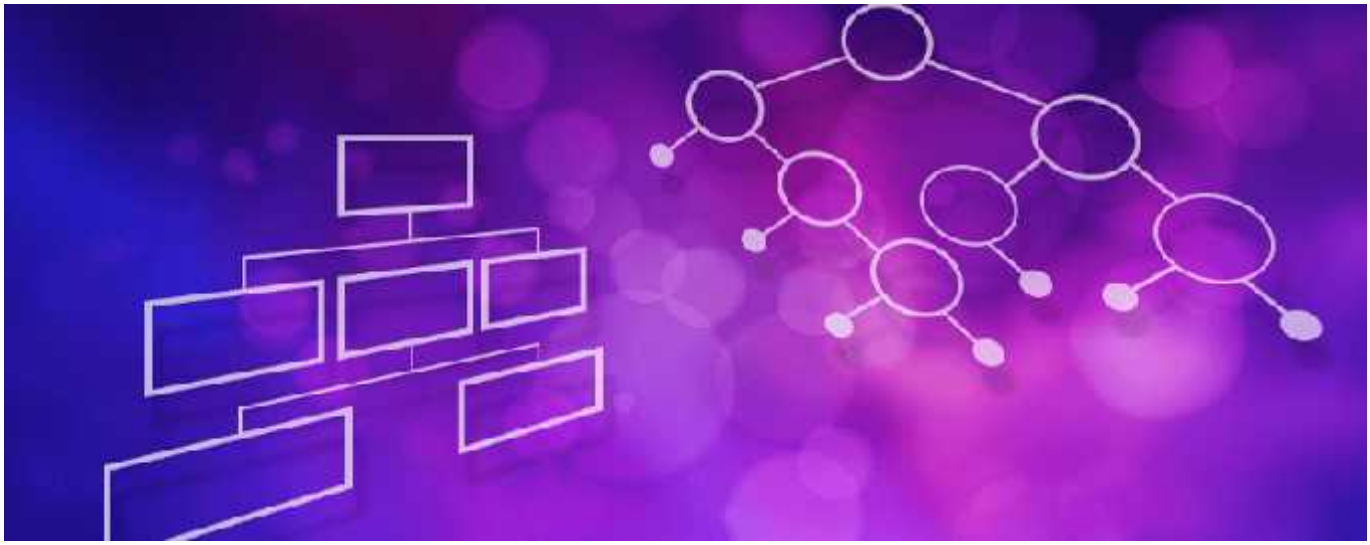
VVP ENGINEERING COLLEGE
DEPARTMENT OF COMPUTER ENGINEERING BE SEM 3
LAB MANUAL
A.Y. 2019-20 , 2020-21 (ODD SEM)

SUBJECT NAME :- DATA STRUCTURES

SUBJECT CODE :- 3130702

SUBJECT FACULTY:- Mr. VIRAJ C. DAXINI



VVP ENGINEERING COLLEGE

DEPARTMENT OF COMPUTER ENGINEERING

Vision of the Department

Transforming students into globally efficient professionals with moral values.

Mission of the Department

To provide a strong foundation of computer engineering through effective teaching learning process.

To enhance industry linkage & alumni network for better placement and real-world exposure.

To provide various opportunities & platforms for all round development of students & encourage them for value-based practices.

VVP ENGINEERING COLLEGE

DEPARTMENT OF COMPUTER ENGINEERING

Course Outcomes

1. Analyze the time and space complexity of algorithms and data structures.
2. Describe linear data structures with their representations.
3. Describe non-linear data structures with their representations.
4. Apply sorting and searching technique on data set.

Data Structure Lab - Assignment-1

1. Write a c program to implement function Swap using two different parameter passing mechanism.
2. Write a c program to store 5 values in appropriate data structure and compute addition for the same, modify the size to store 10 values and compute addition.
3. Write a c program to get record (Player name, team name & runs of 3 innings) of any three players from Indian cricket team. Print the record according to name of players in ascending order.
4. Perform 3rd Program using pointer.

Data Structure Lab - Assignment-2

1. Write a C Program to perform addition on 2 complex number using structure. (Use user defined function addition)
2. Write a C Program to implement searching on unordered array of 10 integer value.
3. Write a C Program to find largest value from array of 10 integers.
4. Write a C Program to Swap Max and Min Value from array of 10 integer value.
5. Write a C Program to insert a value in array of 10 integers at specific position.
6. Write a C Program to delete a value in array of 10 integers from specific position.

Data Structure Lab - Assignment-3

1. Write a C Program to implement searching on ordered array of 10 integer value.
2. Write a C Program to implement Stack with all necessary overflow and underflow condition (Use array as data structure). 1) PUSH 2) POP 3) DISPLAY.
3. Write a C Program to implement Stack with all necessary overflow and underflow condition (Use structure as data structure).
4. Write a C Program to implement Stack with all necessary overflow and underflow condition (Using Pointer).
5. Write a C Program to convert infix notation into its equivalent postfix notation using stack.

Data Structure Lab - Assignment-4

1. Write a c Program to implement Tower of Hanoi problem.
2. Write a C Program to reverse a string using stack.
3. Write a C Program to implement Infix to prefix conversion using stack.
4. Write a C Program to Create a Binary Search Tree (Insertion, deletion and traversal).

Data Structure Lab - Assignment-5

1. Write a C program to implement simple queue (Insertion, deletion and traversal).
2. Write a C program to implement circular queue (Insertion, deletion and traversal).
3. Write a C program to implement priority queue (Insertion , deletion and traversal).

Data Structure Lab - Assignment-6

1. Write a c program to organize 10 integer values in ascending order by repeatedly finding the minimum element and putting it at the beginning of list.
2. Write a C program to organize 10 integer values in ascending order by repeatedly swapping the adjacent elements if they are in wrong order.
3. Write a C program to organize 10 integer values in ascending order by divide and conquer technique. Divide the input array in two halves, calls itself for the two halves and then merges the two sorted halves.
4. Write a C Program to implement QUICK sort.

Data Structure Lab - Assignment-7

1. Write a c program to implement DFS graph traversal.
2. Write a c program to implement BFS graph traversal.
3. Write a c program to implement Prim's algorithm to find MST from given graph.
4. Write a c program to implement Kruskal's algorithm to find MST from given graph.

Data Structure Lab - Assignment-8

1. Write a c program to implement singly linked list for the following function.

(a) Insert a node at the front of the linked list. (b) Insert a node at the end of the linked list. (c) Insert a node in order. (d) Delete any node from the linked list. (e) Count total no of nodes in list. (f) Insert a node at any position in linked list. (g) Display all nodes (traversal of Linked list). (h) Copy List.

2. Write a program to implement following operations on the Circular linked list.

(a) Insert a node at the front of the linked list. (b) Insert a node at the end of the linked list. (c) Insert a node in order. (d) Delete any node from the linked list. (e) Count total no of nodes in list. (f) Display all nodes (traversal of Linked list).

3. Write a program to implement following operations on the Doubly linked list.

(a) Insert a node at the front of the linked list. (b) Insert a node at the end of the linked list. (c) Insert a node at any position in linked list. (d) Delete any node from the linked list. (e) Count total no of nodes in list. (f) Display all nodes (traversal of Linked list).

Data Structure Lab - Assignment-9

1. Write a c program to implement stack using Linked list. (PUSH, POP, DISPLAY)

2. Write a c program to implement queue using Linked List. (enqueue, dequeue , display).

3. Write a c program to implement Double-ended queue using Linked List. (enqueue, dequeue , display).

4. Write a c program to swap max and min element from singly linked ist without swapping address.

5. Write a c program to remove duplicate values from singly Linked list.

6. Write a c program to implement bubble sort for singly linked list.

ASSIGNMENT -1:

1) Write a C program to implement function swap using two different parameter passing mechanism.

```
#include<stdio.h>

#include<conio.h>

void swap(int x ,int y)

{

int temp; temp=x; x=y;

y=temp;

printf("After swapping the values are a= %d, b= %d",x,y);

}

void main()

{

int a,b;

printf("NAME:- VVP\n");

printf("ENROLLMENT_NUMBER:- --\n");

printf("BATCH:- --\n");

printf("Enter 1st Number a:");

scanf("%d",&a);

printf("Enter 2nd Number b:");

scanf("%d",&b);

swap(a,b);
getch();

}
```

OUTPUT:

2) Write a C program to store 5 values in appropriate data structure and compute addition for the same, modify the size to store 10 values and compute addition.

```
#include<stdio.h>
#include<conio.h>
#include<stdlib.h>
void main()
{
int *ptr;
int i,n,a[5],sum=0,sum1=0; printf("NAME:- VVP\n");
printf("ENROLLMENT_NUMBER:- --\n"); printf("BATCH:- --\n");
printf("Enter 5 elements:");
for(i=0;i<5;i++)
{
scanf("%d",&a[i]);
sum+=a[i];
}
printf("Sum=%d",sum);
printf("\nEnter total no of element :");
scanf("%d",&n);
ptr=(int *)malloc(n*sizeof(int));
printf("Enter elements:");
for(i=0;i<n;i++)
{
scanf("%d",ptr+i);
sum1+=*(ptr+i);
```

```
} printf("Sum=%d",sum1); free(ptr);
}
```

OUTPUT:

3) Write a C program to get record (player name, team name &runs of 3 innings) of any 3 players from Indian cricket team. Print the record according to name of players in ascending order.

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

struct player
{
char Pname[50];
char tname[50];
int runs[3];
int sum;
float avg;
}v[3],temp;
void main()
{
int i,j;
printf("NAME:- VVP\n");
printf("ENROLLMENT_NUMBER:- --\n");
printf("BATCH:- --\n");
for(i=0;i<3;i++)
{
printf("\nEnter %d Player name:",i+1);
scanf("%s",&v[i].Pname);
printf("\nEnter %d player's teamname:",i+1);
```

```
scanf("%s",&v[i].tname);
printf("\nEnter the runs of last 3 innings:");
for(j=0;j<3;j++)
{
scanf("%d",&v[i].runs[j]);
v[i].sum = v[i].sum + v[i].runs[j];
}
v[i].avg=(float)v[i].sum/3;
}
for(i=0;i<3;i++)
{
for(j=0;j<3;j++)
{
        if(v[j].avg < v[j+1].avg)
        {
                temp=v[j];
                v[j]=v[j+1];
                v[j+1]=temp;
        }
}
}
for(i=0;i<3;i++)
{
printf("\n Name of Player %d is %s",i+1,v[i].Pname);
printf("\n Name of Team %d is %s",i+1,v[i].tname);
printf("\n Average of runs in lst 3 matches is %f",v[i].avg);
```

```
}
}
```

OUTPUT

4) Write a C program to get record (player name, team name &runs of 3 innings) of any 3 players from Indian cricket team. Print the record according to name of players in ascending order (using pointer).

```
#include<stdio.h>
#include<conio.h>
#include<string.h>
#include<stdlib.h>
struct player
{
char Pname[10];
char tname[10];
int runs[3];
int sum;
float avg;
}*ptr,temp;
void main()
{
int i,j,n;
printf("NAME:- VVP\n");
printf("ENROLLMENT_NUMBER:- --\n");
printf("BATCH:- --\n");
printf("\n How many players record you want to enter?");
scanf("%d",&n);
ptr=(struct player *)malloc(n*sizeof(struct player));
```

```
printf("\nEnter Record:");
for(i=0;i<n;i++)
{
printf("\nEnter %d Player name:",(i+1));
scanf("%s",&(ptr+i)->Pname);
printf("\nEnter %d  player's Team name:",(i+1));
scanf("%s",&(ptr+i)->tname); printf("Enter Runs in last 3 matches:");
        (ptr+i)->sum=0;
        for(j=0;j<3;j++)
        {
                scanf("%d",&(ptr+i)->runs[j]); (ptr+i)->sum+=(ptr+i)->runs[j];
        }
        (ptr+i)->avg=(float)(ptr+i)->sum/3.0;
}
for(i=0;i<n;i++)
{
        for(j=0;j<n-1;j++)
        {
                if((ptr+j)->avg < (ptr+j+1)->avg)
                {
                temp=*(ptr+j);
                *(ptr+j)=*(ptr+j+1);
                *(ptr+j+1)=temp;
                }
        }
}
```

```
printf("\nAfter sorting their average in ascending order ,data is as follows:\n");

for(i=0;i<n;i++)

{

printf("\nName of Player %d is %s",i+1,(ptr+i)->Pname);

printf("\nName of Team %d is %s",i+1,(ptr+i)->tname);

printf("\nAverage of last 3 matches is %f",(ptr+i)->avg);

printf("\n");

}
free(ptr);

}
```

OUTPUT:

ASSIGNMENT -2:

1) Write a C program to perform addition on 2 complex number using structure.(Use user defined function)

```
#include<stdio.h>
#include<conio.h>
struct complex
{
int r;
float i;
};
struct complex addition (struct complex ,struct complex);
void main()
{
struct complex c1,c2,ans;
printf("NAME:- VVP\n");
printf("ENROLLMENT_NUMBER:- --\n");
printf("BATCH:- --\n");
printf("Enter the real part of  1st complex number:");
scanf("%d",&c1.r);
printf("Enter the imaginary part of 1st complex number:");
scanf("%f",&c1.i);
printf("Enter the real part of 2nd complex number:");
scanf("%d",&c2.r);
printf("Enter the imaginary part of 2nd complex number:");
scanf("%f",&c2.i);
```

```
ans=addition(c1,c2);
printf("Addition of this 2 complex numbers is %d +
%fi",ans.r,ans.i);
}
struct complex addition(struct complex x1,struct complex x2)
{
struct complex temp;
temp.r=x1.r+x2.r; temp.i=x1.i+x2.i;
return temp;
}
```

OUTPUT:

2) Write a C program to implement searching on unordered array of 10 integer value.

```
#include<stdio.h>

#include<conio.h>

void main()

{

int size,i,search,count=0;
printf("NAME:- VVP\n");
printf("ENROLLMENT_NUMBER:- --\n");
printf("BATCH:- --\n");
printf("Enter the size of array:");

scanf("%d",&size);

int arr[size];

printf("Enter array elements:");

for(i=0;i<size;i++)

{

scanf("%d",&arr[i]);

}

printf("\nEnter the element to be searched:");

scanf("%d",&search);

for(i=0;i<size;i++)

{

if(arr[i]==search)

{

printf("Element %d is found at position
%d\n",search,i+1);

count++;

}
```

```
}

if(count==0)

printf("Element is not found");
}
```

OUTPUT:

3) Write a C program to find largest value from array of 10 integers.

```
#include<stdio.h>
#include<conio.h>
void main()
{
int size,i,max=0;
printf("NAME:- VVP\n");
printf("ENROLLMENT_NUMBER:- --\n"); printf("BATCH:- --\n");
printf("Enter size of array:");
scanf("%d",&size);
int arr[size];
printf("Enter array elements:");
for(i=0;i<size;i++)
{
scanf("%d",&arr[i]);
}
for(i=0;i<size;i++)
{
        if(arr[i]>max)
                max=arr[i];
}
printf("Maximum value is %d",max);
}
```

OUTPUT:

4) Write a C program to Swap Max and Min Value from array of 10 integer value.

```
#include<stdio.h>
#include<conio.h>
void main()
{
int i,size,max,maxpos,min,minpos,temp;
printf("NAME:- VVP\n");
printf("ENROLLMENT_NUMBER:- --\n");
printf("BATCH:- --\n");
printf("Enter the size of array:");
scanf("%d",&size);
int arr[size];
printf("Enter the array elements:");
for(i=0;i<size;i++)
{
scanf("%d",&arr[i]);
}
max=arr[0]; maxpos=0;
min=arr[0]; minpos=0;
for(i=0;i<size;i++)
{
        if(arr[i]>max)
        {
                max=arr[i];
                maxpos=i;
```

```
        }
        if(arr[i]<min)
        {
                min=arr[i];
                minpos=i;
        }
}
temp=arr[maxpos];
arr[maxpos]=arr[minpos];
arr[minpos]=temp;
printf("After swapping Maximum and Minimum numbers, Array is as follows:");
for(i=0;i<size;i++)
{
printf("\n%d",arr[i]);
}
}
```

OUTPUT:

5) Write a C program to insert a value in array of 10 integers at specific position.

```
#include<stdio.h>
#include<conio.h>
void main()
{
int i,size,insert,loc;
printf("NAME:- VVP\n");
printf("ENROLLMENT_NUMBER:- --\n"); printf("BATCH:- --\n");
printf("Enter the size of array:");
scanf("%d",&size);
int arr[size+1];
printf("Enter array elements:");
for(i=0;i<size;i++)
{
scanf("%d",&arr[i]);
}
printf("Enter the location of element at what you want to insert an element:");
scanf("%d",&loc);
printf("Enter the element you want to insert:");
scanf("%d",&insert);
for(i=size-1;i>=loc-1;i--)
{
        arr[i+1]=arr[i];
}
arr[loc-1]=insert;
```

```
printf("After insertion,Array is as follows: ");
for(i=0;i<size+1;i++)
{
printf("\n%d",arr[i]);
}
}
```

OUTPUT:

6) Write a C program to delete a value in array of 10 integers from specific position.

```
#include<stdio.h>
#include<conio.h>
void main()
{
int size,i,loc;
printf("NAME:- VVP\n");
printf("ENROLLMENT_NUMBER:- --\n");
printf("BATCH:- --\n");
printf("Enter the size of array:");
scanf("%d",&size);
int a[size];
printf("Enter array elements:");
for(i=0;i<size;i++)
{
scanf("%d",&a[i]);
}
printf("Enter the array location:");
scanf("%d",&loc);
for(i=loc-1;i<size;i++)
{
a[i]=a[i+1];
}
```

```
a[size]=0;
printf("Array after deletion is as follows:");
for(i=0;i<size-1;i++)
{
printf("\n%d",a[i]);
}
}
```

OUTPUT:

ASSIGNMENT -3:

1)Write a C program to implement searching on ordered array of 10 integer value.

```
#include<stdio.h>
#include<conio.h>
void main()
{
int a[11];
int i,search,high,low,mid;
printf("NAME:- VVP\n");
printf("ENROLLMENT_NUMBER:- --\n");
printf("BATCH:- --\n");
printf("Enter 10 values in ascending order:");
for(i=0;i<10;i++)
{
scanf("%d",&a[i]);
}
printf("Enter the element you want to search:");
scanf("%d",&search);
low=0; high=9;
while(low<=high)
{
mid=(low+high)/2;
        if(a[mid]>search)
        {
                high=mid-1;
        }
        else if(a[mid]<search)
                {
                low=mid+1;
                }
                else
                {
                printf("\nElement is found at %d position",(mid+1));
                break;
                }
```

```
if(low>high)
{
printf("Element is not found");
}
}
}
```
OUTPUT:

2) Write a C program to implement Stack with all necessary overflow and underflow condition. (Use array as data structure).

```
#include<stdio.h>
#include<conio.h>
#define max 10 int st[max];
int top=-1;

void push(int);
int pop();
void display();

void main()
{
int ch,val,ans,loc,number;
printf("NAME:- VVP\n");
printf("ENROLLMENT_NUMBER:- --\n");
printf("BATCH:- --\n");

while(1)
{
printf("\n 1)PUSH 2)POP 3)DISPLAY 4)PEEP5)CHANGE");
scanf("%d",&ch);

switch(ch)
{

case 1:
printf("\n Enter the value to be inserted:");
scanf("%d",&val);
push(val);
break;

case 2:
ans=pop();
printf("\nElement poped from stack = %d",ans);
break;
```

```
case 3:
display();
break;

case 4:
printf("Enter the location of element, you wanted to look at:");
scanf("%d",&loc);
peep(loc);
break;

case 5:
printf("Enter the location of element you wanted to change:");
scanf("%d",&loc);
printf("\nEnter the value you wanted to insert:");
scanf("%d",&number);
change(loc,number);
break;

}
}
}

void push(int x)
{
if(top==max-1)
{
printf("\nOVERFLOWW");
}
else
{
top++;
st[top]=x;
}
}

int pop()
{
int y;
if(top==-1)
```

```
{
printf("\n UNDERFLOWWW");
return 0;
}
Else
{
y=st[top]; top--; return y;
}
}

void display()
{
int i;
        if(top==-1)
        {
        printf("UNDERFLOWW");
        }
        else
        {
        for(i=top;i>=0;i--)
                {
                printf("%d\n",st[i]);
                }
        }
}

void peep(int x)
{
        if(top==-1)
        {
        printf("UNDERFLOWWW");
        }
        else
        {
        printf("Element at %d location is %d",x,st[top-x+1]);
        }
}
```

```
void change(int x,int y)
{
int i;
        if(top==-1)
        {
        printf("UNDERFLOWW");
        }
        else
        {
        for(i=top;i>=0;i--)
                {
                        if(i==x-1)
                        {
                        st[i]=y;
                        }
                }
        }
}
```

OUTPUT:

3) Write a C program to implement stack with all necessary overflow and underflow conditions (Use structure as data structure).

```
#include<stdio.h>
#include<conio.h>
#define max 10

struct stack
{
int st[max];
int top;
}s;

void push(int);
int pop();
void display();

void main()
{
int ch,val,ans,loc,number;
s.top = -1;
printf("NAME:- VVP\n");
printf("ENROLLMENT_NUMBER:- --\n");
printf("BATCH:- --\n");

while(1)
{
printf("\n 1)PUSH 2)POP 3)DISPLAY 4)PEEP5)CHANGE");
scanf("%d",&ch); switch(ch)
{
case 1:
printf("\n Enter the value to be inserted:");
scanf("%d",&val);
push(val);
break;

case 2:
ans=pop();
printf("\nElement poped from stack = %d",ans);
```

```
break;

case 3:
display();
break;

case 4:
printf("Enter the location of element, you wanted to look at:");
scanf("%d",&loc);
peep(loc);
break;

case 5:
printf("Enter the location of element you wanted to change:");
scanf("%d",&loc);
printf("\nEnter the value you wanted to insert:");
scanf("%d",&number);
change(loc,number);
break;
}
}
}

void push(int x)
{
        if(s.top==max-1)
        {
                printf("\nOVERFLOWW");
        }
        else
        {
                (s.top)++;
                s.st[s.top]=x;
        }
}
```

```
int pop()
{
int y;
        if((s.top)==-1)
        {
                printf("\n UNDERFLOWWW");
                return 0;
        }
        else
        {
                y=s.st[s.top]; s.top--;
                return y;
        }
}

void display()
{
int i;
        if((s.top)==-1)
        {
                printf("UNDERFLOWW");
        }
        else
        {
                for(i=s.top;i>=0;i--)
                {
                        printf("%d\n",s.st[i]);
                }
        }
}

void peep(int x)
{
        if((s.top)==-1)
        {
                printf("UNDERFLOWWW");
        }
        else
        {
                printf("Element at %d location is %d",x,s.st[s.top-x+1]);
```

```
        }
}

void change(int x,int y)
{
int i;
        if((s.top)==-1)
        {
        printf("UNDERFLOWW");
        }
        else
        {
                for(i=s.top;i>=0;i--)
                {
                if(i==(x-1))
                {
                        s.st[i]=y;
                }
        }
}

}
```

4)Write a C program to implement Stack with all necessary overflow and underflow conditions (Using pointer)

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

struct stack
{
int data;
struct stack *loc;
}*top=NULL;

void main()
{
int ch,val,ans,loc,number;
printf("NAME:- VVP\n");
printf("ENROLLMENT_NUMBER:- --\n");
printf("BATCH:- --\n");

while(1)
{
printf("\n 1)PUSH 2)POP 3)DISPLAY");
scanf("%d",&ch);

switch(ch)
{
case 1:
printf("\n Enter the value to be inserted:");
scanf("%d",&val);
push(val);
break;

case 2:
ans=pop();
printf("\nElement poped from stack = %d",ans);
break;
```

```
case 3:
display();
break;
}
}
}
void push(int j)
{
struct stack *temp;
temp=(struct stack *)malloc(sizeof(struct stack));
temp->data = j;
temp->loc=top;
top=temp;
}

int pop()
{
int ans;
        if(top==NULL)
        {
                printf("UNDERFLOWW");
                return 0;
        }
        else
        {
                ans=top->data;
                top=top->loc;
                return ans;
        }
}
```

```
void display()
{
struct stack *ptr=NULL;
ptr=top;
while(ptr != NULL)
{
        printf("%d \n",ptr->data);
        ptr=ptr->loc;
}
}
```
OUTPUT:

5) Write a C program to convert infix notation into its equivalent postfix notation using stack.

```
#include<stdio.h>
#include<conio.h>
#include<stdlib.h>
char st[20];
int top=-1;
int f(char);

void push(char);
char pop();
int r(char);
void main()
{
char infix[15],ans[15]; int i,c=0,rank=0;
push('#');
printf("NAME:- VVP\n");
printf("ENROLLMENT_NUMBER:- --\n");
printf("BATCH:- --\n");
printf("Enter the INFIX string:");
gets(infix);
for(i=0;infix[i]!=0;i++)
{
        while(f(infix[i])<=f(st[top]))
        {
                ans[c]=pop();
                rank=rank +r(ans[c]);
                c++;
                if(rank<1)
                {
                        printf("\n INVALID");
                        exit(0);
                }
        }

push(infix[i]);
}
```

```
while(st[top]!='#')
{
        ans[c]=pop();
        rank=rank+r(ans[c]);
        c++;

        if(rank<1)
        {
                printf("\n INVALID");
                exit(0);
        }
}
ans[c]=NULL;

        if(rank==1)
        {
        printf("POSTFIX string=%s",ans);
        }
}

int f(char a)
{
switch(a)
{
case '+':
case '-':
return 1;
case '*':
case '/':
return 2;
case '#':
return 0; case 'a': case 'b':
case 'c':
return 4;
}
}
```

```
int r(char z)
{
switch(z)
{
case '+': case '-': case '*':
case '/':
return -1;
case 'a': case 'b': case 'c':
return 1;
}
}

void push(char x)
{
        if(top==19)
        {
        printf("\nUNDERFLOWW");
        }
        else
        {
        top++;
        st[top]=x;
        }
}

char pop()
{
char y;
y=st[top];
top--;
return y;
```

OUTPUT:

ASSIGNMENT – 4 :

1) Write a C program to implement Tower of Hanoi problem.

```
#include<stdio.h>
#include<conio.h>
void towerOfHanoi(int n, char from_rod, char to_rod, char aux_rod)
{
if (n == 1)
{
printf("\n Move disk 1 from rod %c to rod %c", from_rod, to_rod);
return;
}
towerOfHanoi(n-1, from_rod, aux_rod, to_rod);
printf("\n Move disk %d from rod %c to rod %c", n, from_rod, to_rod);
towerOfHanoi(n-1, aux_rod, to_rod, from_rod);
}

int main()
{
int n;
printf("NAME:- VVP\n"); printf("ENROLLMENT_NUMBER:- --\n");
printf("BATCH:- --\n");
printf("\nEnter number of disks in Rod A:");
scanf("%d",&n); towerOfHanoi(n, 'A', 'C', 'B');
return 0;
}
```

OUTPUT:

2) Write a C program to reverse a string using stack.

```
#include<stdio.h>
#include<conio.h>
char st[20];
char pop();
int top=-1;
void main()
{
char string[25],reversed[25];
int i,length;
printf("\nEnter the string:");
scanf("%s",&string);
length=strlen(string);

for(i=0;i<length;i++)
{
top++;
st[top]=string[i];
}

for(i=0;i<length;i++)
{
reversed[i]=pop();
}

reversed[i]=NULL;
printf("Reversed string is : %s",reversed);
}
char pop()
{
char y;
y=st[top];
top--;
return y;
}
```

OUTPUT:

3) Write a C program to implement Infix to Prefix conversion using stack.

```
#include<stdio.h>
#include<conio.h>
#include<string.h>
char st[20];
int top=-1;
char initial_pop();
int f(char);
void push(char);
char pop();
int r(char);
void main()
{
char infix[15],ans[15],reversed[15],final_ans[15];
int i,c=0,rank=0,length=0;
push('#');
printf("\nEnter the INFIX string:");
gets(infix);
length=strlen(infix);
for(i=0;i<length;i++)
{
top++;
st[top]=infix[i];
}
```

```
for(i=0;i<length;i++)
{
reversed[i]=initial_pop();
}
reversed[i]=NULL;
for(i=0;reversed[i]!=0;i++)
{
        while(f(reversed[i])<=f(st[top]))
        {
                ans[c]=pop();
                rank=rank +r(ans[c]);
                c++;
                if(rank<1)
                {
                printf("\n INVALID");
                exit(0);
                }
        }
push(reversed[i]);
}
while(st[top]!='#')
{
ans[c]=pop();
rank=rank+r(ans[c]);
c++;
if(rank<1)
{
```

```
printf("\n INVALID");
exit(0);
}
}
ans[c]=NULL;

for(i=0;i<length;i++)
{
top++;
st[top]=ans[i];
}
for(i=0;i<length;i++)
{
final_ans[i]=initial_pop();
}
final_ans[length]=NULL;
printf("PREFIX string=%s",final_ans);

}
char initial_pop()
{
char y; y=st[top]; top--;
return y;
}
```

```
int f(char a)
{
switch(a)
{
case '+':
case '-':
return 1;
case '*':
case '/':
return 2;
//break;
case '#':
return 0; case 'a': case 'b':
case 'c':
return 4;
}
}
int r(char z)
{
switch(z)
{
case  '+':  case  '-':  case
'*': case '/':
return -1;
case  'a':  case  'b':  case
'c':
```

```
return 1;

}

}

void push(char x)

{
        if(top==19)
        {
                printf("\nUNDERFLOWW");
        }
        else
        {

              top++;
                st[top]=x;
        }
}

char pop()

{

char y; y=st[top]; top--;

return y;

}
```

OUTPUT:

4) Write a C program to implement Binary Search Tree (Insertion, Deletion and Traversal).

```
#include<stdio.h>
#include<conio.h>
struct node
{
int value;
struct node *left;
struct node *right;
};
struct node *root;
struct node *insert(struct node *r, int data);
void inOrder(struct node *r); void
preOrder(struct node *r); void
postOrder(struct node *r);
struct node *Delete(struct node *r, int data);
struct node * FindMin(struct node *root);
void main()
{
root = NULL; int n, v, m;
char ch;
printf("\nHow many data's do you want to insert ?\n");
scanf("%d", &n);
```

```
for(int i=0; i<n; i++)
{
printf("Data %d: ", i+1);
scanf("%d", &v);
root = insert(root, v);
}

printf("Inorder Traversal: ");

inOrder(root);

printf("\nPreorder Traversal: ");

preOrder(root);

printf("\nPostorder Traversal: ");

postOrder(root);

printf("\nEnter the data you want to delete :");

scanf("%d",&m);

root = Delete(root,m);

printf("\nInorder traversal is :");

inOrder(root);

printf("\nPreorder traversal is :");

preOrder(root);

printf("\nPostorder traversal is :");

postOrder(root);

}
```

```
struct node* insert(struct node* r, int data)
{
if(r==NULL)
{
r = (struct node*) malloc(sizeof(struct node));
r->value = data;
r->left = NULL;
r->right = NULL;
}
else if(data   <   r->value)
{
r->left = insert(r->left, data);
}
else
{
r->right = insert(r->right, data);
}
return r;
}
```

```
void preOrder(struct node* r)
{
if(r!=NULL)
{
printf("%d ", r->value); preOrder(r->left);
preOrder(r->right);
}
}

void postOrder(struct node* r)
{
if(r!=NULL)
{
postOrder(r->left); postOrder(r->right);
printf("%d ", r->value);
}
}

void inOrder(struct node* r)
{
if(r!=NULL)
{
inOrder(r->left); printf("%d ", r->value);
inOrder(r->right);
}
}
```

```
struct node *Delete(struct node *r, int data)
{
if (r== NULL)
{
return NULL;
}
if (data < r->value)
{
r->left = Delete(r->left, data);
}
else if (data> r->value)
{
r->right = Delete(r->right, data);
}
else
{
if (r->left == NULL && r->right == NULL)
{
free(r);
r= NULL;
}
else if (r->left == NULL)
{
struct Node *temp = r;
r= r->right;
free (temp);
```

```
}
else if (r->right == NULL)
{
struct Node *temp = r;
r = r->left;
free (temp);
}
else
{
struct node *temp = FindMin(r->right);
r->value= temp->value;
r->right = Delete(r->right, temp->value);
}
}
return r;
}
struct node * FindMin(struct node *root)
{
if (root->left != NULL)
{
return FindMin(root->left);
}
return root;
}
```

OUTPUT:

ASSIGNMENT -5:

1) Write a C program to implement simple queue(Insertion, deletion and traversal)

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

#define max 10

int f=-1;

int r=-1;

int queue[max];

void enqueue(int);

int dequeue();

void display();

void main()

{

int ch,val,ans;

while(1)

{

printf("\n 1)ENQUEUE 2)DEQUEUE 3)DISPLAY");

scanf("%d",&ch);

switch(ch)

{

case 1:

printf("\n Enter the value to be inserted:");

scanf("%d",&val);
enqueue(val);

break;
```

```
case 2:
ans=dequeue();
printf("\nElement poped from stack = %d",ans);
break;
case 3: display();
break;
}
}
}
void enqueue(int x)
{
        if(r==(max-1))
        {
                printf("QUEUE IS IN OVERFLOWW CONDITION");
        }
        else
        {
                if(f==-1)
                {
                  f++;
                }
        r++;
        queue[r]=x;
        }
}
```

```
int dequeue()
{
int y;
if(r==-1 || f>r)
{
printf("QUEUE IS IN UNDERFLOWW CONDITION");
return 0;
}
y=queue[f];
f++;
return y;
}
void display()
{
int i;
if(r==-1 || f>r)
{
printf("QUEUE IS IN UNDERFLOWW CONDITION");
}
Else
{
        for(i=f;i<=r;i++)
        {
        printf("\n%d\n",queue[i]);
        }
}
}
```

OUTPUT:

2) Write a C program to implement circular queue (Insertion, Deletion and Traversal).

```
#include<stdio.h>
#include<conio.h>
#define max 5

int f=-1;
int r=-1;

int queue[max];
void enqueue(int);
int dequeue();
void display();

void main()
{
int ch,val,ans;

while(1)
{
printf("\nENTER YOUR CHOICE:\n");
printf("\n 1)ENQUEUE 2)DEQUEUE 3)DISPLAY 4)EXIT");
```

```
scanf("%d",&ch);
switch(ch)
{
case 1:
printf("\n Enter the value to be inserted:");
scanf("%d",&val); enqueue(val); break;

case 2:
ans=dequeue();
printf("\nElement deleted from Queue = %d",ans);
break;

case 3:
display();
break;

case 4:
exit(0);
break;
}
}
}
```

```
void enqueue(int x)
{
	if(f==(r+1)%max)
	{
		printf("QUEUE IS IN OVERFLOWW CONDITION");
	}
	else
	{
		if(f==-1)
		{
		f=r=0;
		}
		else
		{
		r=(r+1)%max;
		}
	queue[r]=x;
	}
}
int dequeue()
{
int y;
	if(f==-1)
	{
	printf("QUEUE IS IN UNDERFLOWW CONDITION");
	return 0;
```

```
        }
        else
        {
                y=queue[f];
                if(f==r)
                {
                        f=r=-1;
                }
                else
                {
                        f=(f+1)%max;
                }
        return y;
        }
}
void display()
{
int i;
        if(f==-1)
        {
        printf("QUEUE IS IN UNDERFLOWW CONDITION");
        return;
        }
        else
        {
```

```
            if(r>=f)
            {
                  for(i=f;i<=r;i++)
                  {
                  printf("\n%d\n",queue[i]);
                  }
            }
            else
            {
                  for(i=f;i<max;i++)
                  {
                  printf("\n%d\n",queue[i]);
                  }
                  for(i=0;i<=r;i++)
                  {
                  printf("\n%d\n",queue[i]);
                  }
            }
      }
}
```

OUTPUT:

3) Write a C program to implement Priority queue(Insertion, deletion and traversal)

```
#include<stdio.h>
#include<conio.h>
#define max 10
int f=-1;
int r=-1;
int queue[max];
void enqueue_priority(int);
void check(int data);
int dequeue();
void display();
void main()
{

int ch,val,ans;
while(1)
{
printf("\n 1)ENQUEUE 2)DEQUEUE 3)DISPLAY 4)EXIT : ");
scanf("%d",&ch);
switch(ch)
{
case 1:
printf("\n Enter the value to be inserted:");
scanf("%d",&val); enqueue_priority(val);
break;
```

```
case 2:
ans=dequeue();
printf("\n Element deleted from Queue= %d",ans);
break;
case 3:
display();
break;
case 4:
exit(0);
break;
}
}
}

void enqueue_priority(int x)
{
        if(r==(max-1))
        {
        printf("QUEUE IS IN OVERFLOWW CONDITION");
        }
        else
        {
                if(f==-1)
                {
                f++; r++;
                queue[r]=x;
                }
```

```
                else
                {
                r++;
                check(x);
                }
        }
}
void check(int data)
{
int i,j;
for(i=0;i<=r;i++)
{
        if(data>queue[i])
        {
                for(j=r+1;j>i;j--)
                {
                queue[j]=queue[j-1];
                }
                queue[i]=data;
                return;
           }
}
queue[i]=data;
}
```

```
int dequeue()
{
int y;
if(r==-1 || f>r)
{
printf("QUEUE IS IN UNDERFLOWW CONDITION");
return 0;
}
y=queue[f];
f++;
return y;
}
void display()
{
int i;
if(r==-1 || f>r)
{
printf("QUEUE IS IN UNDERFLOWW CONDITION");
}
else
{
for(i=f;i<=r;i++)
{
printf("\n%d\n",queue[i]);
}
}
}
```

OUTPUT:

ASSIGNMENT -6:

1) Write a C Program to organize 10 integer values in ascending order by repeatedly finding the minimum element and putting it at the beginning of list.

```
#include<stdio.h>
#include<conio.h>
void main()
{
int n,i,j,min_index,temp;

printf("\nHOW MANY ELEMENTS DO YOU WANT TO ENTER IN ARRAY?");
scanf("%d",&n);
int a[n];
printf("ENTER ARRAY ELEMENTS:\n");
for(i=0;i<n;i++)
{
scanf("%d",&a[i]);
}
for(i=0;i<n;i++)
{
min_index=i;
for(j=i+1;j<n;j++)
{
if(a[j]<a[min_index])
{
        min_index=j;
}
```

```
}
if(min_index != i)
{
temp=a[min_index];
a[min_index]=a[i];
a[i]=temp;
}
}
printf("\nARRAY AFTER SORTING IS AS FOLLOWS:\n");
for(i=0;i<n;i++)
{
printf("%d\n",a[i]);
}
return;
}
```

OUTPUT:

2) Write a C Program to organize 10 integer values in ascending order by repeatedly swapping the adjacent elements if they are in wrong order.

```
#include<stdio.h>
#include<conio.h>
void main()
{
int n,i,j,count,temp;
printf("\nHOW MANY ELEMENTS DO YOU WANT TO ENTER IN ARRAY? ");
scanf("%d",&n);
int a[n];
printf("ENTER ARRAY ELEMENTS:");
for(i=0;i<n;i++)
{
scanf("%d",&a[i]);
} count=0; for(i=0;i<n;i++)
{
for(j=0;j<n-1;j++)
{
if(a[j]>a[j+1])
{
temp=a[j]; a[j]=a[j+1];
a[j+1]=temp; count++;
}
}
if(count==0)
{
```

```
printf("ARRAY IS ALREADY SORTED.");
return;
}
}
printf("\nARRAY AFTER SORTING IS AS FOLLOWS:\n");
for(i=0;i<n;i++)
{
printf("%d\n",a[i]);
}
}
```

OUTPUT:

3) Write a C Program to organize 10 integer values in ascending order by divide and conquer technique. Divide the input array in two halves, calls itself for the two halves and then merges the two sorted halves.

```
#include<stdio.h>
#include<conio.h>
void mergesort(int a[], int f ,int l);
void merge(int a[], int i1 ,int j1 , int i2, int j2);
void main()
{
int n,i,j,first_index,last_index;

printf("\nHOW MANY ELEMENTS DO YOU WANT TO ENTER IN ARRAY?");
scanf("%d",&n); first_index=0;
last_index=n-1; int a[50];
printf("ENTER ARRAY ELEMENTS:");
for(i=0;i<n;i++)
{
scanf("%d",&a[i]);
}
mergesort(a,first_index,last_index);
printf("SORTED ARRAY IS:\n");
for(i=0;i<n;i++)
{
        printf("%d\n",a[i]);
}
}
```

```
void mergesort(int a[], int f ,int l)
{
int mid;
        if(f<l)
        {
        mid=(f+l)/2;
        mergesort(a,f,mid);
        mergesort(a,mid+1,l);
        merge(a,f,mid,mid+1,l);
        }
}
void merge(int a[], int i1 ,int j1 , int i2, int j2)
{
int temp[50];
int i,j,k;
i=i1;
j=i2;
k=0;
while(i<=j1 && j<=j2)
{
        if(a[i]<a[j])
        {
        temp[k]=a[i];
        k++;
        i++;
        }
```

```
        else
        {
        temp[k++]=a[j++];
        }
}
while(i<=j1)
{
temp[k++]=a[i++];
}
while(j<=j2)
{
temp[k++]=a[j++];
}
for(i=i1,j=0;i<=j2;i++,j++)
{
a[i]=temp[j];
}
}
```

OUTPUT

3) Write a C program to implement QUICK sort.

```
#include<stdio.h>

#include<conio.h>

void main()

{

int n,i;

printf("\nHOW MANY ELEMENTS DO YOU WANT TO ENTER IN ARRAY? ");

scanf("%d",&n);
int a[n];

printf("ENTER ARRAY ELEMENTS:");

for(i=0;i<n;i++)

{

scanf("%d",&a[i]);

}

quicksort(a,0,n-1);

printf("SORTED ARRAY IS:\n");

for(i=0;i<n;i++)

{

printf("%d\n",a[i]);

}

}
```

```
void quicksort(int array[], int first, int last)
{
int i, j, pivot, temp;

if(first<last)
{

        pivot=first;

        i=first; j=last;

        while(i<j)

        {
                while(array[i]<=array[pivot] && i<last)
                        i++;

                while(array[j]>array[pivot])
                        j--;
        if(i<j)
        {
        temp=array[i];

        array[i]=array[j];

        array[j]=temp;

        }
}

temp=array[pivot];

array[pivot]=array[j];

array[j]=temp;

quicksort(array,first,j-1);

quicksort(array,j+1,last);

}

}
```

OUTPUT

ASSIGNMENT -7:

1) Write a C program to implement DFS graph traversal.

```
#include<stdio.h>
#include<conio.h>
int G[10][10], visited[10],v;
void main()
{
int i,j,x;
printf("\nEnter the vertex size : ");
scanf("%d",&v);
printf("Enter the elements of adjacency matrix: ");
for(i=0;i<v;i++)
{
        for(j=0;j<v;j++)
        {
                scanf("%d",&G[i][j]);
        }
}
printf("Enter the element from which,you want to do DFS Traversal:");
scanf("%d",&x);
printf("\nDFS traversal of Graph from %d is:",x);
dfs(x);
}
```

```
void dfs(int i)
{
int j;
visited[i]=1;
for(j=0;j<v;j++)
{
        if(G[i][j]==1 && visited[j]!=1)
        {
                dfs(j);
                printf("%d",j);
        }
}
}
```

OUTPUT:

2) Write a C program to implement BFS graph traversal

```
#include<stdio.h>
#include<conio.h>
int G[10][10],visited[10],visit[10],n,v;
void main()
{
int i,j,k,r=0,f=0;
int queue[10];
printf("\nEnter the vertex size: ");
scanf("%d",&n);
printf("Enter the elements of adjacency matrix: ");
for(i=0;i<n;i++)
{
        for(j=0;j<n;j++)
        {
                scanf("%d",&G[i][j]);
        }
}
printf("Enter the initial vertex:");
scanf("%d",&v);
visited[v]=1;
k=1;
```

```
printf("\nBFS traversal of Graph from %d vertex is: ",v);
while(k<n)
{
for(j=0;j<n;j++)
{
        if(G[v][j]!=0 && visited[j]!=1 && visit[j]!=1)
        {
                visit[j]=1;
                queue[r++]=j;
        }
}

v=queue [f++];
printf("%d",v);
k++;
}

}
```

OUTPUT:

3) Write a C program to implement Prim's algorithm to find MST from given graph.

```
#include<stdio.h>
#include<conio.h>
int G[10][10],visited[10];
void main()
{
int n,a=0,b=0,i,j,ne=1,min=999,mincost=0;
printf("\nTotal vertices:");
scanf("%d",&n);
printf("Enter the elements of adjacency matrix: ");
for(i=0;i<n;i++)
{
        for(j=0;j<n;j++)
        {
                scanf("%d",&G[i][j]);
                if(G[i][j]==0)
                {
                        G[i][j]=999;
                }
        }
}
visited[0]=1;
```

```
while(ne<n)
{
min=999;
        for(i=0;i<n;i++)
        {
                for(j=0;j<n;j++)
                {
                        if(G[i][j]< min && visited[i]!=0)
                        {
                        min=G[i][j];
                        a=i;
                        b=j;
                        }
                }
        }
if(visited[b] == 0)
{
visited[b]=1;
mincost+=min;
printf("\n Edge %d :(%d to %d) and cost=%d",ne++,a,b,min);
}
G[a][b]=G[b][a]=999;
}
printf("\nFinal cost of MST is %d",mincost);
}
```

OUTPUT:

4) Write a C program to implement Kruskal's algorithm to find MST from given graph.

```
#include<stdio.h>
#include<conio.h>
int G[5][5],parent[5];
int min,mincost=0;
void kruskal(int array[5][5]);
int find(int );
void merge(int,int);
void main()
{
int i,j;
printf("\nEnter the elements of adjacency matrix: ");
for(i=0;i<5;i++)
{
for(j=0;j<5;j++)
        {
                scanf("%d",&G[i][j]);
                if(G[i][j]==0)
                {
                        G[i][j]=999;
                }
        }
}
kruskal(G);
}
```

```
void kruskal(int G[5][5])
{
int i,edge_count=0,j,a,b;
for(i=0;i<5;i++)
{
parent[i]=i;
}
while(edge_count<4)
{
min=999;
        for(i=0;i<5;i++)
        {
                for(j=0;j<5;j++)
                {
                        if(G[i][j]< min && find(i)!=find(j))
                       {
                        min=G[i][j];
                        a=i;
                        b=j;
                        }
                }
}
merge(a,b);
printf("\n%d (EDGE FROM %d to %d)%d\n",edge_count++,a,b,min);
mincost+=min;
}
printf("\nMINCOST=%d",mincost);
}
```

```
int find(int i)
{
while(parent[i]!= i)
{
        i=parent[i];
}
return i;
}
void merge(int i,int j)
{
int a=find(i);
int b=find(j);
parent[a]=b;    //parent[b]=a
}
```

OUTPUT:

ASSIGNMENT -8:

1) Write a C program to implement Singly Linked list for the following functions.

(a) Insert a node at the front of the linked list.
(b) Insert a node at the end of the linked list.
(c) Insert a node in order
(d) Delete any node from the linked list
(e) Count total no. of nodes.
(f) Insert a node at any position in linked list.
(g)Display all nodes
(h) Copy list

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

struct node
{

int info;

struct node *link;

}*first=NULL, *new=NULL, *temp=NULL, *temp2=NULL;

struct node2

{

int info2;

struct node *link2;

}*begin=NULL, *new2=NULL, *t=NULL, *t2=NULL;


struct node * insert_first(int x);

struct node * insert_last(int x);

struct node * insert_in_order(int x);

void display(struct node *);

struct node * insert_position(int x,int y);

void delete(int data);

int count();
```

```
void copy();

void main()

{

int n,x,a,b,value,ans=0;

while(1)
{

printf("\n(1) INSERTING NODE AT FIRST  \n(2) INSERTING NODE AT LAST \n(3) INSERTING NODE IN_BETWEEN(IN_ORDER) \n(4) DISPLAY ALL ELEMENTS: \n(5) INSERTING NODE AT SPECIFIC POSITION \n(6) DELETING A NODE \n(7) COUNTING TOTAL NODES \n(8) COPYING THE WHOLE LINKED LIST");

printf("\n\nENTER YOUR CHOICE:");

scanf("%d",&n);

switch(n)

{

case 1:

printf("Enter the element you want to insert at first:");

scanf("%d",&x); first=insert_first(x);

break;

case 2:

printf("Enter the element you want to insert at last:");

scanf("%d",&x); first=insert_last(x);

break;

case 3:

printf("Enter the element you want to insert in middle:");

scanf("%d",&x); first=insert_in_order(x);

break;

case 4:
```

```
printf("Elements are as follows:\n");
display(first);
break;
case 5:
printf("Enter position where you want to insert element:");
scanf("%d",&a);
printf("Enter the element you want to insert element:");
scanf("%d",&b); first=insert_position(a,b);
break;
case 6:
printf("Enter the element you want to delete:");
scanf("%d",&value);
delete(value);
break; case 7: ans=count();
printf("There are total %d nodes.",ans);
break;
case 8:
copy();
break;
}
}
}
```

```
struct node * insert_first(int x)
{
new=(struct node *) malloc(sizeof(struct node));
new->info=x; new->link= first;
return new;
}
struct node * insert_last(int x)
{
new=(struct node *) malloc(sizeof(struct node));
new->info=x;
new->link= NULL;
if(first==NULL)
{
return new;
}
else
{
temp=first;
while(temp->link!= NULL)
{
temp=temp->link;
}
temp->link=new;
return first;
}
}
```

```
struct node * insert_in_order(int x)
{
new=(struct node *) malloc(sizeof(struct node));
new->info=x;
new->link= NULL;
if(first==NULL)
{
new->link=NULL;
return new;
}
else
{
if(new->info<=first->info)
{
new->link=first;
return new;
}
else
{
temp=first;
while(new->info > temp->info && temp->link != NULL)
{
temp2=temp;
temp=temp->link;
}
if(new->info > temp->info)
```

```
{
temp2=temp;
}
new->link=temp2->link;
temp2->link=new;
return first;
}
}
}
void display(struct node *temp)
{
temp=first;
if(first==NULL)
{
printf("LINK IS EMPTY.");
}
while(temp != NULL)
{
printf("%4d --> %6d",temp->info,temp->link);
temp=temp->link;
}
}
struct node * insert_position(int x,int y)
{
int i=0;
new=(struct node *) malloc(sizeof(struct node));
```

```
new->info=y; temp=first;
if(x==1)
{
new->link=first; first=new; return
first;
}
while(i<x-1)
{
temp2=temp; temp=temp->link; i++;
}
temp2->link=new;
new->link=temp;
return first;
}
void delete(int data)
{
if(first==NULL)
{
printf("\nUNDERFLOWW CONDITION");
return;
}
temp=first;
while(temp->info!= data)
{
temp2=temp; temp=temp->link;
if(temp== NULL)
```

```
{
printf("Value %d is not present in the linked list",data);
return;
}
}
if(first->info==data)
{
temp=first;
first=first->link;
free (temp);
return;
}
else
{
temp2->link=temp->link;
free(temp);
}
}
int count()
{
int count=0; temp=first;
if(first==NULL)
{
return count;
}
while(temp->link != NULL)
```

```
{
count++;
temp=temp->link;
}
if(temp->link == NULL)
count++;
return count;
}
void copy()
{
if(first==NULL && begin==  NULL)
{
printf("LINK IS EMPTY.");
return;
} begin=first; temp=first;
t=begin;
while(temp->link!= NULL)
{
t=(struct node *) malloc(sizeof(struct node));
t->info2=temp->info;     t->link2=temp-
>link; temp=temp->link;
t=t->link2;
}

if(temp->link==NULL)
{
t=(struct node *) malloc(sizeof(struct node));
```

```
t->info2=temp->info;
t->link2=temp->link;
}
t=begin;
printf("\nCOPIED LINKED LIST IS AS FOLLOWS:\n");
while(t != NULL)
{
printf("%4d --> %6d",t->info2,t->link2);
t=t->link2;
}
}
```

OUTPUT:

2) Write a C program to implement Circular Linked list.

(a) Insert a node at the front of the linked list.

(b) Insert a node at the end of the linked list.

(c) Insert a node in order.

(d) Delete any node from the linked list.

(e) Count total no of nodes in list.

(f) Display all nodes (traversal of Linked list).

```
#include<stdio.h>
#include<conio.h>
struct node
{
int info;
struct node *link;
}*first=NULL,*last=NULL, *new=NULL, *temp=NULL,*temp2=NULL;
struct node * insert_first(int x);
struct node * insert_end(int x);
struct node * insert_inorder(int x);
void display(struct node *temp);
int count();
void delete(int x);
void main()
{
int n,x,a,b,value,ans=0;
```

```
printf("NAME:- VVP\n"); printf("ENROLLMENT_NUMBER:- --\n");

printf("BATCH:- --\n");

while(1)

{

printf("\n(1) INSERTING NODE AT FIRST  \n(2) INSERTING NODE AT LAST
\n(3) INSERTING NODEIN_BETWEEN(IN_ORDER) \n(4) DISPLAY ALL
ELEMENTS  \n(5) COUNTING NUMBER OF NODES \n(6) DELETING A NODE");

printf("\n\nENTER YOUR CHOICE:");

scanf("%d",&n);

switch(n)

{

case 1:

printf("Enter the element you want to insert at first:");

scanf("%d",&x); first=insert_first(x);

break;

case 2:

printf("Enter the element you want to insert at last:");

scanf("%d",&x); first=insert_end(x);

break;

case 3:

printf("Enter the element you want to insert in middle:");

scanf("%d",&x); first=insert_inorder(x);

break;

case 4: display(first); break;

case 5:

ans=count();
```

```
printf("There are total %d nodes.",ans);
break;
case 6:
printf("Enter the element you want to delete:");
scanf("%d",&value);
delete(value);
break;
}
}
}
struct node * insert_first(int x)
{
new=(struct node *) malloc(sizeof(struct node));
new->info=x;
if(first==NULL)
{
                new->link=new;
                first=last=new;
}
else
{
new->link=first;
last->link=new;
first=new;
}
return first;
```

```
}
struct node * insert_end(int x)
{
new=(struct node *) malloc(sizeof(struct node));
new->info=x;
if(first==NULL)
{
new->link=new;
first=last=new;
}
else
{
new->link=first;
last->link=new;
last=new;
}
return first;
}
struct node * insert_inorder(int x)
{
new=(struct node *) malloc(sizeof(struct node));
new->info=x;
if(first==NULL)
{
new->link=new;
first=last=new;
```

```
}
else
{
if(new->info<=first->info)
{
new->link=first;
last->link=new; first=new;
}
else
{
temp=first;
while(new->info>=temp->info && temp != last)
{
temp2=temp;
temp=temp->link;
}
if(new->info>last-
>info)
{
new->link=first;
last->link=new;
last=new;
return first;
}
temp2->link=new;
new->link=temp;
```

```
return first;
}
}
}
void display(struct node *temp)
{
temp=first;
if(first==NULL && last==NULL)
{
printf("LINKED LIST IS EMPTY.NO NODES ARE PRESENT IN THE LIST");
return;
}
printf("Elements are as follows:\n");
while(temp != NULL && temp!= last)
{
printf("%4d --> %6d",temp->info,temp->link);
temp=temp->link;
}
printf("%4d --> %6d",temp->info,temp->link);
}
int count()
{
int count=0; temp=first;
if(first==NULL)
{
return count;
}
```

```
while(temp->link != NULL && temp!=last)
{
count++;
temp=temp->link;
}
if(temp==last) count++;
return count;
}
void delete(int x)
{
if(first==NULL && last == NULL)
{
printf(" CIRCULAR LINKED LIST IS EMPTY.");
return;
}
if(first->info==x && last->info==x)
{
temp=first; first=last=NULL;
free(temp); free(first); free(last);
return;
}
temp=first;
while(temp->info != x && temp!=last)
{
temp2=temp;
temp=temp->link;
```

```
}
if(temp ->info != x)
{
printf("Value %d is not present in the linked list",x);
return;
}
if(first->info==x)
{
temp=first; first=first->link; last-
>link=first; free(temp); return;
}
if(last->info==x)
{
last=temp2;
last->link=first; free(temp);
return;
}
else
{
temp2->link=temp->link;
free(temp);
return;
}
}
```

OUTPUT:

2) Write a C program to implement Doubly Linked list.

```
#include<stdio.h>
#include<conio.h>
struct node
{
int info;
struct node *next;
struct node *previous;
}*head=NULL,*temp=NULL, *new=NULL, *temp2=NULL,*tail=NULL;
struct node * insert_first(int x);
struct node * insert_end(int x);
struct node * insert_in_order(int x);
void forward_display(struct node* temp);
void backward_display(struct node* temp);
void delete(int y);
void main()
{
int n,x,a,b,value,ans=0;
while(1)
{
printf("\n(1) INSERTING NODE AT FIRST  \n(2) INSERTING NODE AT LAST \n(3) FORWARD TRAVERSALS OF ALL ELEMENTS \n(4) REVERSE TRAVERSALS OF ALL ELEMENTS\n(5) INSERTING NODE IN ORDER \n(6) DELETING A NODE");
printf("\n\nENTER YOUR CHOICE:");
scanf("%d",&n);
```

```
switch(n)
{
case 1:
printf("Enter the element you want to insert at first:");
scanf("%d",&x); head=insert_first(x);
break;
case 2:
printf("Enter the element you want to insert at last:");
scanf("%d",&x); head=insert_end(x);
break;
case 3:
forward_display(head);
break; case 4: backward_display(tail);
break;
case 5:
printf("Enter the element you want to insert in middle:");
scanf("%d",&x); head=insert_in_order(x);
break;
case 6:
printf("Enter the element you want to delete:");
scanf("%d",&x);
delete(x);
break;
}
}
}
```

```
struct node * insert_first(int x)
{
new=(struct node *) malloc(sizeof(struct node));
new->info=x;
if(head==NULL && tail== NULL)
{
new->next=NULL;
new->previous=NULL;
head=new;
tail=new;
head->next=NULL; head->previous=NULL;
tail->next=NULL;
tail->previous=NULL;
return new;
}
else
{
head->previous=new;
new->next=head;
new->previous=NULL;
head=new;
head->previous=NULL;
return new;
}
}
```

```
struct node * insert_end(int x)
{
new=(struct node *) malloc(sizeof(struct node));
new->info=x;
if(head==NULL && tail== NULL)
{
new->next=NULL;
new->previous=NULL;
head=new;
tail=new;
head->next=NULL; head->previous=NULL;
tail->next=NULL;
tail->previous=NULL;
return new;
}
else
{
tail->next=new;
new->previous=tail; new->next=NULL;
tail=new;
tail->next=NULL;
return head;
}
}
```

```
void forward_display(struct node *temp)
{
temp=head;
if(head==NULL && tail==NULL)
{
printf("DOUBLY LINKED LIST IS EMPTY.");
}
while(temp!=NULL)
{
printf(" %6d --> %4d --> %6d\n",temp->previous,temp->info,temp->next);
temp=temp->next;
}
}
void backward_display(struct node* temp)
{
temp=tail;
if(head==NULL && tail==NULL)
{
printf("DOUBLY LINKED LIST IS EMPTY.");
}
while(temp!=NULL)
{
printf(" %6d --> %4d --> %6d\n",temp->previous ,temp->info,temp->next);
temp=temp->previous;
}
}
```

```
struct node * insert_in_order(int x)
{
new=(struct node *) malloc(sizeof(struct node));
new->info=x;
if(head==NULL && tail==NULL)
{
new->next=NULL;
new->previous=NULL;
head=new;
tail=new;
head->next=NULL; head->previous=NULL;
tail->next=NULL;
tail->previous=NULL;
return new;
}
else
{
if(new->info<=head->info)
{
        head->previous=new;
new->next=head;
new->previous=NULL;
head=new;
head->previous=NULL;
return new;
}
```

```
else
{
temp=head;
while(new->info > temp->info && temp->next != NULL)
{
temp2=temp;
temp=temp->next;
}
if(new->info > temp->info)
{
temp->next=new;
new->previous=temp;
tail=new;
tail->next=NULL;
return head;
}
new->next=temp;
temp2->next=new;
new->previous=temp2;
temp->previous=new;
return head;
}
}
}
```

```
void delete(int y)
{
if(head==NULL && tail== NULL)
{
printf("list is empty");

            return;
}

temp = head;

while(temp != NULL)
{
if(temp->info == y)

{
        if(temp== head)
              {
                head=head->next;

                head->previous=NULL;
              }

        else if(temp==tail)
              {
                tail=tail->previous;
                tail->next=NULL;
                }
                else
                {
                temp->previous->next = temp->next;

                temp->next->previous=temp->previous;

                }

free (temp);

}

temp=temp->next;

}
printf("\ndata not found");
}
```

OUTPUT:

ASSIGNMENT -9:

1)Write a C program to implement stack using Linked list .

```
#include<stdio.h>
#include<conio.h>
struct node
{
int info;
struct node *link;
}*top=NULL, *new=NULL, * temp=NULL;
void push(int x);
int pop();
void display();
void main()
{
int ch,val,ans=0;
while(1)
{
printf("\n 1)PUSH 2)POP 3)DISPLAY");
scanf("%d",&ch);
switch(ch)
{
case 1:
printf("\n Enter the value to be inserted:");
scanf("%d",&val);
push(val);
break;
```

```
case 2:
ans=pop();
printf("\nElement poped from stack = %d",ans);
break;
case 3:
display();
break;
}
}
}
void push(int x)
{
new=(struct node *) malloc(sizeof(struct node));
new->info=x; new->link=top;
top=new;
}
int pop()
{
int y;
if(top==NULL)
{
printf("STACK IS EMPTY");
return 0;
}
else
{
temp=top; y=top->info; top=top->link; free(temp);
```

```
return y;
}
}
void display()
{
temp=top;
if(top==NULL)
{
printf("STACK IS EMPTY");
return;
}
printf("Elements in stack are:");
while(temp != NULL)
{
printf("%4d --> %6d",temp->info,temp->link);
temp=temp->link;
}
return;
}
```

OUTPUT:

2) Write a C program to implement Queue using linked list.

```
#include<stdio.h>
#include<conio.h>
struct node
{
int info;
struct node *link;
}*first=NULL, *last=NULL, *new=NULL, * temp=NULL;
void enqueue(int x);
int dequeue();
void display();
void main()
{
int ch,val,ans=0;

printf("BATCH:- --\n");
while(1)
{
printf("\n 1)ENQUEUE 2)DEQUEUE 3)DISPLAY");
scanf("%d",&ch);
switch(ch)
{
case 1:
printf("\n Enter the value to be inserted:");
```

```
scanf("%d",&val); enqueue(val);
break;
case 2:
ans=dequeue();
printf("\nElement deleted from queue= %d",ans);
break;
case 3:
display();
break;
}
}
}
void enqueue(int x)
{
new=(struct node *) malloc(sizeof(struct node));
new->info=x;
if(first==NULL)
{
first=last=new;
new->link=NULL;
}
last->link=new;
last=new;
last->link=NULL;
}
int dequeue()
```

```
{
int y;
if(first==NULL)
{
printf("UNDERFLOWW CONDITION");
return 0;
} temp=first; y=temp->info;
first=first->link; free(temp);
return y;

}
void display()
{
temp=first;
if(first==NULL)
{
printf("UNDERFLOWW CONDITION");
return;
}
printf("Elements in Queue are:");
while(temp->link!= NULL)
{
printf("\n%d",temp->info);
temp=temp->link;
}
printf("\n%d",temp->info);
}
```

OUTPUT:

2) Write a C program to implement double ended queue using linked list .

```
#include<stdio.h>
#include<conio.h>
struct node
{
int info;
struct node *link;
}*first=NULL, *last=NULL, *new=NULL, *temp=NULL,*temp2=NULL;
void enqueue(int x);
int dequeue();
void display();
void main()
{
int n,ch,val,ans=0;
while(1)
{
printf("\n1)OPERATE FROM FRONT \n2)OPERATE FROM REAR \n3)DISPLAY
QUEUE: ");
scanf("%d",&n);
switch(n)
{
case 1:
printf("\n 1)ENQUEUE 2)DEQUEUE "); printf("\nEnter your
choice: "); scanf("%d",&ch);
switch(ch)
{
```

```
case 1:
printf("\n Enter the value to be inserted: ");
scanf("%d",&val); front_enqueue(val);
break;
case 2:

ans=front_dequeue();
printf("\nElement poped from stack = %d\n",ans);
break;
} break; case 2:
printf("\n 1)ENQUEUE 2)DEQUEUE"); printf("\nEnter your
choice: "); scanf("%d",&ch);
switch(ch)
{
case 1:
printf("\n Enter the value to be inserted: ");
scanf("%d",&val); rear_enqueue(val);
break;
case 2:
ans=rear_dequeue();
printf("\nElement poped from stack = %d\n",ans);
break;
} break; case 3:
display();

break;
```

```
}
}
}
void front_enqueue(int x)
{
new=(struct node *) malloc(sizeof(struct node));
new->info=x;
if(first==NULL)
{
first=last=new; new->link=NULL;
return;
}
new->link=first;
first=new;
}
void rear_enqueue(int x)
{
new=(struct node *) malloc(sizeof(struct node));
new->info=x;
if(first==NULL)
{
first=last=new;
new->link=NULL;
return;
}
last->link=new;
```

```
last=new;
last->link=NULL;
}
int front_dequeue()
{
int y;
if(first==NULL)
{
printf("\nUNDERFLOWW CONDITION \n");
        return 0;
} temp=first; y=temp->info;
first=first->link; free(temp);
return y;
}
int rear_dequeue()
{
int y;
if(first==NULL)
{
printf("\nUNDERFLOWW CONDITION \n");
return 0;
}
if(first==last)
{
y=first->info; free(first); free(last);
first=last=NULL; return y;
```

```
}
temp=first;
while(temp->link!=NULL)
{
temp2=temp;
temp=temp->link;
}
y=temp->info;
last=temp2;
last->link=NULL;
free(temp);
return y;
}
void display()
{
temp=first;
if(first==NULL)
{
printf("\nUNDERFLOWW CONDITION\n");
return;
}
printf("Elements in stack are:");
while(temp!= NULL)
{
printf("\n%d",temp->info);
temp=temp->link;
}
}
```

OUTPUT:

3)Write a C program to swap max & min element from singly linked list.

```
#include<stdio.h>

#include<conio.h>

struct node

{

int info;

struct node *link;
}*first=NULL, *new=NULL, *temp=NULL, *temp2=NULL,*minnode=NULL,
*maxnode=NULL;

struct node * insert_first(int x); struct node * insert_last(int x);

void display(struct node *);

void swap_print(); void

delete(int data);

void main()

{

int n,x,a,b,value,ans=0;

while(1)

{

printf("\n(1) INSERTING NODE AT FIRST  \n(2) INSERTING NODE AT LAST
\n(3) DISPLAY ALL ELEMENTS (BEFORE SWAPPING): \n(4) DELETING A NODE
\n(5) DISPLAY ALL ELEMENTS (AFTER SWAPPING): ");

printf("\n\nENTER YOUR CHOICE:");

scanf("%d",&n);

switch(n)

{

case 1:

printf("Enter the element you want to insert at first:");
```

```
scanf("%d",&x); first=insert_first(x); break;
case 2:
printf("Enter the element you want to insert at last:");
scanf("%d",&x); first=insert_last(x);
break;
case 3:
printf("Elements are as follows:\n");
display(first);
break;
case 4:
printf("\nEnter the element you want to delete:");
scanf("%d",&value);
delete(value);
break;
case 5:
printf("\nElements  after swapping, are as follows:\n");
swap_print();
break;
}
}
}
struct node * insert_first(int x)
{
new=(struct node *) malloc(sizeof(struct node));
new->info=x; new->link= first;
return new;
```

```
}
struct node * insert_last(int x)
{
new=(struct node *) malloc(sizeof(struct node));
new->info=x;
new->link= NULL;
if(first==NULL)
{
return new;
}
else
{
temp=first;
while(temp->link!= NULL)
{
temp=temp->link;
}
temp->link=new;
return first;
}
}
void display(struct node *temp)
{
temp=first;
if(first==NULL)
{
```

```
printf("LINK IS EMPTY.");
}
while(temp != NULL)
{
printf("%4d --> %6d",temp->info,temp->link);
temp=temp->link;
}
}
void delete(int data)
{
if(first==NULL)
{
printf("\nUNDERFLOWW CONDITION");
return;
}
temp=first;
while(temp->info!= data)
{
temp2=temp; temp=temp->link;
if(temp== NULL)
{
printf("Value %d is not present in the linked list",data);
return;
}
}
if(first->info==data)
```

```
{
temp=first; first=first->link; free
(temp); return;
}
else
{
temp2->link=temp->link;
free(temp);
}
}
void swap_print()
{
int min=999,max=0,swap;
//min=max=first->info;
if(first==NULL)
{
printf("LINK IS EMPTY.");
return;
} temp=first; while(temp != NULL)
{
if(temp->info<=min)
{
                min=temp->info;
               minnode=temp;
}
if(temp->info>=max)
```

```
{
                max=temp->info;
               maxnode=temp;
}
temp=temp->link;
}
swap=minnode->info;
minnode->info=maxnode->info; maxnode->info=swap; temp=first;
while(temp != NULL)
{
printf("%4d --> %6d",temp->info,temp->link);
temp=temp->link;
}
}
```

OUTPUT:

4) Write a C program to remove duplicate values from linked list.

```
#include<stdio.h>

#include<conio.h>

struct node

{

int info;

struct node *link;
}*first=NULL, *ptr=NULL, *delete=NULL, *new=NULL,*temp=NULL,
*temp2=NULL;

struct node * insert_first(int x);
struct node * insert_last(int x);
void display(struct node *);
void distinct_print();

void main()

{

int n,x,a,b,value,ans=0;

while(1)

{

printf("\n(1) INSERTING NODE AT FIRST  \n(2) INSERTING NODE AT LAST \n(3)
DISPLAY ALL ELEMENTS (WITH REPITION)\n(4) DISPLAY ALL ELEMENTS
WITHOUT REPITION) :");

printf("\n\nENTER YOUR CHOICE:");

scanf("%d",&n);

switch(n)

{

case 1:

printf("Enter the element you want to insert at first:");
```

```
scanf("%d",&x);
first=insert_first(x);
break;
case 2:
printf("Enter the element you want to insert at last:");
scanf("%d",&x); first=insert_last(x);
break;
case 3:
printf("Elements are as follows:\n");
display(first);
break;
case 4:
printf("\nElements, after removing repeated elements are as follows:\n");
distinct_print();
break;
}
}
}
struct node * insert_first(int x)
{
new=(struct node *) malloc(sizeof(struct node));
new->info=x; new->link= first;
return new;
}
```

```
struct node * insert_last(int x)
{
new=(struct node *) malloc(sizeof(struct node));
new->info=x;
new->link= NULL;
if(first==NULL)
{
//last=new;
return new;
}
else
{
temp=first;
while(temp->link!= NULL)
{
temp=temp->link;
}
temp->link=new;
// last=new;
return first;
}
}
void display(struct node *temp)
{
temp=first;
if(first==NULL)
```

```
{
printf("LINK IS EMPTY.");
}
while(temp != NULL)
{
printf("%4d --> %6d",temp->info,temp->link);
temp=temp->link;
}
}
void distinct_print()
{
temp=first;
while(temp->link != NULL)
{
ptr=temp->link;
while(ptr->link != NULL)
{
if(ptr->info == temp->info)
{
delete=ptr;
temp2->link=ptr->link->link;
ptr=ptr->link;
free(delete);
}
else
{
```

```
temp2=ptr;
ptr=ptr->link;
}
}
if(ptr->info == temp->info)
{
temp2->link=NULL;
free(ptr);
}
temp=temp->link;
} temp=first; while(temp != NULL)
{
printf("%4d --> %6d",temp->info,temp->link);
temp=temp->link;
}
}
```

OUTPUT:

5) Write a C program to implement Bubble Sort for singly linked list.

```
#include<stdio.h>

#include<conio.h>

struct node

{

int info;
struct node *link;

}*first=NULL, *new=NULL, *temp=NULL, *temp2=NULL,*ptr=NULL;

struct node * insert_first(int x);

struct node * insert_last(int x);

void display(struct node *);

void delete(int data);

void sort();

void main()

{

int n,x,a,b,value,ans=0;

while(1)

{

printf("\n(1) INSERTING NODE AT FIRST  \n(2) INSERTING NODE AT LAST
\n(3) DISPLAY ALL ELEMENTS (BEFORE SORTING): \n(4) DELETING A NODE
\n(5) DISPLAY ALL ELEMENTS (AFTER SORTING): ");

printf("\n\nENTER YOUR CHOICE:");

scanf("%d",&n);

switch(n)

{

case 1:
printf("Enter the element you want to insert at first:");
```

```
scanf("%d",&x); first=insert_first(x);
break;
case 2:
printf("Enter the element you want to insert at last:");
scanf("%d",&x); first=insert_last(x);
break;
case 3:
printf("Elements are as follows:\n");
display(first);
break;
case 4:
printf("\nEnter the element you want to delete:");
scanf("%d",&value);
delete(value);
break; case 5: sort();
break;
}
}
}
struct node * insert_first(int x)
{
new=(struct node *) malloc(sizeof(struct node));
new->info=x; new->link= first;
return new;
}
struct node * insert_last(int x)
```

```
{
new=(struct node *) malloc(sizeof(struct node));
new->info=x;
new->link= NULL;
if(first==NULL)
{
return new;
}
else
{
temp=first;
while(temp->link!= NULL)
{
temp=temp->link;
}
temp->link=new;
return first;
}
}
void display(struct node *temp)
{
temp=first;
if(first==NULL)
{
printf("LINK IS EMPTY.");
}
```

```
while(temp != NULL)
{
printf("%4d --> %6d",temp->info,temp->link);
temp=temp->link;
}
}
void delete(int data)
{
if(first==NULL)
{
printf("\nUNDERFLOWW CONDITION");
return;
}
temp=first;
while(temp->info!= data)
{
temp2=temp; temp=temp->link;
if(temp== NULL)
{
printf("Value %d is not present in the linked list",data);
return;
}
}
if(first->info==data)
{
```

```
temp=first; first=first->link; free
(temp); return;
}
else
{
temp2->link=temp->link;
free(temp);
}
}
void sort()
{
int count=0,swap;
if(first==NULL)
{
printf("Linked list is empty");
return;
} temp=first; while(temp != NULL)
{
temp2=first;
while(temp2->link !=NULL)
{
if(temp2->info > temp2->link->info)
{
swap=temp2->info;
temp2->info=temp2->link->info;
temp2->link->info=swap;
```

```
}
temp2=temp2->link;
}
temp=temp->link;
}
printf("AFTER SORTING, THE ELEMENTS IN LIST ARE:\n");
temp=first;
while(temp != NULL)
{
printf("%4d --> %6d",temp->info,temp->link);
temp=temp->link;
}
}
```

OUTPUT: