf("\n Duplicate information");
                return R;
            }
        }
        if(p->data>x)
            p->lc=temp;
        else
            p->rc=temp;
    }
    return R;
}
void search(BST *R,int x)
{
    BST *t;
    t=R;
    while(t!=NULL)
    {
        if(x<t->data)
            t=t->lc;
        else if (x>t->data)
            t=t->rc;
        else
        {
            printf("\n Information found");
            return;
        }
    }
    printf("\n Information not found");
}
void nrpreorder(BST *R)
{
    BST *stk[10],*t;
    int top=-1;
    t=R;
    do
    {
        while(t!=NULL)

```

```

{
    printf(" %d ",t->data);
    stk[++top]=t;
    t=t->lc;
}
t=stk[top--];
t=t->rc;
}while(t!=NULL || top>-1);
}

void nrinorder(BST *R)
{
    BST *stk[10],*t;
    int top=-1;
    t=R;
    do
    {
        while(t!=NULL)
        {
            stk[++top]=t;
            t=t->lc;
        }
        t=stk[top--];
        printf(" %d ",t->data);
        t=t->rc;
    }while(t!=NULL || top>-1);
}

void nrlevelorder(BST *R)
{
    BST *queue[max],*t;
    int f=0,r=-1;
    t=R;
    queue[++r]=t;
    while(f<=r)
    {
        t=queue[f++];
        if(t!=NULL)
        {
            printf(" %d ",t->data);
            queue[++r]=t->lc;
            queue[++r]=t->rc;
        }
    }
}

void nrpostorder(BST *R)
{
    BST *stk1[max],*stk2[max],*t;
    int t1=-1,t2=-1;
    stk1[++t1]=R;
    while(t1!=-1)

```

```

{
    R=stk1[t1--];
    stk2[++t2]=R;
    if(R->lc)
        stk1[++t1]=R->lc;
    if(R->rc)
        stk1[++t1]=R->rc;
}
while(t2!=-1)
printf(" %d ",stk2[t2--]->data);
}

BST * nrdelete(BST *R, int x)
{
    BST *t,*p,*q,*t1,*t2,*t3;
    if(R==NULL)
    {
        printf("\n Deletion not possible as binary tree is empty");
        return R;
    }
    else
    {
        t=R;
        while((t!=NULL) && (x!=t->data))
        {
            p=t;
            if(x<t->data)
                t=t->lc;
            else if(x>t->data)
                t=t->rc;
        }
    }
    if(t==NULL)
    {
        printf("\n Element not found");
        return R;
    }
    else if(t==R)
    {
        p=t1=t->lc;
        t2=t->rc;
        while(t1->rc!=NULL)
            t1=t1->rc;
        t1->rc=t2;
        R=p;
        free(t);
        return R;
    }
}

```

```

if((t->lc==NULL)&&(t->rc==NULL))
{
    if(t->data<p->data)
        p->lc=NULL;
    else
        p->rc=NULL;
}
else if((t->lc!=NULL)&&(t->rc==NULL))
{
    if(t->data<p->data)
        p->lc=t->lc;
    else
        p->rc=t->lc;
}
else if((t->lc==NULL)&&(t->rc!=NULL))
{
    if(t->data<p->data)
        p->lc=t->rc;
    else
        p->rc=t->rc;
}
else
{
    t1=t->lc;
    t2=t->rc;
    while(t1->rc!=NULL)
        t1=t1->rc;
    t1->rc=t2;
    if(t->data<p->data)
        p->lc=t->lc;
    else
        p->rc=t->lc;
}
}
free(t);
return R;
}

```

**OUTPUT –**

```

E:\DS LAB\bin\Debug E:\DS LAB\bin\Debug
***** NON-RECURSIVE PROGRAM *****
1. INITIALIZATION
2. INSERT
3. DISPLAY
4. DELETE
5. SEARCH
6. EXIT
Enter your choice : 2
Enter total no. of elements in the binary tree : 6
Enter element to insert : 30
Enter element to insert : 25
Enter element to insert : 16
Enter element to insert : 28
Enter element to insert : 25
Enter element to insert : 67
***** NON-RECURSIVE PROGRAM *****
1. INITIALIZATION
2. INSERT
3. DISPLAY
4. DELETE
5. SEARCH
6. EXIT
Enter your choice : 3
1. INORDER DISPLAY
2. PREORDER DISPLAY
3. POSTORDER DISPLAY
4. LEVELORDER DISPLAY
0. EXIT
Enter your choice : 4
25 75 16 28 67
1. INORDER DISPLAY
2. PREORDER DISPLAY
3. POSTORDER DISPLAY
4. LEVELORDER DISPLAY
0. EXIT
Enter your choice : 0
***** NON-RECURSIVE PROGRAM *****
1. INITIALIZATION
2. INSERT
3. DISPLAY
4. DELETE
5. SEARCH
6. EXIT
Enter your choice : 4
Enter element to be deleted : 50
***** NON-RECURSIVE PROGRAM *****
1. INITIALIZATION
2. INSERT
3. DISPLAY
4. DELETE
5. SEARCH
6. EXIT
Enter your choice : 3
1. INORDER DISPLAY
2. PREORDER DISPLAY
3. POSTORDER DISPLAY
4. LEVELORDER DISPLAY
0. EXIT
Enter your choice : 1
1. INORDER DISPLAY
2. PREORDER DISPLAY
3. POSTORDER DISPLAY
4. LEVELORDER DISPLAY
0. EXIT
Enter your choice : 0
***** NON-RECURSIVE PROGRAM *****
1. INITIALIZATION
2. INSERT
3. DISPLAY
4. DELETE
5. SEARCH
6. EXIT
Enter your choice : 0
Exiting .....
Process exited after 39.22 seconds with return value 0
Press any key to continue . .

```

## Q41. WAP for Binary Search Create and all recursive Traversals.

Ans41.

### C PROGRAM –

```

#include<stdio.h>
#include<malloc.h>
struct bst
{
    int data;
    struct bst *lc;
    struct bst *rc;
};
typedef struct bst BST;
BST * init(BST *);
BST * insert(BST *,int);
BST * search(BST *,int);
void inorder(BST *);
void preorder(BST *);
void postorder(BST *);
void levelorder(BST *);
BST * rdelete(BST *,int);
BST * MinValueNode(BST *);
int height_of_tree(BST *);
void currentlevel(BST *,int);

```

```

void main()
{
    BST *R=NULL,*t;
    int n,i=0,x,ch,op;
    do
    {
        printf("\n ***** RECURSIVE PROGRAM *****\n");
        printf("\n 1. INITIALIZATION \n 2. INSERT \n 3. DISPLAY \n 4. DELETE \n 5. SEARCH \n 0. EXIT");
        printf("\n Enter your choice : ");
        scanf("%d",&ch);
        switch(ch)
        {
            case 1:
                R=init(R);
                break;
            case 2:
                printf("\n Enter total no. of elements in the binary tree : ");
                scanf("%d",&n);
                i=0;
                while(i<n)
                {
                    i++;
                    printf("\n Enter element to insert : ");
                    scanf("%d",&x);
                    R=insert(R,x);
                }
                break;
            case 3:
                do
                {
                    printf("\n 1. INORDER DISPLAY \n 2. PREORDER DISPLAY \n 3. POSTORDER DISPLAY \n 4. LEVELORDER DISPLAY \n 0. EXIT");
                    printf("\n Enter your choice : ");
                    scanf("%d",&op);
                    switch(op)
                    {
                        case 1:
                            if(R==NULL)
                                printf("\n Binary tree is empty");
                            else
                                inorder(R);
                            break;
                        case 2:
                            if(R==NULL)
                                printf("\n Binary tree is empty");
                            else
                                preorder(R);
                            break;
                        case 3:

```

```

        if(R==NULL)
            printf("\n Binary tree is empty");
        else
            postorder(R);
            break;
    case 4:
        if(R==NULL)
            printf("\n Binary tree is empty");
        else
            levelorder(R);
            break;
    case 0:
        break;
    default:
        printf("\n Wrong choice");
    }
}while(op!=0);
break;
case 4:
if(R==NULL)
printf("\n Deletion not possible as binary tree is empty");
else
{
    printf("\n Enter element to be deleted : ");
    scanf("%d",&x);
    R=rdelete(R,x);
}
break;
case 5:
if(R==NULL)
printf("\n Search not possible as binary tree is empty");
else
{
    printf("\n Enter element to search : ");
    scanf("%d",&x);
    t=search(R,x);
    if(t!=NULL)
        printf("\n Information found");
    else
        printf("\n Information not found");
}
break;
case 0:
printf("\n Exiting");
break;
default:
printf("\n Wrong choice");
}
}while(ch!=0);

```

```

}

BST * init(BST *R)
{
    R=NULL;
    return R;
}

BST * insert(BST *R,int x)
{
    BST *temp;
    temp=(BST *)malloc(sizeof(BST));
    temp->data=x;
    temp->lc=temp->rc=NULL;
    if(R==NULL)
        R=temp;
    else
    {
        if(x<R->data)
            R->lc=insert(R->lc,x);
        else if(x>R->data)
            R->rc=insert(R->rc,x);
    }
    return R;
}

BST * search(BST *R,int x)
{
    BST *t;
    t=R;
    if(t!=NULL)
    {
        if(x<t->data)
            t=search(R->lc,x);
        else if (x>t->data)
            t=search(R->rc,x);
    }
    return t;
}

void preorder(BST *R)
{
    if(R!=NULL)
    {
        printf(" %d ",R->data);
        preorder(R->lc);
        preorder(R->rc);
    }
}

void inorder(BST *R)
{
    if(R!=NULL)
    {

```

```

inorder(R->lc);
printf(" %d ",R->data);
inorder(R->rc);
}
}
void postorder(BST *R)
{
    if(R!=NULL)
    {
        postorder(R->lc);
        postorder(R->rc);
        printf(" %d ",R->data);
    }
}
void levelorder(BST *R)
{
    int h=height_of_tree(R);
    int i;
    for(i=1;i<=h;i++)
        currentlevel(R,i);
}
void currentlevel(BST *R,int l)
{
    if(R==NULL)
        return;
    else if(l==1)
        printf(" %d ",R->data);
    else
    {
        currentlevel(R->lc,l-1);
        currentlevel(R->rc,l-1);
    }
}
int height_of_tree(BST *R)
{
    int lh,rh;
    if(R==NULL)
        return 0;
    else
    {
        lh=height_of_tree(R->lc);
        rh=height_of_tree(R->rc);
        if(lh>rh)
            return(lh+1);
        else
            return(rh+1);
    }
}
BST * MinValueNode(BST *R)

```

```

{
    BST *c=R;
    while(c && c->lc!=NULL)
        c=c->lc;
    return c;
}
BST * rdelete(BST *R,int x)
{
    BST *temp;
    if(R==NULL)
        return R;
    if(x<R->data)
        R->lc=rdelete(R->lc,x);
    else if(x>R->data)
        R->rc=rdelete(R->rc,x);
    else
    {
        if(R->lc==NULL && R->rc==NULL)
            return NULL;
        else if(R->lc==NULL)
        {
            temp=R->rc;
            free(R);
            return temp;
        }
        else if(R->rc==NULL)
        {
            temp=R->lc;
            free(R);
            return temp;
        }
        temp=MinValueNode(R->rc);
        R->data=temp->data;
        R->rc=rdelete(R->rc,temp->data);
    }
    return R;
}

```

**OUTPUT –**

```

E:\CDS LAB\bin\new.exe
***** RECURSIVE PROGRAM *****
1. INITIALIZATION
2. INSERT
3. DISPLAY
4. DELETE
5. SEARCH
6. EXIT
Enter your choice : 2
Enter total no. of elements in the binary tree : 6
Enter element to insert : 50
Enter element to insert : 25
Enter element to insert : 16
Enter element to insert : 28
Enter element to insert : 35
Enter element to insert : 67
***** RECURSIVE PROGRAM *****
1. INITIALIZATION
2. INSERT
3. DISPLAY
4. DELETE
5. SEARCH
6. EXIT
Enter your choice : 3
1. INORDER DISPLAY
2. PREORDER DISPLAY
3. POSTORDER DISPLAY
4. LEVELORDER DISPLAY
0. EXIT
Enter your choice : 1
16 25 28 50 67 75
***** RECURSIVE PROGRAM *****
1. INITIALIZATION
2. INSERT
3. DISPLAY
4. DELETE
5. SEARCH
6. EXIT
Enter your choice : 4
Enter element to be deleted : 50
***** RECURSIVE PROGRAM *****
1. INITIALIZATION
2. DISPLAY
3. DELETE
4. SEARCH
0. EXIT
Enter your choice : 3
1. INORDER DISPLAY
2. PREORDER DISPLAY
3. POSTORDER DISPLAY
4. LEVELORDER DISPLAY
0. EXIT
Enter your choice : 2
16 25 28 50 67 75
***** RECURSIVE PROGRAM *****
1. INITIALIZATION
2. DISPLAY
3. DELETE
4. SEARCH
0. EXIT
Enter your choice : 0
Exiting .....
Process exited after 25.54 seconds with return value 0
Press any key to continue . .

```

## Q42. WAP Binary Search Create and Traversals and other remaining operations.

Ans42.

**C PROGRAM –**

**NON-RECURSIVE –**

```

#include<stdio.h>
#include<malloc.h>
#define max 100
struct bst
{
    int data;
    struct bst *lc;
    struct bst *rc;
};
typedef struct bst BST;
BST * init(BST *);
BST * insert(BST *,int);
void search(BST *,int);
void nrinorder(BST *);
void nrpreorder(BST *);
void nrpostorder(BST *);
void nrlevelorder(BST *);
BST * nrdelete(BST *,int);
int height_of_tree(BST *);

```

```

int count_total(BST *);
int count_term(BST *);
int count_non_term(BST *);
int mindata(BST *);
int maxdata(BST *);
void mirror(BST *);
void cleartree(BST *);
void main()
{
    BST *R=NULL;
    int n,i=0,x,ch,op;
    do
    {
        printf("\n ***** NON-RECURSIVE PROGRAM *****\n");
        printf("\n 1. INITIALIZATION \n 2. INSERT \n 3. DISPLAY \n 4. DELETE \n 5. HEIGHT OR MAX
DEPTH \n 6. SEARCH");
        printf("\n 7. COUNT NODES \n 8. MINIMUM DATA \n 9. MAXIMUM DATA \n 10. MIRROR \n 11.
CLEAR TREE \n 0. EXIT");
        printf("\n Enter your choice : ");
        scanf("%d",&ch);
        switch(ch)
        {
            case 1:
                R=init(R);
                break;
            case 2:
                printf("\n Enter total no. of elements in the binary tree : ");
                scanf("%d",&n);
                i=0;
                while(i<n)
                {
                    i++;
                    printf("\n Enter element to insert : ");
                    scanf("%d",&x);
                    R=insert(R,x);
                }
                break;
            case 3:
                do
                {
                    printf("\n 1. INORDER DISPLAY \n 2. PREORDER DISPLAY \n 3.
POSTORDER DISPLAY \n 4. LEVELORDER DISPLAY \n 0. EXIT");
                    printf("\n Enter your choice : ");
                    scanf("%d",&op);
                    switch(op)
                    {
                        case 1:
                            if(R==NULL)
                                printf("\n Binary tree is empty");

```

```

        else
            nrinorder(R);
            break;
    case 2:
        if(R==NULL)
            printf("\n Binary tree is empty");
        else
            nrpreorder(R);
            break;
    case 3:
        if(R==NULL)
            printf("\n Binary tree is empty");
        else
            nrpostorder(R);
            break;
    case 4:
        if(R==NULL)
            printf("\n Binary tree is empty");
        else
            nrlevelorder(R);
            break;
    case 0:
        break;
    default:
        printf("\n Wrong choice");
    }
}
}while(op!=0);
break;
case 4:
if(R==NULL)
    printf("\n Deletion not possible as binary tree is empty");
else
{
    printf("\n Enter element to be deleted : ");
    scanf("%d",&x);
    R=nrdelete(R,x);
}
break;
case 5:
printf("\n Height or Max depth of tree : %d",height_of_tree(R));
break;
case 6:
if(R==NULL)
    printf("\n Search not possible as binary tree is empty");
else
{
    printf("\n Enter element to search : ");
    scanf("%d",&x);
    search(R,x);
}

```

```

    }
    break;
case 7:
    do
    {
        printf("\n 1. TOTAL NODES \n 2. TERMINAL NODES \n 3. NON-TERMINAL
NODES \n 0. EXIT");
        printf("\n Enter your choice : ");
        scanf("%d",&op);
        switch(op)
        {
            case 1:
                i=count_total(R);
                printf("\n Total no. of nodes : %d",i);
                break;
            case 2:
                i=count_term(R);
                printf("\n Total no. of terminal nodes : %d",i);
                break;
            case 3:
                i=count_non_term(R);
                printf("\n Total no. of non-terminal nodes : %d",i);
                break;
            case 0:
                break;
            default:
                printf("\n Wrong choice");
        }
    }while(op!=0);
    break;
case 8:
    if(R==NULL)
        printf("\n Binary tree is empty");
    else
    {
        i=mindata(R);
        printf("\n Minimum data : %d",i);
    }
    break;
case 9:
    if(R==NULL)
        printf("\n Binary tree is empty");
    else
    {
        i=maxdata(R);
        printf("\n Maximum data : %d",i);
    }
    break;
case 10:

```

```

        if(R==NULL)
            printf("\n Binary tree is empty");
        else
            mirror(R);
            break;
    case 11:
        if(R==NULL)
            printf("\n Binary tree is empty");
        else
            cleartree(R);
            R=NULL;
            break;
    case 0:
        printf("\n Exiting");
        break;
    default:
        printf("\n Wrong choice");
    }
}
}while(ch!=0);
}

BST * init(BST *R)
{
    R=NULL;
    return R;
}

BST * insert(BST *R,int x)
{
    BST *temp,*p,*t;
    temp=(BST *)malloc(sizeof(BST));
    temp->data=x;
    temp->lc=temp->rc=NULL;
    if(R==NULL)
        R=temp;
    else
    {
        t=R;
        while(t!=NULL)
        {
            p=t;
            if(x<t->data)
                t=t->lc;
            else if(x>t->data)
                t=t->rc;
            else
            {
                printf("\n Duplicate information");
                return R;
            }
        }
    }
}

```

```

        if(p->data>x)
            p->lc=temp;
        else
            p->rc=temp;
    }
    return R;
}
void search(BST *R,int x)
{
    BST *t;
    t=R;
    while(t!=NULL)
    {
        if(x<t->data)
            t=t->lc;
        else if (x>t->data)
            t=t->rc;
        else
        {
            printf("\n Information found");
            return;
        }
    }
    printf("\n Information not found");
}
void nrpreorder(BST *R)
{
    BST *stk[10],*t;
    int top=-1;
    t=R;
    do
    {
        while(t!=NULL)
        {
            printf(" %d ",t->data);
            stk[++top]=t;
            t=t->lc;
        }
        t=stk[top--];
        t=t->rc;
    }while(t!=NULL || top>-1);
}
void nrinorder(BST *R)
{
    BST *stk[10],*t;
    int top=-1;
    t=R;
    do
    {

```

```

        while(t!=NULL)
        {
            stk[++top]=t;
            t=t->lc;
        }
        t=stk[top--];
        printf(" %d ",t->data);
        t=t->rc;
    }while(t!=NULL || top>-1);
}
void nrlevelorder(BST *R)
{
    BST *queue[max],*t;
    int f=0,r=-1;
    t=R;
    queue[++r]=t;
    while(f<=r)
    {
        t=queue[f++];
        if(t!=NULL)
        {
            printf(" %d ",t->data);
            queue[++r]=t->lc;
            queue[++r]=t->rc;
        }
    }
}
void nrpostorder(BST *R)
{
    BST *stk1[max],*stk2[max],*t;
    int t1=-1,t2=-1;
    stk1[++t1]=R;
    while(t1!=-1)
    {
        R=stk1[t1--];
        stk2[++t2]=R;
        if(R->lc)
        stk1[++t1]=R->lc;
        if(R->rc)
        stk1[++t1]=R->rc;
    }
    while(t2!=-1)
    printf(" %d ",stk2[t2--]->data);
}
BST * nrdelete(BST *R, int x)
{
    BST *t,*p,*q,*t1,*t2,*t3;
    if(R==NULL)
    {

```

```

printf("\n Deletion not possible as binary tree is empty");
return R;
}
else
{
    t=R;
    while((t!=NULL) && (x!=t->data))
    {
        p=t;
        if(x<t->data)
            t=t->lc;
        else if(x>t->data)
            t=t->rc;
    }
}
if(t==NULL)
{
    printf("\n Element not found");
    return R;
}
else if(t==R)
{
    p=t1=t->lc;
    t2=t->rc;
    while(t1->rc!=NULL)
        t1=t1->rc;
    t1->rc=t2;
    R=p;
    free(t);
    return R;
}
else
{
    if((t->lc==NULL)&&(t->rc==NULL))
    {
        if(t->data<p->data)
            p->lc=NULL;
        else
            p->rc=NULL;
    }
    else if((t->lc!=NULL)&&(t->rc==NULL))
    {
        if(t->data<p->data)
            p->lc=t->lc;
        else
            p->rc=t->lc;
    }
    else if((t->lc==NULL)&&(t->rc!=NULL))
    {

```

```

    if(t->data<p->data)
        p->lc=t->rc;
    else
        p->rc=t->rc;
    }
else
{
    t1=t->lc;
    t2=t->rc;
    while(t1->rc!=NULL)
        t1=t1->rc;
    t1->rc=t2;
    if(t->data<p->data)
        p->lc=t->lc;
    else
        p->rc=t->lc;
}
}
free(t);
return R;
}

int count_total(BST *R)
{
    int c=0;
    BST *stk[10],*t;
    int top=-1;
    t=R;
    if(R==NULL)
        return c;
    else
    {
        do
        {
            while(t!=NULL)
            {
                stk[++top]=t;
                t=t->lc;
            }
            t=stk[top--];
            c++;
            t=t->rc;
        }while(t!=NULL || top>-1);
        return c;
    }
}

int count_term(BST *R)
{
    int tc=0;
    BST *stk[10],*t;

```

```

int top=-1;
t=R;
if(R==NULL)
return tc;
else
{
    do
    {
        while(t!=NULL)
        {
            stk[++top]=t;
            t=t->lc;
        }
        t=stk[top--];
        if(t->lc==NULL && t->rc==NULL)
        tc++;
        t=t->rc;
    }while(t!=NULL || top>-1);
    return tc;
}
}

int count_non_term(BST *R)
{
int ntc=0;
BST *stk[10],*t;
int top=-1;
t=R;
if(R==NULL)
return ntc;
else
{
    do
    {
        while(t!=NULL)
        {
            stk[++top]=t;
            t=t->lc;
        }
        t=stk[top--];
        if(t->lc==NULL && t->rc==NULL)
        ntc++;
        t=t->rc;
    }while(t!=NULL || top>-1);
    return ntc;
}
}

int mindata(BST *R)
{
    while(R->lc!=NULL)

```

```

    R=R->lc;
    return R->data;
}
int maxdata(BST *R)
{
    while(R->rc!=NULL)
        R=R->rc;
    return R->data;
}
void cleartree(BST *R)
{
    BST *stk[10],*t;
    int top=-1;
    t=R;
    do
    {
        while(t!=NULL)
        {
            stk[++top]=t;
            t=t->lc;
        }
        t=stk[top--];
        free(t);
        t=t->rc;
    }while(t!=NULL || top>-1);
}
void mirror(BST *R)
{
    BST *stk[10],*t,*temp;
    int top=-1;
    t=R;
    do
    {
        while(t!=NULL)
        {
            stk[++top]=t;
            t=t->lc;
        }
        t=stk[top--];
        temp=t->lc;
        t->lc=t->rc;
        t->rc=temp;
        t=t->rc;
    }while(t!=NULL || top>-1);
}
int height_of_tree(BST *R)
{
    BST *queue[max],*t;
    int f=0,r=-1,size=0,i=0;

```

```

t=R;
int c=-1;
if(R==NULL)
return c;
else
{
    queue[++r]=t;
    while(f<=r)
    {
        size=r-f+1;
        while(i<size)
        {
            t=queue[f++];
            if(t!=NULL)
            {
                queue[++r]=t->lc;
                queue[++r]=t->rc;
            }
            size--;
            if(size==0)
                c++;
        }
    }
    return c;
}

```

**OUTPUT –**

```

E:\DS LAB\vbnotepad
***** NON-RECURSIVE PROGRAM *****
1. INITIALIZATION
2. INSERT
3. DISPLAY
4. DELETE
5. HEIGHT OR MAX DEPTH
6. COUNT NODES
7. COUNT NODES
8. MINIMUM DATA
9. MAXIMUM DATA
10. MIRROR
11. MIRROR TREE
0. EXIT
Enter your choice : 2
Enter total no. of elements in the binary tree : 6
Enter element to insert : 50
Enter element to insert : 25
Enter element to insert : 16
Enter element to insert : 28
Enter element to insert : 75
Enter element to insert : 67
***** NON-RECURSIVE PROGRAM *****
1. INITIALIZATION
2. INSERT
3. DISPLAY
4. DELETE
5. HEIGHT OR MAX DEPTH
6. COUNT NODES
7. COUNT NODES
8. MINIMUM DATA
9. MAXIMUM DATA
10. MIRROR
11. MIRROR TREE
0. EXIT
Enter your choice : 5
Height or Max depth of tree : 3
***** NON-RECURSIVE PROGRAM *****
E:\Select EDS LAB\vbnotepad
Type here to search
O Select EDS LAB\vbnotepad
***** NON-RECURSIVE PROGRAM *****
1. INITIALIZATION
2. INSERT
3. DISPLAY
4. DELETE
5. HEIGHT OR MAX DEPTH
6. SEARCH
7. COUNT NODES
8. MINIMUM DATA
9. MAXIMUM DATA
10. MIRROR
11. MIRROR TREE
0. EXIT
Enter your choice : 6
Enter element to search : 5
Information not found
***** NON-RECURSIVE PROGRAM *****
1. INITIALIZATION
2. INSERT
3. DISPLAY
4. DELETE
5. HEIGHT OR MAX DEPTH
6. SEARCH
7. COUNT NODES
8. MINIMUM DATA
9. MAXIMUM DATA
10. MIRROR
11. MIRROR TREE
0. EXIT
Enter your choice : 9
Minimum data : 16
***** NON-RECURSIVE PROGRAM *****
1. INITIALIZATION
2. INSERT
3. DISPLAY
4. DELETE
5. HEIGHT OR MAX DEPTH
6. SEARCH
7. COUNT NODES
8. MINIMUM DATA
9. MAXIMUM DATA
10. MIRROR
11. MIRROR TREE
0. EXIT
Enter your choice : 7
1. TOTAL NODES
2. TERMINAL NODES
3. NON-TERMINAL NODES
0. EXIT
Enter your choice : 1
Total no. of nodes : 6
1. TOTAL NODES
2. TERMINAL NODES
3. NON-TERMINAL NODES
0. EXIT
Enter your choice : 0
***** NON-RECURSIVE PROGRAM *****
1. INITIALIZATION
2. INSERT
3. DISPLAY
4. DELETE
5. HEIGHT OR MAX DEPTH
6. COUNT NODES
7. COUNT NODES
8. MINIMUM DATA
9. MAXIMUM DATA
10. MIRROR
11. MIRROR TREE
0. EXIT
Enter your choice : 0
Exiting
Process exited after 59.3 seconds with return value 0
Press any key to continue . . .
E:\Select EDS LAB\vbnotepad
Type here to search
O Select EDS LAB\vbnotepad

```

## C PROGRAM – RECURSIVE –

```

#include<stdio.h>
#include<malloc.h>
struct bst
{
    int data;
    struct bst *lc;
    struct bst *rc;
};
typedef struct bst BST;

```

```

BST * init(BST *);
BST * insert(BST *,int);
BST * search(BST *,int);
void inorder(BST *);
void preorder(BST *);
void postorder(BST *);
void levelorder(BST *);
BST * rdelete(BST *,int);
BST * MinValueNode(BST *);
int height_of_tree(BST *);
void currentlevel(BST *,int);
int count_total(BST *);
int count_term(BST *);
int count_non_term(BST *);
int mindata(BST *);
int maxdata(BST *);
void mirror(BST *);
void cleartree(BST *);
void main()
{
    BST *R=NULL,*t;
    int n,i=0,x,ch,op;
    do
    {
        printf("\n ***** RECURSIVE PROGRAM *****\n");
        printf("\n 1. INITIALIZATION \n 2. INSERT \n 3. DISPLAY \n 4. DELETE \n 5. HEIGHT OR MAX
DEPTH \n 6. SEARCH");
        printf("\n 7. COUNT NODES \n 8. MINIMUM DATA \n 9. MAXIMUM DATA \n 10. MIRROR \n 11.
CLEAR TREE \n 0. EXIT");
        printf("\n Enter your choice : ");
        scanf("%d",&ch);
        switch(ch)
        {
            case 1:
                R=init(R);
                break;
            case 2:
                printf("\n Enter total no. of elements in the binary tree : ");
                scanf("%d",&n);
                i=0;
                while(i<n)
                {
                    i++;
                    printf("\n Enter element to insert : ");
                    scanf("%d",&x);
                    R=insert(R,x);
                }
                break;
            case 3:

```

```

do
{
    printf("\n 1. INORDER DISPLAY \n 2. PREORDER DISPLAY \n 3.
POSTORDER DISPLAY \n 4. LEVELORDER DISPLAY \n 0. EXIT");
    printf("\n Enter your choice : ");
    scanf("%d",&op);
    switch(op)
    {
        case 1:
            if(R==NULL)
                printf("\n Binary tree is empty");
            else
                inorder(R);
            break;
        case 2:
            if(R==NULL)
                printf("\n Binary tree is empty");
            else
                preorder(R);
            break;
        case 3:
            if(R==NULL)
                printf("\n Binary tree is empty");
            else
                postorder(R);
            break;
        case 4:
            if(R==NULL)
                printf("\n Binary tree is empty");
            else
                levelorder(R);
            break;
        case 0:
            break;
        default:
            printf("\n Wrong choice");
    }
}while(op!=0);
break;
case 4:
if(R==NULL)
printf("\n Deletion not possible as binary tree is empty");
else
{
    printf("\n Enter element to be deleted : ");
    scanf("%d",&x);
    R=rdelete(R,x);
}
break;

```

```

case 5:
    printf("\n Height or Max depth of tree : %d",height_of_tree(R));
    break;
case 6:
    if(R==NULL)
        printf("\n Search not possible as binary tree is empty");
    else
    {
        printf("\n Enter element to search : ");
        scanf("%d",&x);
        t=search(R,x);
        if(t!=NULL)
            printf("\n Information found");
        else
            printf("\n Information not found");
    }
    break;
case 7:
    do
    {
        printf("\n 1. TOTAL NODES \n 2. TERMINAL NODES \n 3. NON-TERMINAL
NODES \n 0. EXIT");
        printf("\n Enter your choice : ");
        scanf("%d",&op);
        switch(op)
        {
            case 1:
                i=count_total(R);
                printf("\n Total no. of nodes : %d",i);
                break;
            case 2:
                i=count_term(R);
                printf("\n Total no. of terminal nodes : %d",i);
                break;
            case 3:
                i=count_non_term(R);
                printf("\n Total no. of non-terminal nodes : %d",i);
                break;
            case 0:
                break;
            default:
                printf("\n Wrong choice");
        }
    }while(op!=0);
    break;
case 8:
    if(R==NULL)
        printf("\n Binary tree is empty");
    else

```

```

        {
            i=mindata(R);
            printf("\n Minimum data : %d",i);
        }
        break;
    case 9:
        if(R==NULL)
            printf("\n Binary tree is empty");
        else
        {
            i=maxdata(R);
            printf("\n Maximum data : %d",i);
        }
        break;
    case 10:
        if(R==NULL)
            printf("\n Binary tree is empty");
        else
            mirror(R);
        break;
    case 11:
        if(R==NULL)
            printf("\n Binary tree is empty");
        else
            cleartree(R);
        R=NULL;
        break;
    case 0:
        printf("\n Exiting");
        break;
    default:
        printf("\n Wrong choice");
    }
}while(ch!=0);
}
BST * init(BST *R)
{
    R=NULL;
    return R;
}
BST * insert(BST *R,int x)
{
    BST *temp;
    temp=(BST *)malloc(sizeof(BST));
    temp->data=x;
    temp->lc=temp->rc=NULL;
    if(R==NULL)
        R=temp;
    else

```

```

    {
        if(x<R->data)
            R->lc=insert(R->lc,x);
        else if(x>R->data)
            R->rc=insert(R->rc,x);
    }
    return R;
}
BST * search(BST *R,int x)
{
    BST *t;
    t=R;
    if(t!=NULL)
    {
        if(x<t->data)
            t=search(R->lc,x);
        else if (x>t->data)
            t=search(R->rc,x);
    }
    return t;
}
void preorder(BST *R)
{
    if(R!=NULL)
    {
        printf(" %d ",R->data);
        preorder(R->lc);
        preorder(R->rc);
    }
}
void inorder(BST *R)
{
    if(R!=NULL)
    {
        inorder(R->lc);
        printf(" %d ",R->data);
        inorder(R->rc);
    }
}
void postorder(BST *R)
{
    if(R!=NULL)
    {
        postorder(R->lc);
        postorder(R->rc);
        printf(" %d ",R->data);
    }
}
void levelorder(BST *R)

```

```

{
    int h=height_of_tree(R);
    int i;
    for(i=1;i<=h;i++)
        currentlevel(R,i);
}
void currentlevel(BST *R,int l)
{
    if(R==NULL)
        return;
    else if(l==1)
        printf(" %d ",R->data);
    else
    {
        currentlevel(R->lc,l-1);
        currentlevel(R->rc,l-1);
    }
}
int height_of_tree(BST *R)
{
    int lh,rh;
    if(R==NULL)
        return 0;
    else
    {
        lh=height_of_tree(R->lc);
        rh=height_of_tree(R->rc);
        if(lh>rh)
            return(lh+1);
        else
            return(rh+1);
    }
}
int count_total(BST *R)
{
    if(R==NULL)
        return 0;
    else
        return(1+ count_total(R->lc) + count_total(R->rc));
}
int count_term(BST *R)
{
    if(R==NULL)
        return 0;
    else if(R->rc==NULL && R->lc==NULL)
        return 1;
    else
        return(count_term(R->lc)+count_term(R->rc));
}

```

```

int count_non_term(BST *R)
{
    if(R==NULL || R->rc==NULL && R->lc==NULL)
        return 0;
    else
    {
        if(R->lc!=NULL || R->rc!=NULL)
            return(1+ count_non_term(R->lc) + count_non_term(R->rc));
    }
}
int mindata(BST *R)
{
    if(R->lc==NULL)
        return R->data;
    else
        return(mindata(R->lc));
}
int maxdata(BST *R)
{
    if(R->rc==NULL)
        return R->data;
    else
        return(maxdata(R->rc));
}
void mirror(BST *R)
{
    BST *temp;
    if(R==NULL)
        return;
    else
    {
        mirror(R->lc);
        mirror(R->rc);
        temp=R->lc;
        R->lc=R->rc;
        R->rc=temp;
    }
}
void cleartree(BST *R)
{
    if(R!=NULL)
    {
        cleartree(R->lc);
        cleartree(R->rc);
        free(R);
    }
}
BST * MinValueNode(BST *R)
{

```

```

BST *c=R;
while(c && c->lc!=NULL)
    c=c->lc;
    return c;
}
BST * rdelete(BST *R,int x)
{
    BST *temp;
    if(R==NULL)
        return R;
    if(x<R->data)
        R->lc= rdelete(R->lc,x);
    else if(x>R->data)
        R->rc= rdelete(R->rc,x);
    else
    {
        if(R->lc==NULL && R->rc==NULL)
            return NULL;
        else if(R->lc==NULL)
        {
            temp=R->rc;
            free(R);
            return temp;
        }
        else if(R->rc==NULL)
        {
            temp=R->lc;
            free(R);
            return temp;
        }
        temp=MinValueNode(R->rc);
        R->data=temp->data;
        R->rc=rdelete(R->rc,temp->data);
    }
    return R;
}

```

**OUTPUT –**

```

E:\DS LAB\Bstree
***** RECURSIVE PROGRAM *****
1. INITIALIZATION
2. INSERT
3. DISPLAY
4. DELETE
5. HEIGHT OR MAX DEPTH
6. PRINT
7. COUNT NODES
8. MINIMUM DATA
9. MAXIMUM DATA
10. MIRROR
11. CLEAR TREE
0. EXIT
Enter your choice : 2
Enter total no. of elements in the binary tree : 6
Enter element to insert : 50
Enter element to insert : 25
Enter element to insert : 16
Enter element to insert : 28
Enter element to insert : 75
Enter element to insert : 67
***** RECURSIVE PROGRAM *****
1. INITIALIZATION
2. INSERT
3. DISPLAY
4. DELETE
5. HEIGHT OR MAX DEPTH
6. PRINT
7. COUNT NODES
8. MINIMUM DATA
9. MAXIMUM DATA
10. MIRROR
11. CLEAR TREE
0. EXIT
Enter your choice : 3
1. INORDER DISPLAY
2. PREORDER DISPLAY
3. POSTORDER DISPLAY
***** RECURSIVE PROGRAM *****
2. PREORDER DISPLAY
3. POSTORDER DISPLAY
4. INORDER DISPLAY
0. EXIT
Enter your choice : 1
Height or Max depth of tree : 25
1. INORDER DISPLAY
2. PREORDER DISPLAY
3. POSTORDER DISPLAY
4. LEVELORDER DISPLAY
5. SEARCH
6. COUNT NODES
7. MINIMUM DATA
8. MAXIMUM DATA
9. HEIGHT OR MAX DEPTH
10. MIRROR
11. CLEAR TREE
0. EXIT
Enter your choice : 0
***** RECURSIVE PROGRAM *****
1. INITIALIZATION
2. INSERT
3. DISPLAY
4. DELETE
5. HEIGHT OR MAX DEPTH
6. SEARCH
7. COUNT NODES
8. MINIMUM DATA
9. MAXIMUM DATA
10. MIRROR
11. CLEAR TREE
0. EXIT
Enter your choice : 7
Total no. of terminal nodes : 3
1. TOTAL NODES
2. TERMINAL NODES
3. NON-TERMINAL NODES
0. EXIT
Enter your choice : 0
***** RECURSIVE PROGRAM *****
1. INITIALIZATION
2. INSERT
3. DISPLAY
4. DELETE
5. HEIGHT OR MAX DEPTH
6. SEARCH
***** RECURSIVE PROGRAM *****
0. EXIT
Enter your choice : 0
***** RECURSIVE PROGRAM *****
1. INITIALIZATION
2. INSERT
3. DISPLAY
4. DELETE
5. HEIGHT OR MAX DEPTH
6. SEARCH
7. COUNT NODES
8. MINIMUM DATA
9. MAXIMUM DATA
10. MIRROR
11. CLEAR TREE
0. EXIT
Enter your choice : 5
Height or Max depth of tree : 3
***** RECURSIVE PROGRAM *****
1. INITIALIZATION
2. INSERT
3. DISPLAY
4. DELETE
5. HEIGHT OR MAX DEPTH
6. SEARCH
7. COUNT NODES
8. MINIMUM DATA
9. MAXIMUM DATA
10. MIRROR
11. CLEAR TREE
0. EXIT
Enter your choice : 0
Exiting
Process exited after 88.55 seconds with return value 0
Press any key to continue . . .

```

### Q43. WAP for implementing Heap Sort.

Ans43.

#### C PROGRAM –

```

#include<stdio.h>
#define max 20
void heapSort(int [], int);
void heapify(int [], int, int);
void printArray(int [], int);
void swap(int *, int *);
void main()
{

```

```

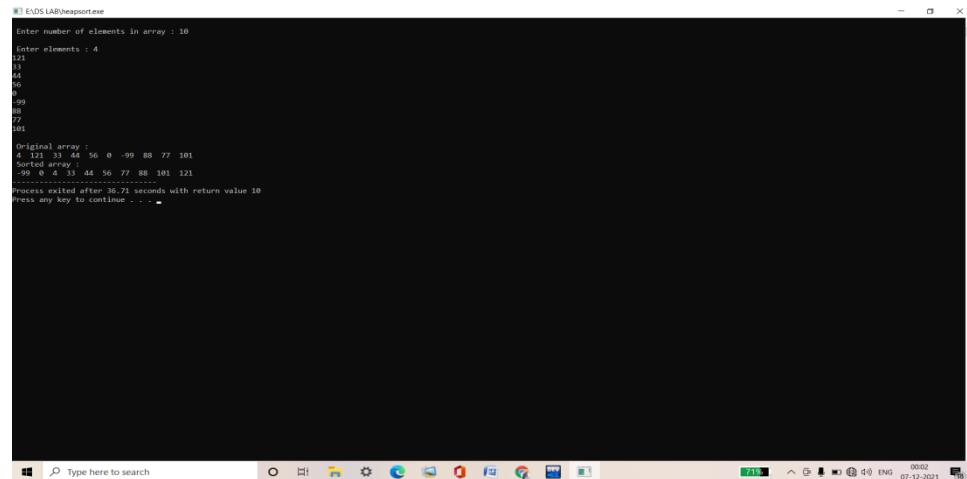
int a[max];
int n,i;
printf("\n Enter number of elements in array : ");
scanf("%d",&n);
printf("\n Enter elements : ");
for (i=0;i<n;i++)
    scanf("%d",&a[i]);
printf("\n Original array : \n");
for (i=0;i<n;i++)
    printf(" %d ",a[i]);
heapSort(a,n);
printf("\n Sorted array : \n");
printArray(a,n);
}
void swap(int *a, int *b)
{
    int temp=*a;
    *a=*b;
    *b=temp;
}
void heapify(int arr[], int n, int i)
{
    int largest=i;
    int left=2*i+1;
    int right=2*i+2;
    if (left<n && arr[left]>arr[largest])
        largest=left;
    if (right<n && arr[right]>arr[largest])
        largest=right;
    if (largest!=i)
    {
        swap(&arr[i],&arr[largest]);
        heapify(arr,n,largest);
    }
}
void heapSort(int arr[], int n)
{
    int i;
    for(i=n/2-1;i>=0;i--)
        heapify(arr,n,i);
    for(i=n-1;i>=0;i--)
    {
        swap(&arr[0],&arr[i]);
        heapify(arr,i,0);
    }
}
void printArray(int arr[], int n)
{
    int i;

```

```
for(i=0;i<n;++i)
printf(" %d ",arr[i]);
```

```
}
```

## OUTPUT –



A screenshot of a Windows command prompt window titled "C:\ENDS L&R\haspotree". The window displays the following text:

```
Enter number of elements in array : 10
Enter elements : 4
121
33
44
56
0
-99
88
77
101
103
Original array :
4 121 33 44 56 0 -99 88 77 101
Sorted array :
0 -99 33 44 56 77 88 101 121
-----
```

The window also shows the system tray with icons for network, battery, volume, and date/time (07-12-2021).

