Shri Vile Parle Kelavani Mandal's
DWARKADAS J. SANGHVI COLLEGE OF ENGINEERING
(Autonomous College Affiliated to the University of Mumbai)
NAAC Accredited with "A" Grade (CGPA : 3.18)

Roll No.: C107 **Name: PARAM SHUKLA** **SAP ID: 60004220046**

DIV:C-2 **BATCH:2**

EXPERIMENT 1

Problem statement 1: Write a c program to implement stacks and queues using arrays

CODE :

```
//STACK

#include<stdio.h>
struct stack{
    int top;
    int a[5];
};

void display(struct stack s);
int stacktop(struct stack s);
void push(struct stack *ps,int x);
int pop(struct stack *ps);
int stacktop(struct stack s);
void main()
{
    struct stack s;
    int x;
    char c;
        s.top=-1;
    do
    {
    printf("\nmenu\n");
    printf("1:push\n");
    printf("2:pop\n");
    printf("3:stacktop\n");
    printf("4:exit\n");
    printf("enter a choice\n");
    scanf("%c",&c);

    switch(c)
    {
    case('1'):
    printf("enter x\n");
    scanf("%d",&x);
    push(&s,x);
    display(s);
    break;

    case('2'):

        x=pop(&s);
      printf("The element that is poped is %d\n",x);
        display(s);
        break;

    case('3'):

        x=stacktop(s);
      printf("The topmost element of stack is %d",x);
        break;

    case('4'):

        break;

}
}
while(c!='4');
}
void push(struct stack *ps,int x)
{
    ps->top++;
    ps->a[ps->top]=x;

}
int empty(struct stack *ps)
{
    if(ps->top==-1)
        return 1;
    else
        return 0;
}
int pop(struct stack *ps)
{
    if(empty(ps))
    {
        printf("cannot pop.Stack is empty");

    }
    else
    {
        return ps->a[ps->top];
        ps->top--;
    }

}
int stacktop(struct stack s)
{
    if(empty(&s))
    {
        printf("stack is empty");

    }
    else
    {
        return s.a[s.top];
    }
}
void display(struct stack s)
{
        int i;
        printf("stack from topmost element is:");
        for(i=s.top;i>=0;i--)
        {
        printf("%d",s.a[i]);
        }
  }


//QUEUE
```

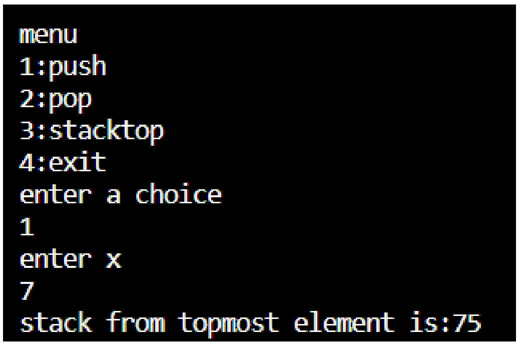
```
#include<stdio.h>
# define qs 10

struct queue{
   int  front,rear;
   int items[qs];
};

void enqueue(struct queue *pq,int x);
int empty (struct queue *pq);
int dequeue(struct queue *pq);
void show(struct queue pq);

void enqueue(struct queue *pq,int x)
{
   if(pq->rear==qs-1)
   {
      printf("Queue is full cannot insert.\n");
   }
   pq->rear++;
   pq->items[pq->rear]=x;
}
int empty(struct queue *pq)
{
   if(pq->rear<pq->front)
   {
      return 1;
   }
   else
   {
      return 0;
   }
}
int dequeue(struct queue *pq)
{
   if(empty(pq))
   {
      printf("Queue is empty.Cananot remove.\n");
   }
   return pq->items[pq->front];
   pq->front++;
}
void show(struct queue pq)
```

```
{
   int i;
   printf("Queue from frontmost element is:\n");
   for(i=pq.front;i<=pq.rear;i++)
   {
      printf("%d",pq.items[i]);
   }
}
void main()
{
   struct queue q;
   int x;
   char c;
   q.front=0;
   q.rear=-1;
   do
   {
     printf("\nMenu\n");
     printf("1.enqueue\n");
     printf("2.dequeue\n");
     printf("3.exit\n");
     printf("enter a choice\n");
     scanf("%c",&c);

     switch(c)
     {
       case('1'):
       printf("enter x\n");
       scanf("%d",&x);
       enqueue(&q,x);
       show(q);
       break;

       case('2'):
       x=dequeue(&q);
printf("Removed element from the queue is %d\n",x);
       show(q);
       break;

       case('3'):
       break;
     }
   }
```

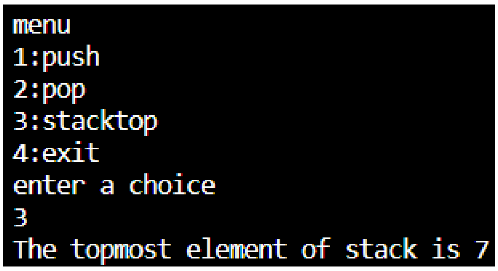
```
    while(c!=3);
}
```

OUTPUT:

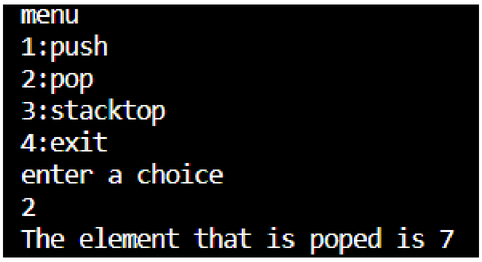
//STACK



```
menu
1:push
2:pop
3:stacktop
4:exit
enter a choice
1
enter x
7
stack from topmost element is:75
```

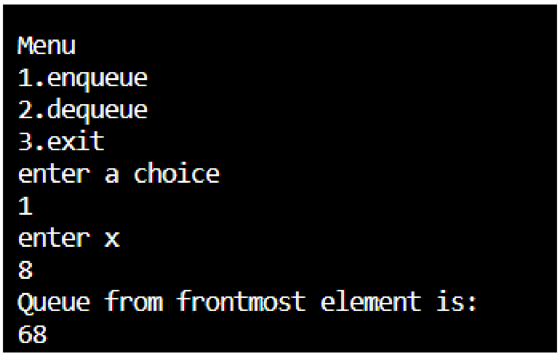


```
menu
1:push
2:pop
3:stacktop
4:exit
enter a choice
3
The topmost element of stack is 7
```

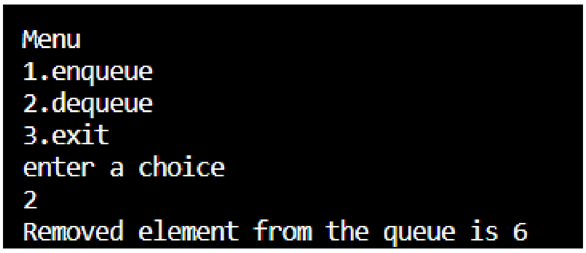


```
menu
1:push
2:pop
3:stacktop
4:exit
enter a choice
2
The element that is poped is 7
```

//QUEUE



```
Menu
1.enqueue
2.dequeue
3.exit
enter a choice
1
enter x
8
Queue from frontmost element is:
68
```



```
Menu
1.enqueue
2.dequeue
3.exit
enter a choice
2
Removed element from the queue is 6
```

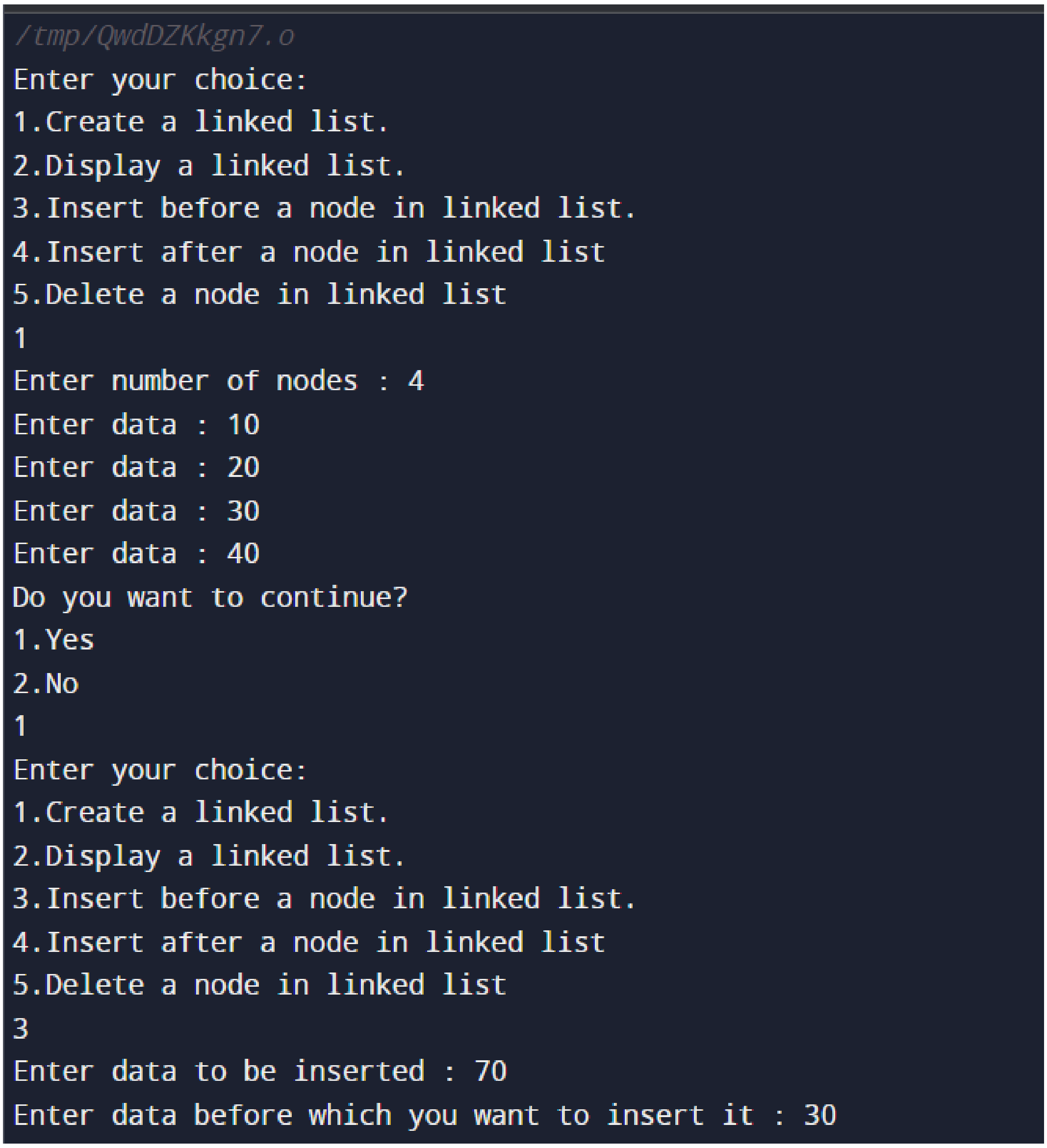
EXPERIMENT 2

Problem statement 2: Write a c program to implement different functions of linked list

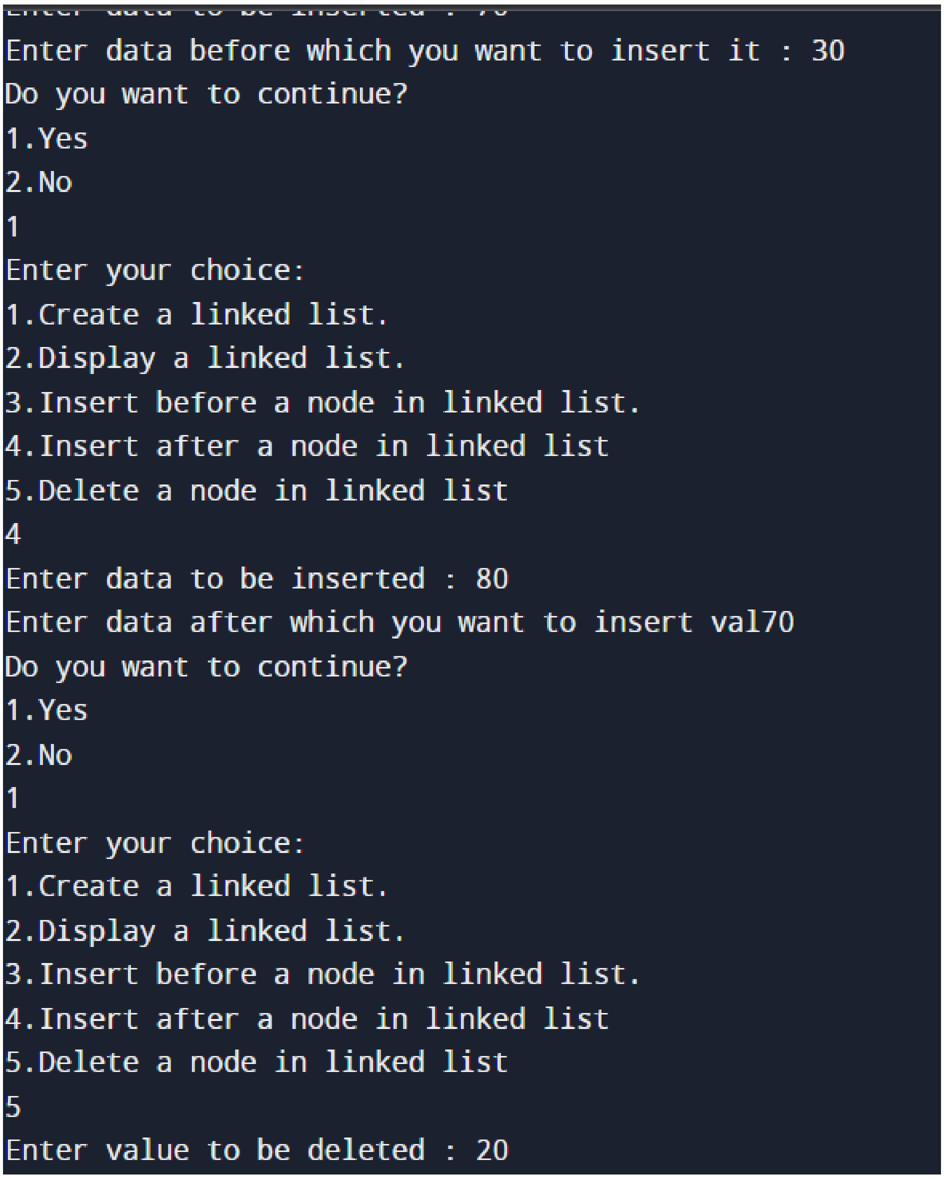
CODE :

```
// Single Linked list
#include<stdio.h>
#include<stdlib.h>

//declare a node
struct node{
    int data;
    struct node *next;
} *start,*temp;
// *start is referred as head in many books and temp is a temporary variable

//creating a linked list
void createLinkedList(){
    int i,n,x;
    printf("Enter number of nodes : ");
    scanf("%d",&n);
    for(i=0;i<n;i++){
        struct node *nn;
        nn=(struct node *)malloc(sizeof(struct node));
        printf("Enter data : ");
        scanf("%d",&x);
        nn->data=x;
        nn->next=NULL;
        if(start==NULL){
            start=nn;
        }
        //this if statement will only make 10|NULL ka node
        else{
            temp=start;
            while(temp->next!=NULL){
                temp=temp->next;
            }
            temp->next=nn;
        }
    }
}

void display(){
    temp=start;
    while(temp!=NULL){
        printf("%d %d\n",temp->data,temp->next);
        temp=temp->next;
    }
}

void insertAfter(){
    int val,x;
    printf("Enter data to be inserted : ");
    scanf("%d",&val);
    printf("Enter data after which you want to insert val");
    scanf("%d",&x);
    struct node *nn;
    nn=(struct node *)malloc(sizeof(struct node));
    nn->data=val;
    //lets say you have a premade linked list with some nodes so you have one start and last node with NULL value
    temp=start;
    while(temp->data!=x){
        temp=temp->next;
    }
    nn->next = temp->next;
    temp->next=nn;
}

void insertBefore(){
    int val,x;
    printf("Enter data to be inserted : ");
    scanf("%d",&val);
    printf("Enter data before which you want to insert it : ");
    scanf("%d",&x);
    struct node *nn,*temp2,*temp;
    nn=(struct node *)malloc(sizeof(struct node));
    nn->data=val;
    //lets say you have a premade linked list with some nodes so you have one start and last node with NULL value
    temp=start;
    // struct node *temp2 //another temporary variable
    while(temp->data!=x){
        temp2=temp;
        temp=temp->next;
    }
    // printf("temp2 is %d \n",temp2->data); //20
    // printf("temp is %d \n",temp->data);  //30
    if(temp==start){
        nn->next = temp;
        start = nn;
    }
    else{
        nn->next=temp;
        temp2->next=nn;
    }
}

void delete(){
    int val;
    struct node *temp,*temp2;
    printf("Enter value to be deleted : ");
    scanf("%d",&val);
    //lets say you have a premade linked list with some nodes so you have one
start and last node with NULL value
    temp=start;
    while(temp->data!=val && temp!=NULL){
        temp2=temp;
        temp=temp->next;
    }
    if(temp==start){
        start=temp->next;
        free(temp);
    }
    else if(temp!=NULL){
        temp2->next=temp->next;
        free(temp);
    }
    else {
        printf("Value not found");
    }
}

void main(){
    // createLinkedList();
    // display();
    // insertAfter();
    // display();
    // insertBefore();
    // display();
    // delete();
    // display();
    int choice,choice2;
    do
    {
        printf("Enter your choice:\n1.Create a linked list.\n2.Display a linked list.\n3.Insert before a node in linked list.\n4.Insert after a node in linked list\n5.Delete a node in linked list\n");
        scanf("%d",&choice);
        switch(choice)
        {
            case 1:createLinkedList();
            break;
            case 2:display();
            break;
            case 3: insertBefore();
            break;
            case 4:insertAfter();
            break;
            case 5:delete();
            break;
        }
        printf("Do you want to continue?\n1.Yes\n2.No\n");
        scanf("%d",&choice2);
    }while(choice2==1);


}
```

OUTPUT :



```
/tmp/QwdDZKkgn7.o
Enter your choice:
1.Create a linked list.
2.Display a linked list.
3.Insert before a node in linked list.
4.Insert after a node in linked list
5.Delete a node in linked list
1
Enter number of nodes : 4
Enter data : 10
Enter data : 20
Enter data : 30
Enter data : 40
Do you want to continue?
1.Yes
2.No
1
Enter your choice:
1.Create a linked list.
2.Display a linked list.
3.Insert before a node in linked list.
4.Insert after a node in linked list
5.Delete a node in linked list
3
Enter data to be inserted : 70
Enter data before which you want to insert it : 30
```



```
Enter data to be inserted : 70
Enter data before which you want to insert it : 30
Do you want to continue?
1.Yes
2.No
1
Enter your choice:
1.Create a linked list.
2.Display a linked list.
3.Insert before a node in linked list.
4.Insert after a node in linked list
5.Delete a node in linked list
4
Enter data to be inserted : 80
Enter data after which you want to insert val70
Do you want to continue?
1.Yes
2.No
1
Enter your choice:
1.Create a linked list.
2.Display a linked list.
3.Insert before a node in linked list.
4.Insert after a node in linked list
5.Delete a node in linked list
5
Enter value to be deleted : 20
```

Experiment 3

Problem statement 3: Write a c program to implement polynomial operations(addition, subtraction) using linked list

CODE :

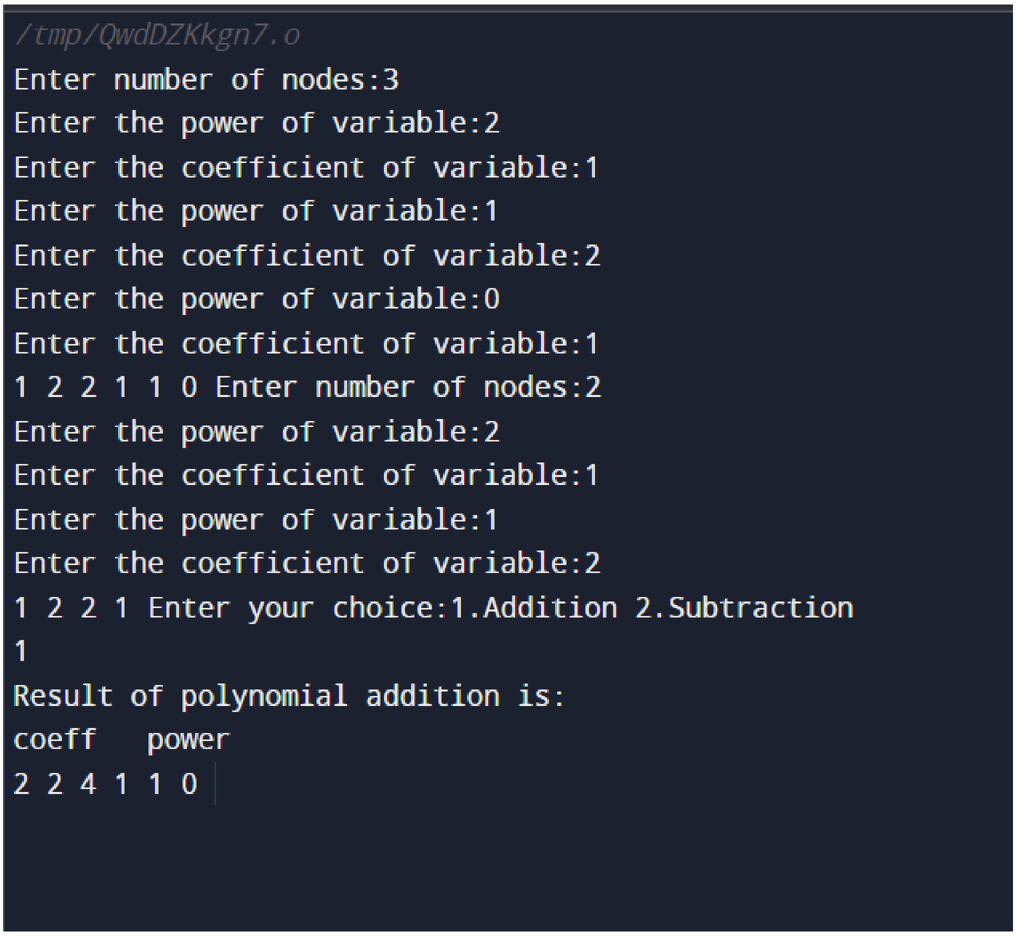
```
#include <stdio.h>
#include <stdlib.h>
struct node
{
    int coeff, power;
    struct node *next;
} * start1, *start2, *start3;
struct node * create_ll(struct node *start)
{
    int n, i, x, coeff, power;
    struct node *temp;
    printf("Enter number of nodes:");
    scanf("%d", &n);
    for (i = 0; i < n; i++)
    {
        struct node *nn;
        if (start == NULL)
        {
            nn = (struct node *)malloc(sizeof(struct node));
            printf("Enter the power of variable:");
            scanf("%d", &power);
            printf("Enter the coefficient of variable:");
            scanf("%d", &coeff);
            nn->power = power;
            nn->coeff = coeff;
            nn->next = NULL;
            start = nn;
        }
        else
        {
            nn = (struct node *)malloc(sizeof(struct node));
            printf("Enter the power of variable:");
            scanf("%d", &power);
            printf("Enter the coefficient of variable:");
            scanf("%d", &coeff);
            nn->power = power;
            nn->coeff = coeff;
            nn->next = NULL;
            temp = start;
            while (temp->next != NULL)
            {
                temp = temp->next;
            }
            temp->next = nn;
        }
    }
    return start;
}
```

```
void add_node(int c, int power)
{
    struct node *newnode, *temp;
    newnode = (struct node *)malloc(sizeof(struct node));
    newnode->coeff = c;
    newnode->power = power;
    newnode->next=NULL;
    if (start3 == NULL)
    {
        start3 = newnode;
    }
    else
    {
        temp = start3;
        while(temp->next!= NULL)
        {
            temp = temp->next;
        }
        temp->next = newnode;
    }
}

void polynomial_add()
{
    struct node *temp1, *temp2;
    int c;
    temp1 = start1;
```

```
temp2 = start2;
while (temp1 != NULL && temp2 != NULL)
{
    if (temp1->power == temp2->power)
    {
        c = temp1->coeff + temp2->coeff;
        add_node(c, temp1->power);
        temp1 = temp1->next;
        temp2 = temp2->next;
    }
    else if (temp1->power > temp2->power)
    {
        add_node(temp1->coeff, temp1->power);
        temp1 = temp1->next;
    }
    else
    {
        add_node(temp2->coeff, temp2->power);
        temp2 = temp2->next;
    }
}
if (temp2 == NULL)
{
    while (temp1 != NULL)
    {
        add_node(temp1->coeff, temp1->power);
        temp1 = temp1->next;
        }
    }
    else if (temp1 == NULL)
    {
        while (temp2 != NULL)
        {
            add_node(temp2->coeff, temp2->power);
            temp2 = temp2->next;
        }
    }
}
void polynomial_subt()
{
    struct node *temp1, *temp2;
    int c;
    temp1 = start1;
    temp2 = start2;
    while (temp1 != NULL && temp2 != NULL)
    {
        if (temp1->power == temp2->power)
        {
            c = temp1->coeff - temp2->coeff;
            add_node(c, temp1->power);
            temp1 = temp1->next;
            temp2 = temp2->next;
        }
        else if (temp1->power > temp2->power)
        {
            add_node(temp1->coeff, temp1->power);
            temp1 = temp1->next;
        }
        else
        {
            add_node(-temp2->coeff, temp2->power);
            temp2 = temp2->next;
        }
    }
    if (temp2 == NULL)
    {
        while (temp1 != NULL)
        {
            add_node(temp1->coeff, temp1->power);
            temp1 = temp1->next;
        }
    }
    else if (temp1 == NULL)
    {
        while (temp2 != NULL)
        {
            add_node(-temp2->coeff, temp2->power);
            temp2 = temp2->next;
        }
    }
}
```

```
void display(struct node *start)
{
    struct node *temp;
    temp = start;
    while (temp != NULL)
    {
        printf("%d %d\t", temp->coeff, temp->power);
        temp = temp->next;
    }
}

void main()
{
    int choice;
    start1 = create_ll(start1);
    display(start1);
    start2 = create_ll(start2);
    display(start2);
    printf("Enter your choice:1.Addition 2.Subtraction\n");
    scanf("%d",&choice);
    switch(choice)
    {
        case 1:polynomial_add();
        printf("Result of polynomial addition is:\ncoeff\tpower\n");
        display(start3);
        break;
        case 2:polynomial_subt();
        printf("Result of polynomial subtraction is:\ncoeff\tpower\n");
        display(start3);
        break;
    }
}
```

OUTPUT :



```
/tmp/QwdDZKkgn7.o
Enter number of nodes:3
Enter the power of variable:2
Enter the coefficient of variable:1
Enter the power of variable:1
Enter the coefficient of variable:2
Enter the power of variable:0
Enter the coefficient of variable:1
1 2 2 1 1 0 Enter number of nodes:2
Enter the power of variable:2
Enter the coefficient of variable:1
Enter the power of variable:1
Enter the coefficient of variable:2
1 2 2 1 Enter your choice:1.Addition 2.Subtraction
1
Result of polynomial addition is:
coeff   power
2 2 4 1 1 0
```

Experiment 4

Problem statement 4: Write a c program to implement stacks and queue using linked list

CODE :

//stacks using LL

```
#include<stdio.h>

#include<stdlib.h>

struct node{

    int data;

    struct node *next;

}*temp,*top;


void push(){

   // int top=-1;

    int val;

    struct node *nn,*temp;

    printf("Enter a value : ");

    scanf("%d",&val);

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

    nn->data=val;

    nn->next=NULL;

    if(top==NULL){

        top=nn;

    }

    else{

        nn->next=top;

        top=nn;
    }


}
void Display()
{
    printf("%d is top \n",top->data);
    printf("The stack  is : \n");
    temp = top;
    while (temp != NULL)
    {
        printf("%d %d\n", temp->data,temp->next);
        temp = temp->next;
    }
}
void pop(){
    int val;
    struct node *temp;
    if(top==NULL){
        printf("Stack is empty...\n");
    }
    else{
       val=top->data;
       temp=top;
       top=top->next;
       free(temp);
    }
}
```

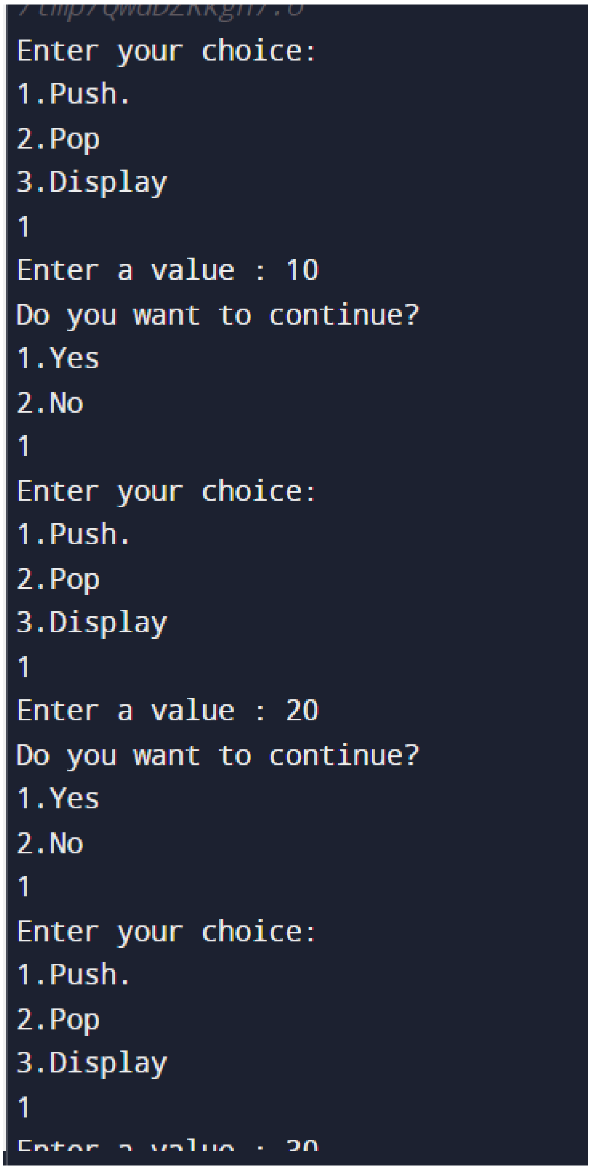
```
void main(){



    int choice,choice2;
    do
    {
        printf("Enter your choice:\n1.Push.\n2.Pop\n3.Display \n");
        scanf("%d",&choice);
        switch(choice)
        {
            case 1:push();
            break;
            case 2:pop();
            break;
            case 3: Display();
            break;

        }
        printf("Do you want to continue?\n1.Yes\n2.No\n");
        scanf("%d",&choice2);
    }while(choice2==1);
}
```

//queue using Linked list

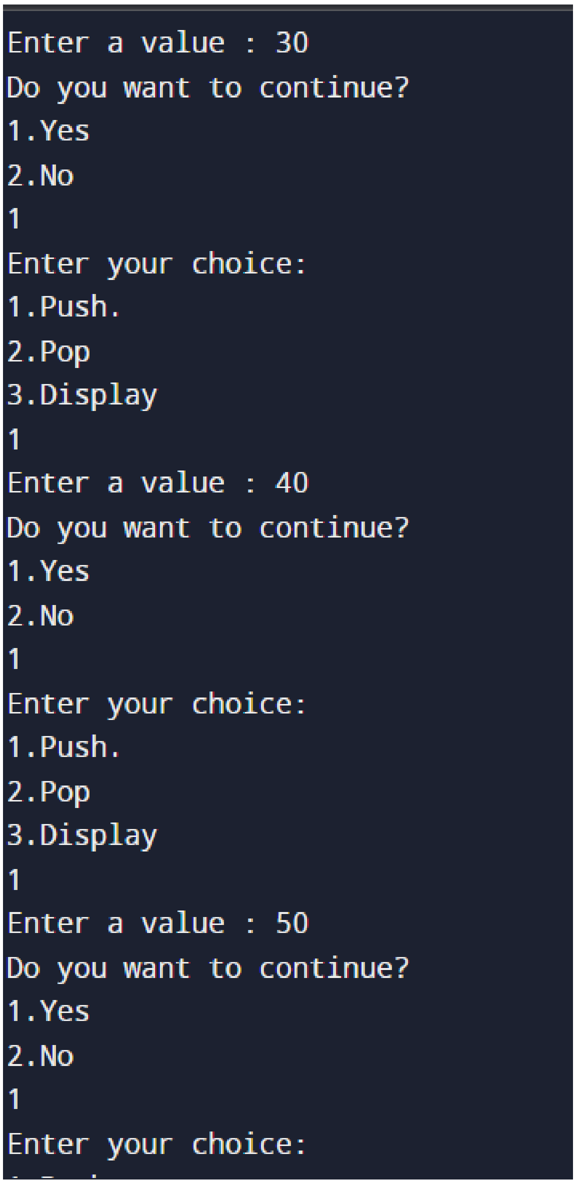
```
#include<stdio.h>
#include<stdlib.h>
struct node{
    int data;
    struct node *next;
}*temp,*front,*rear;
void Enqueue(){
    int val;
    struct node *nn;
    nn=(struct node *)malloc(sizeof(struct node));
    if(rear==NULL){
        printf("Enter a element(1st) : ");
        scanf("%d",&val);
        nn->data=val;
        front=nn;
        rear=nn;
    }
    else{
        printf("Enter a value : ");
        scanf("%d",&val);
        nn->data=val;
        rear->next=nn;
        rear=nn;
    }
}
void Dequeue(){
    int val;
    if(front==NULL){
        printf("Queue is empty \n");
    }
    else if(front==rear){
        val=front->data;
        printf("The deleted element is %d \n",val);
        temp=front;
        front=NULL;
        rear=NULL;
        free(temp);
    }
    else {
        val=front->data;
        printf("The deleted element (else) is %d \n",val);
        temp=front;
        front=front->next;
        free(temp);
    }
}
void Display()
{
    temp = front;
    while (temp != NULL)
    {
        printf("%d %d\n", temp->data,temp->next);
        temp = temp->next;
    }
}
void main(){
    int choice;
    do
    {
        printf("Choices are:\n1.Enqueue\n2.Dequeue\n3.Display\n4.Exit\nEnter your choice:");
        scanf("%d",&choice);
        switch(choice)
        {
            case 1:
            Enqueue();
            break;
            case 2:
            Dequeue();
            break;
            case 3:
            Display();
            break;
            default:
            printf("Invalid choice\n");

        }
    } while (choice!=4);
}
```

OUTPUT:

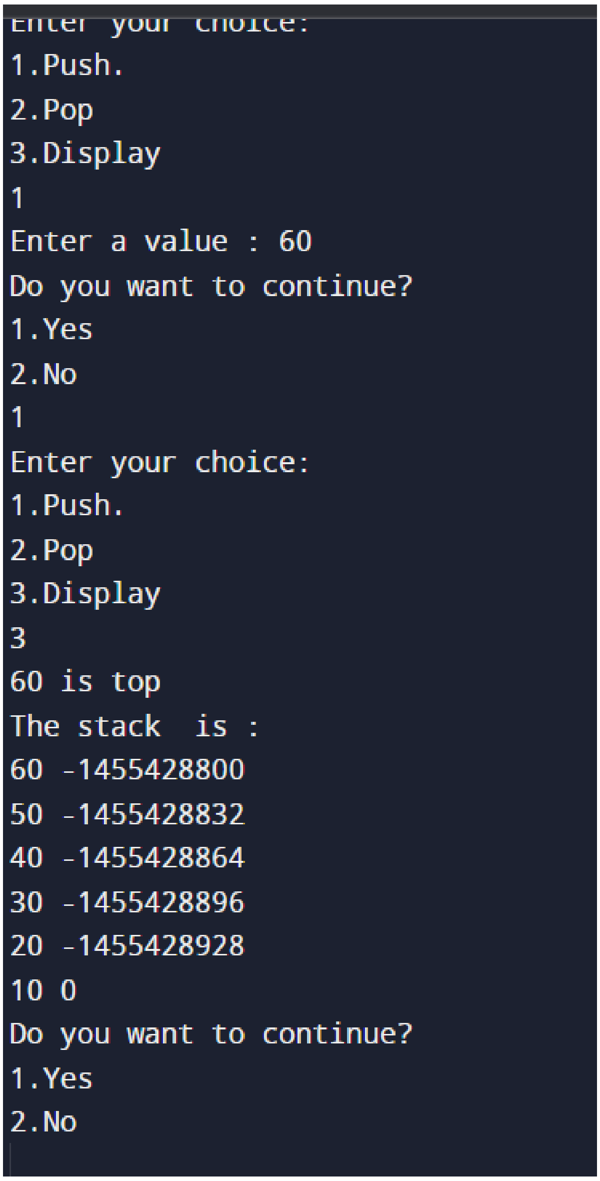
//STACKS:



```
/tmp/QwdDZkkgn7.o
Enter your choice:
1.Push.
2.Pop
3.Display
1
Enter a value : 10
Do you want to continue?
1.Yes
2.No
1
Enter your choice:
1.Push.
2.Pop
3.Display
1
Enter a value : 20
Do you want to continue?
1.Yes
2.No
1
Enter your choice:
1.Push.
2.Pop
3.Display
1
Enter a value : 30
```

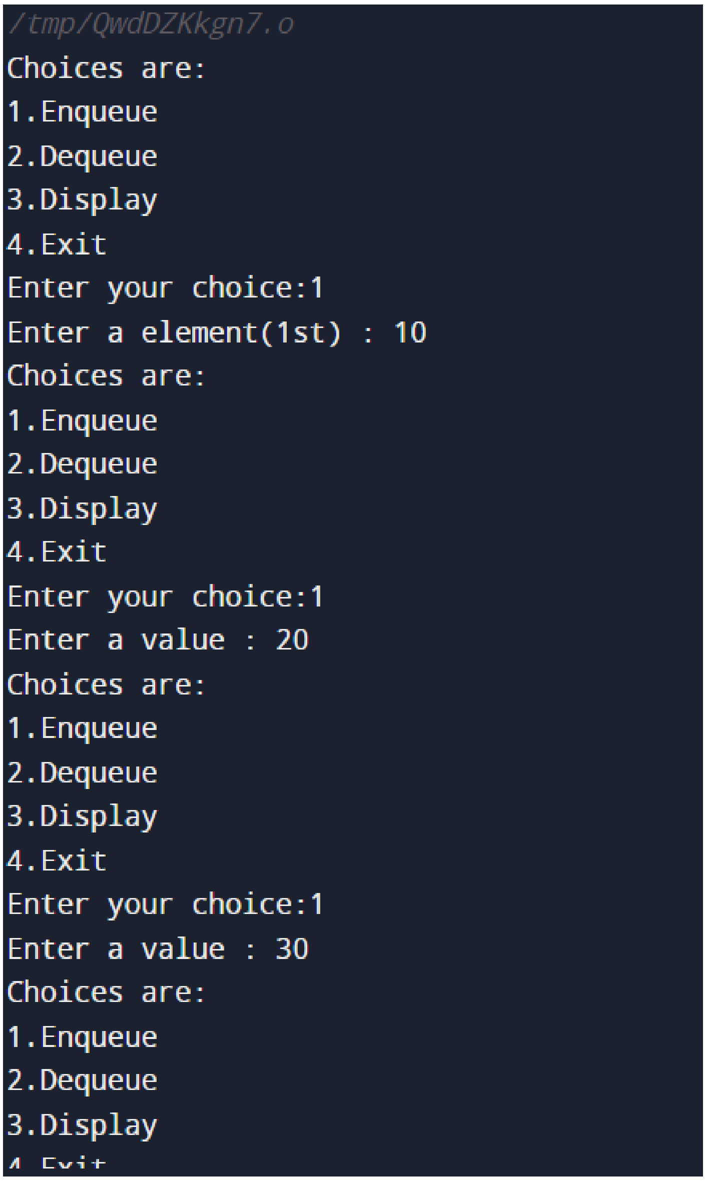


```
Enter a value : 30
Do you want to continue?
1.Yes
2.No
1
Enter your choice:
1.Push.
2.Pop
3.Display
1
Enter a value : 40
Do you want to continue?
1.Yes
2.No
1
Enter your choice:
1.Push.
2.Pop
3.Display
1
Enter a value : 50
Do you want to continue?
1.Yes
2.No
1
Enter your choice:
```

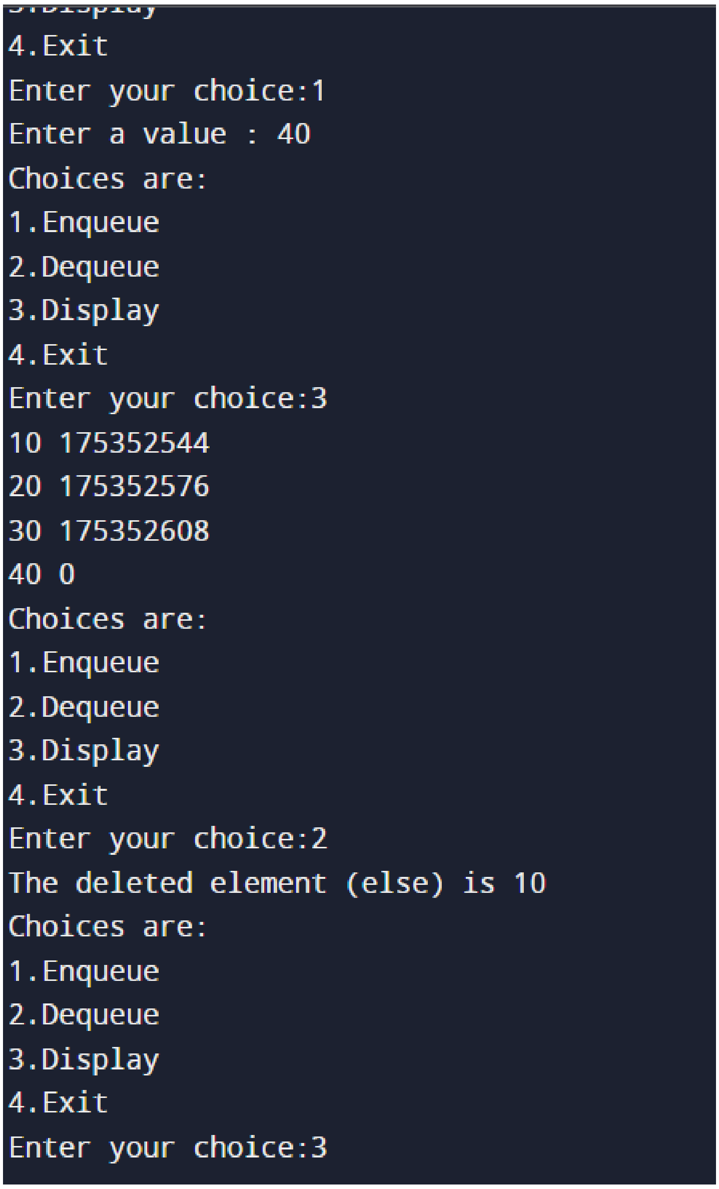


```
Enter your choice:
1.Push.
2.Pop
3.Display
1
Enter a value : 60
Do you want to continue?
1.Yes
2.No
1
Enter your choice:
1.Push.
2.Pop
3.Display
3
60 is top
The stack  is :
60 -1455428800
50 -1455428832
40 -1455428864
30 -1455428896
20 -1455428928
10 0
Do you want to continue?
1.Yes
2.No
```

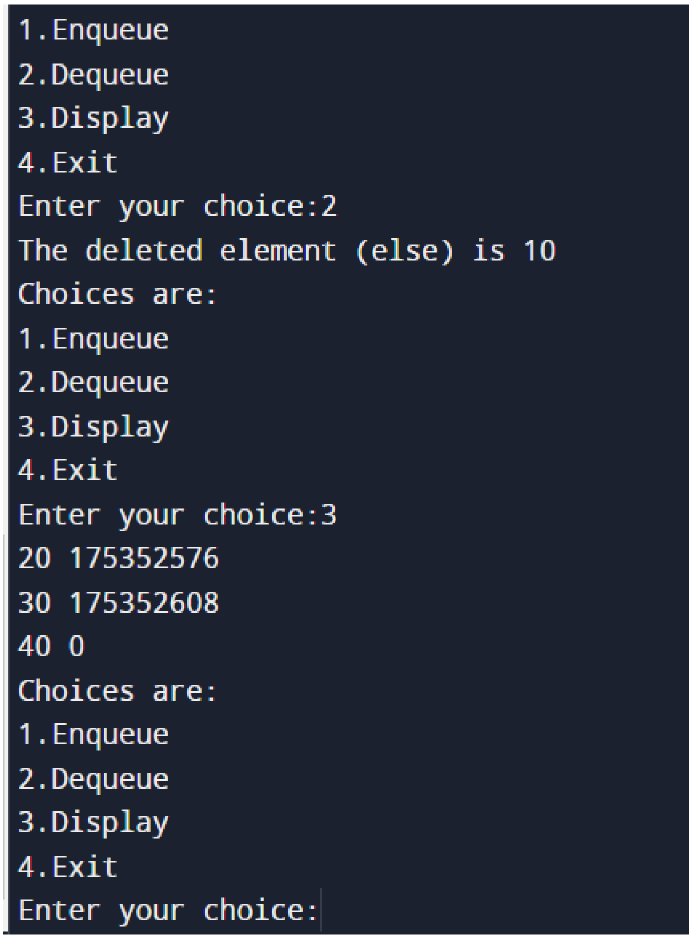
//QUEUE



```
/tmp/QwdDZKkgn7.o
Choices are:
1.Enqueue
2.Dequeue
3.Display
4.Exit
Enter your choice:1
Enter a element(1st) : 10
Choices are:
1.Enqueue
2.Dequeue
3.Display
4.Exit
Enter your choice:1
Enter a value : 20
Choices are:
1.Enqueue
2.Dequeue
3.Display
4.Exit
Enter your choice:1
Enter a value : 30
Choices are:
1.Enqueue
2.Dequeue
3.Display
4.Exit
```



```
3.Display
4.Exit
Enter your choice:1
Enter a value : 40
Choices are:
1.Enqueue
2.Dequeue
3.Display
4.Exit
Enter your choice:3
10 175352544
20 175352576
30 175352608
40 0
Choices are:
1.Enqueue
2.Dequeue
3.Display
4.Exit
Enter your choice:2
The deleted element (else) is 10
Choices are:
1.Enqueue
2.Dequeue
3.Display
4.Exit
Enter your choice:3
```



```
1.Enqueue
2.Dequeue
3.Display
4.Exit
Enter your choice:2
The deleted element (else) is 10
Choices are:
1.Enqueue
2.Dequeue
3.Display
4.Exit
Enter your choice:3
20 175352576
30 175352608
40 0
Choices are:
1.Enqueue
2.Dequeue
3.Display
4.Exit
Enter your choice:
```

Experiment 5

Problem statement 5: Write a c program to transform a infix expression to a postfix expression

CODE :

```
#include <stdio.h>
#include <ctype.h>
int top1 = -1; int
top2 = -1; int
st[100]; int
arr[100]; void
push2(char z){
top2++; st[top2] =
z;
}
char pop(){ char r =
st[top2]; st[top2]
= 0; top2--;
return r;
}
int priority(char a){
switch (a){ case
'+': case '-':
return 1;
break;
case '*':
case '/':
case '%':
```

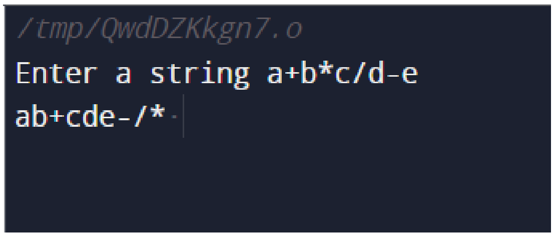
```
return 2;
break;
case '^':
return 3;
break;
}
return 0;
}
void infix_post(char s[]){
for (int i = 0; s[i] != '\0'; i++){
if (isalpha(s[i]) || isdigit(s[i])){
top1++;
arr[top1] = s[i];
}
else{
if (s[i] == '('){ push2(s[i]);
}
else if (s[i] == ')'){
while (st[top2] != '('){
char r = pop();
top1++; arr[top1]
= r;
}
}
else if (priority(s[i]) >= priority(st[top2]) && top2 > -1 && st[top2] != '('){
char r = pop(); top1++;
arr[top1] = r; st[top2]
= s[i];
}
else if (top2 == -1){ push2(s[i]);
}
else{ push2(s[i]);
}
}
}
while (top2 >= -1){
if (st[top2] == '('){
top2--; continue;
}
else{
top1++; arr[top1] =
st[top2]; top2--;
}
}
}
void display(){ for (int i = 0;
i < 10; i++){
printf("%c", arr[i]);
}
}
int main(){
char s[100];
printf("Enter a string ");
scanf("%s", s);
```

infix_post(s); display();

return 0;

}

OUTPUT:



```
/tmp/QwdDZKkgn7.o
Enter a string a+b*c/d-e
ab+cde-/*
```

Experiment 6

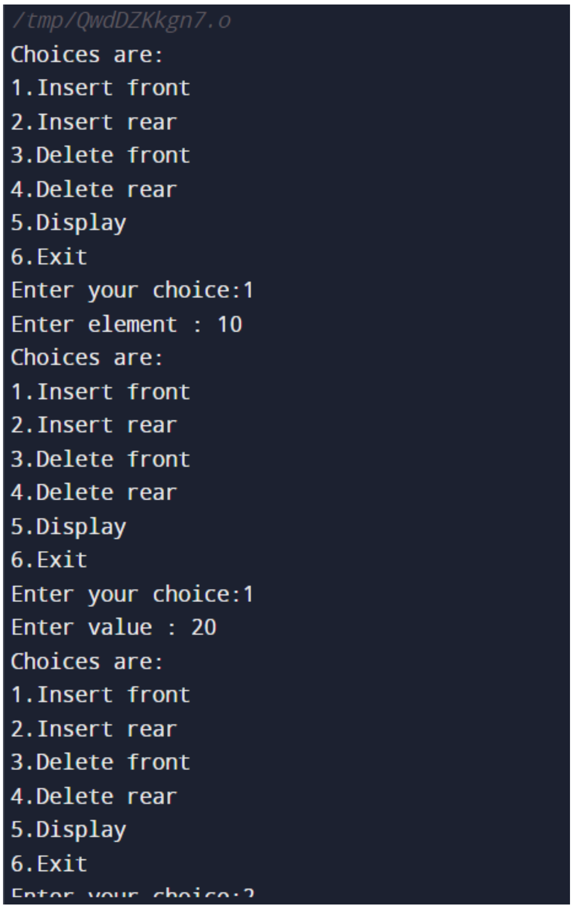
Problem statement 6: Write a c program to implement a double ended queue generalized version of the queue.

CODE :

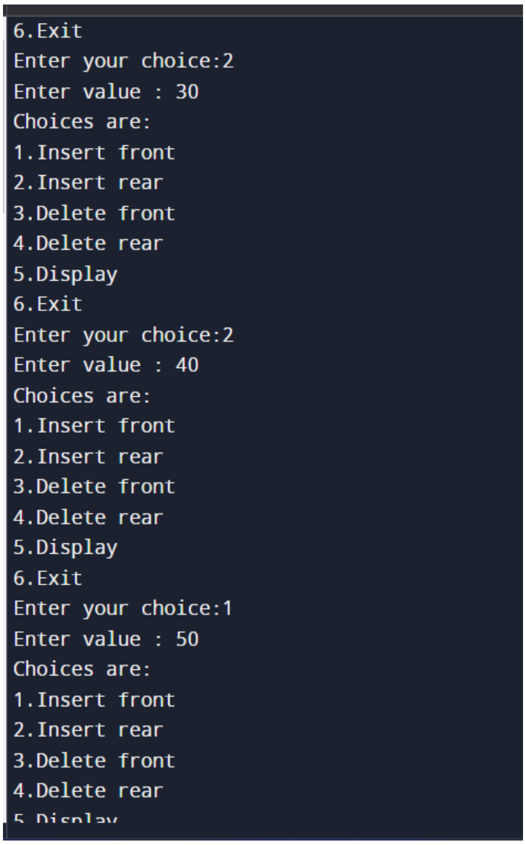
```
//Double Ended Queue
#include<stdio.h>
int front=-1,rear=-1,val,temp,size=5;
int queue[5];
void insertFront(){
    if((rear+1)%size==front){
        printf("Queue is full....\n");
    }
    else if(front==-1){
        printf("Enter element : ");
        scanf("%d",&val);
        front=0;
        rear=0;
        queue[front]=val;
    }
    else{
        printf("Enter value : ");
        scanf("%d",&val);
        front=(front-1+size)%size;
        queue[front]=val;
    }
}
```

```
void Display(){
    int i;
    i=front;
    printf("The Elements are \n");
    while(i!=rear){
        printf("%d ",queue[i]);
        i=(i+1)%size;
    }
    printf("%d \n",queue[rear]);
   // printf("Display front se rear tak gaya \n");
    printf("%d is front and %d is rear\n",front,rear);
    printf("the front is %d and rear is %d\n",queue[front],queue[rear]);
}
void insertRear(){
    if((rear+1)%size==front){
        printf("This queue is full\n");
    }
    else if(rear==-1){
        printf("Enter value to be inserted (rearins) : ");
        scanf("%d",&val);
        front=0;
        rear=0;
        queue[rear]=val;
    }
    else{
        printf("Enter value : ");
        scanf("%d",&val);
        rear=(rear+1)%size;
        queue[rear]=val;
    }
}
void deleteFront(){
    if(front==-1){
        printf("Queue is empty..\n");
    }
    else if(front==rear){
        val=queue[front];
        printf("The deleted element is %d \n",val);
        front=-1;
        rear=-1;
    }
    else{
        val=queue[front];
        printf("The deleted element is %d \n",val);
        front=(front+1)%size;
    }
}
void deleteRear(){
    if(rear==-1){
        printf("Queue is empty..\n");
    }
    else if(front==rear){
        val=queue[front];
        printf("The deleted element is %d \n",val);
        front=-1;
        rear=-1;
    }
    else{
        val=queue[rear];
        printf("The deleted element is %d \n",val);
        rear=(rear-1+size)%size;
    }
}
void main(){
    int choice;
    do
    {
        printf("Choices are:\n1.Insert front\n2.Insert rear\n3.Delete front\n4.Delete
rear\n5.Display\n6.Exit\nEnter your choice:");
        scanf("%d",&choice);
        switch(choice)
        {
            case 1:
            insertFront();
            break;
             case 2:
             insertRear();
             break;
             case 3:
             deleteFront();
             break;
             case 4:
             deleteRear();
             break;
            case 5:
            Display();
            break;
            case 6:
            break;
            default:
            printf("Invalid choice\n");
        }
    } while (choice!=6);
}
```

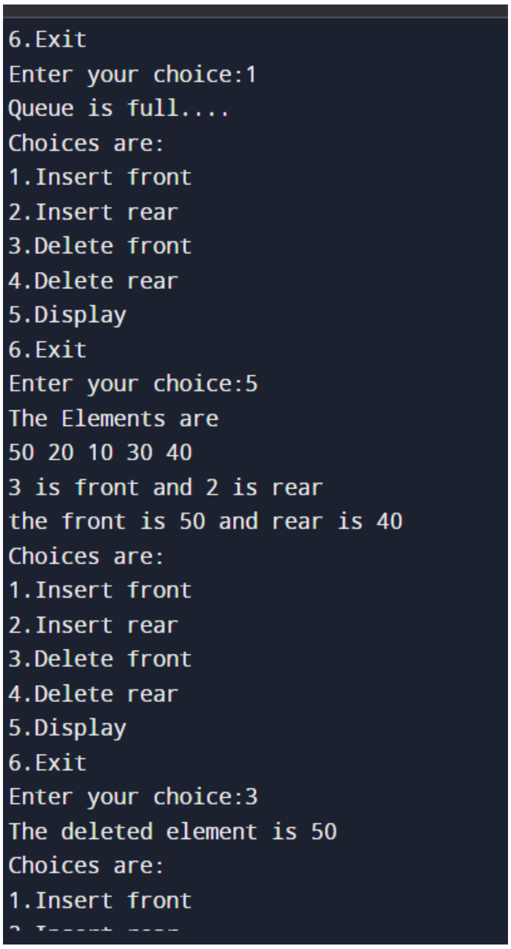
OUTPUT :



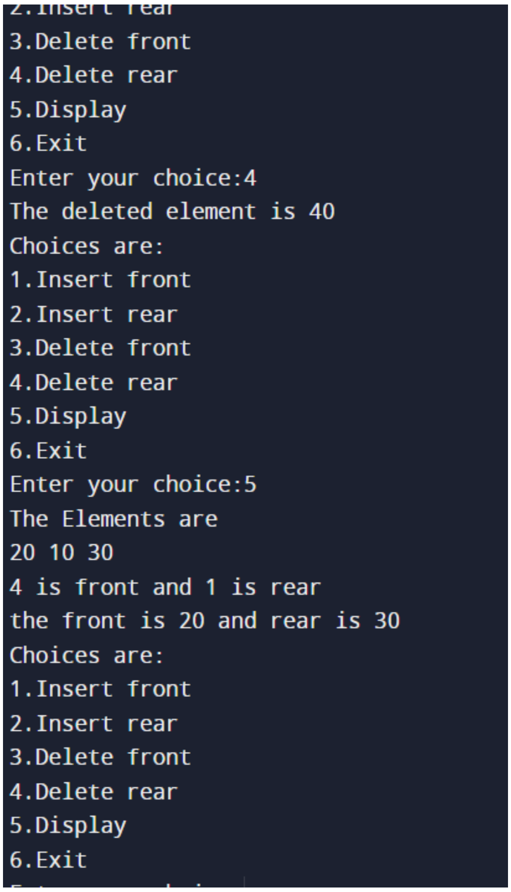
```
/tmp/QwdDZKkgn7.o
Choices are:
1.Insert front
2.Insert rear
3.Delete front
4.Delete rear
5.Display
6.Exit
Enter your choice:1
Enter element : 10
Choices are:
1.Insert front
2.Insert rear
3.Delete front
4.Delete rear
5.Display
6.Exit
Enter your choice:1
Enter value : 20
Choices are:
1.Insert front
2.Insert rear
3.Delete front
4.Delete rear
5.Display
6.Exit
Enter your choice:2
```



```
6.Exit
Enter your choice:2
Enter value : 30
Choices are:
1.Insert front
2.Insert rear
3.Delete front
4.Delete rear
5.Display
6.Exit
Enter your choice:2
Enter value : 40
Choices are:
1.Insert front
2.Insert rear
3.Delete front
4.Delete rear
5.Display
6.Exit
Enter your choice:1
Enter value : 50
Choices are:
1.Insert front
2.Insert rear
3.Delete front
4.Delete rear
5.Display
```



```
6.Exit
Enter your choice:1
Queue is full....
Choices are:
1.Insert front
2.Insert rear
3.Delete front
4.Delete rear
5.Display
6.Exit
Enter your choice:5
The Elements are
50 20 10 30 40
3 is front and 2 is rear
the front is 50 and rear is 40
Choices are:
1.Insert front
2.Insert rear
3.Delete front
4.Delete rear
5.Display
6.Exit
Enter your choice:3
The deleted element is 50
Choices are:
1.Insert front
2.Insert rear
```



```
2.Insert rear
3.Delete front
4.Delete rear
5.Display
6.Exit
Enter your choice:4
The deleted element is 40
Choices are:
1.Insert front
2.Insert rear
3.Delete front
4.Delete rear
5.Display
6.Exit
Enter your choice:5
The Elements are
20 10 30
4 is front and 1 is rear
the front is 20 and rear is 30
Choices are:
1.Insert front
2.Insert rear
3.Delete front
4.Delete rear
5.Display
6.Exit
```

Experiment 7

Problem statement 7: Write a c program to implement a binary search tree

CODE :

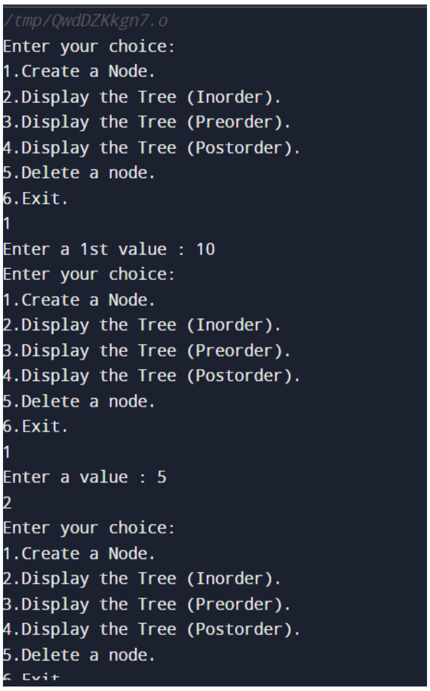
```
//Binary tree
#include<stdio.h>
#include<stdlib.h>
struct node{
   int data;
   struct node *left,*right;
}*root;
void insert(){
   int val;
   struct node *nn,*temp,*par;
   if(root==NULL)
   //which is true for 1st value input
   {
      nn=(struct node *)malloc(sizeof(struct node));
      printf("Enter a 1st value : ");
      scanf("%d",&val);
      nn->data=val;
      nn->left=NULL;
      nn->right=NULL;
      root=nn;
   }
   else{
    //now we to compare and traverse the tree to given other inputs ie to left or right
    temp=root;
    nn=(struct node *)malloc(sizeof(struct node));
    printf("Enter a value : ");
    scanf("%d",&val);
    nn->data=val;
    nn->left=NULL;
    nn->right=NULL;
    //uptil here we created a node
    //now to compare we need to find the parent node
    while(temp!=NULL){
        par=temp;
        if(temp->data<=val){
            temp=temp->right;
        }
        else{
            temp=temp->left;
        }
    }
    if(par->data<=val){
        par->right=nn;
        printf("1\n");
    }
    else{
        par->left=nn;
        printf("2\n");
    }
    }
}
void inorderdis(struct node *temp){
    //printf("The tree is : \n");
    if(temp!=NULL){
    inorderdis(temp->left);
    printf("%d\n",temp->data);
    inorderdis(temp->right);
   }
}
void Deletefun(){
    //lets first traverse at that value in a tree which we want to delete
    int val;
    printf("Enter a value that you want to delete : ");
    scanf("%d",&val);
    struct node *temp,*par,*succ,*ps;
    temp=root;
     while(temp->data!=val &&temp!=NULL){
        par=temp;
        if(temp->data<=val){
            temp=temp->right;
        }
        else{
            temp=temp->left;
        }
    }
    printf("%d is temp now and %d is par\n",temp->data,par->data);
```

```
//what if value entered is not in tree
if(temp==NULL){
    printf("Value not found..\n");
}
//leaf node case for 20 ka value
else if(temp->right==NULL && temp->left==NULL){
    if(par->right==temp){
        par->right=NULL;
        free(temp);
    }
    else{
        par->left=NULL;
        free(temp);
    }
}
else if(temp->left!=NULL&&temp->right==NULL){
    if(temp==par->left){
        par->left=temp->left;
        free(temp);
    }
    else{
        par->right=temp->left;
        free(temp);
    }
}
//printf("%d is par now \n",par->data);
else if(temp->right!=NULL&&temp->left==NULL){
    if(temp==par->right){
        par->right=temp->right;
        free(temp);
    }
    else{
        par->left=temp->right;
        free(temp);
    }
}
//printf("%d is par now \n",par->data);
//case for 2 child node deletion
else{
    succ=temp->left;
    ps=temp;
    while(succ->right!=NULL){
        ps=succ;
        succ=succ->right;
    }
    printf("%d is succ and %d is ps",succ->data,ps->data);

    temp->data=succ->data;
    ps->right=succ->left;
   // ps=succ->right;
    free(succ);
}
printf("%d is ps\n",ps->data);
```

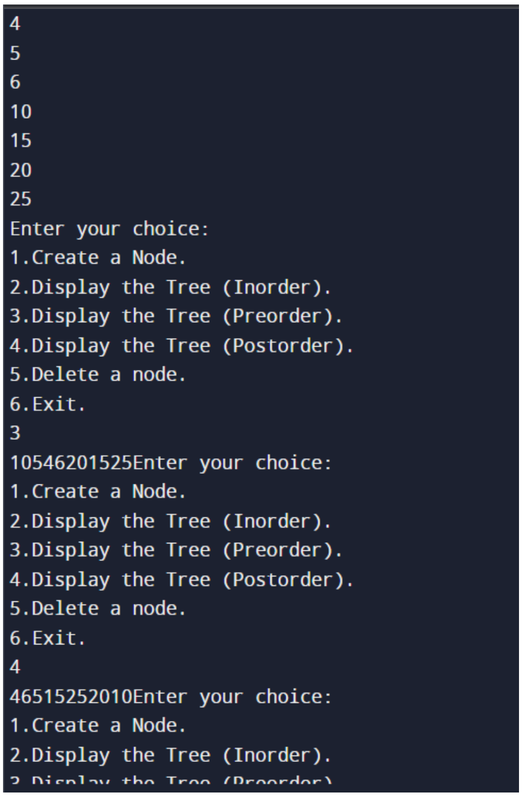
```
}
void preorder_display(struct node *temp)
{
    if(temp!=NULL)
    {
        printf("%d",temp->data);
        preorder_display(temp->left);
        preorder_display(temp->right);
    }
}
void postorder_display(struct node *temp)
{
    if(temp!=NULL)
    {
        postorder_display(temp->left);
        postorder_display(temp->right);
        printf("%d",temp->data);
    }
}
void main(){

    int choice,choice2;
    do
    {
        printf("Enter your choice:\n1.Create a Node.\n2.Display the Tree
(Inorder).\n3.Display the Tree (Preorder).\n4.Display the Tree
(Postorder).\n5.Delete a node.\n6.Exit.\n");
```

```
        scanf("%d",&choice);
        switch(choice)
        {
            case 1:insert();
            break;
            case 2:inorderdis(root);
            break;
            case 3:preorder_display(root);
            break;
            case 4:postorder_display(root);
            break;
            case 5:Deletefun();
            break;
        }
    }while(choice!=6);
}
```

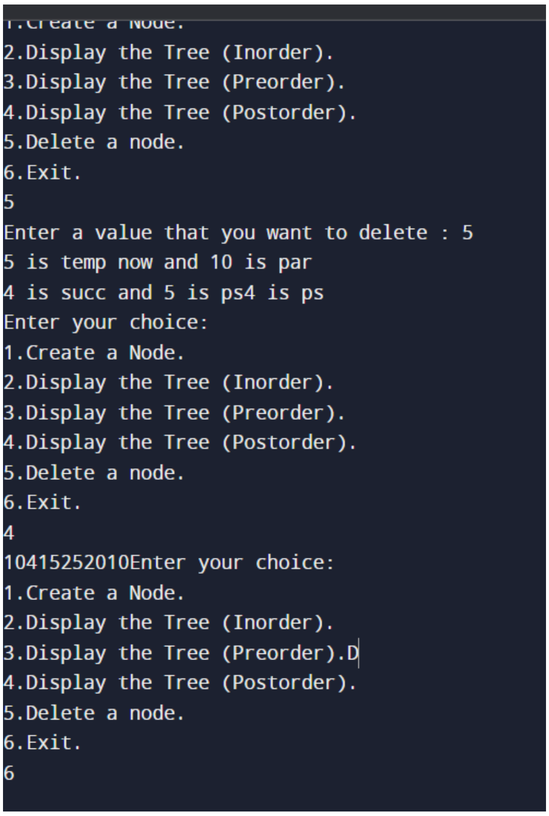
OUTPUT:



```
/tmp/QwdDZKkgn7.o
Enter your choice:
1.Create a Node.
2.Display the Tree (Inorder).
3.Display the Tree (Preorder).
4.Display the Tree (Postorder).
5.Delete a node.
6.Exit.
1
Enter a 1st value : 10
Enter your choice:
1.Create a Node.
2.Display the Tree (Inorder).
3.Display the Tree (Preorder).
4.Display the Tree (Postorder).
5.Delete a node.
6.Exit.
1
Enter a value : 5
2
Enter your choice:
1.Create a Node.
2.Display the Tree (Inorder).
3.Display the Tree (Preorder).
4.Display the Tree (Postorder).
5.Delete a node.
6.Exit.
```



```
4
5
6
10
15
20
25
Enter your choice:
1.Create a Node.
2.Display the Tree (Inorder).
3.Display the Tree (Preorder).
4.Display the Tree (Postorder).
5.Delete a node.
6.Exit.
3
10546201525Enter your choice:
1.Create a Node.
2.Display the Tree (Inorder).
3.Display the Tree (Preorder).
4.Display the Tree (Postorder).
5.Delete a node.
6.Exit.
4
46515252010Enter your choice:
1.Create a Node.
2.Display the Tree (Inorder).
3.Display the Tree (Preorder).
```



```
1.Create a Node.
2.Display the Tree (Inorder).
3.Display the Tree (Preorder).
4.Display the Tree (Postorder).
5.Delete a node.
6.Exit.
5
Enter a value that you want to delete : 5
5 is temp now and 10 is par
4 is succ and 5 is ps4 is ps
Enter your choice:
1.Create a Node.
2.Display the Tree (Inorder).
3.Display the Tree (Preorder).
4.Display the Tree (Postorder).
5.Delete a node.
6.Exit.
4
10415252010Enter your choice:
1.Create a Node.
2.Display the Tree (Inorder).
3.Display the Tree (Preorder).D
4.Display the Tree (Postorder).
5.Delete a node.
6.Exit.
6
```

EXPERIMENT 8

Problem statement 8: Write a c program to implement graph traversal(BFS&DFS)

CODE :

```
#include <stdio.h>
int q[20], top = -1, front = -1, rear = -1, a[20][20], vis[20], stack[20]; int
delete ();
void add(int item);
void bfs(int s, int n);
void dfs(int s, int n);
void push(int item);
int pop();
void main(){
int n, i, s, ch, j;
char c, dummy;
printf("Enter the number of vertices: "); scanf("%d",
&n);
for (i = 1; i <= n; i++){
for (j = 1; j <= n; j++){ printf("Enter 1 if %d has a node with
%d else 0: ", i, j);
scanf("%d", &a[i][j]);
}
}
printf("The Adjacency Matrix is: \n");
for (i = 1; i <= n; i++){ for
(j = 1; j <= n; j++){
printf(" %d", a[i][j]);
}
```

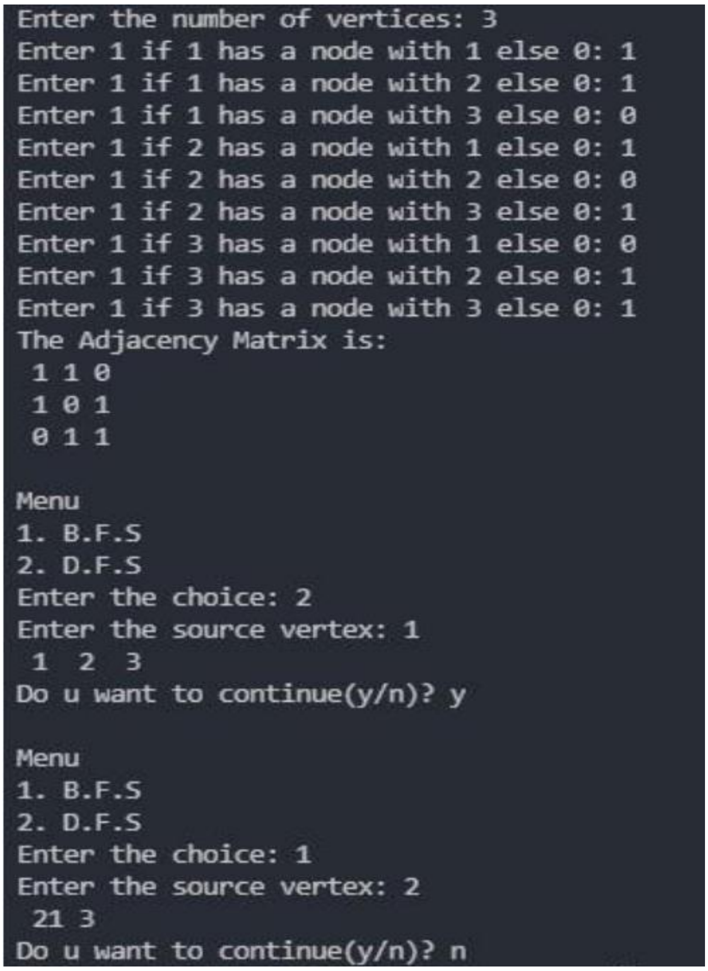
```
printf("\n");
}
do{ for (i = 1; i <= n; i++)
vis[i] = 0;
printf("\nMenu"); printf("\n1.
B.F.S"); printf("\n2. D.F.S");
printf("\nEnter the choice: ");
scanf("%d", &ch); printf("Enter the
source vertex: "); scanf("%d", &s);
switch (ch){
 case 1: bfs(s, n); break;
case 2: dfs(s, n);
 break;
}
printf("\nDo u want to continue(y/n)? ");
scanf("%c", &dummy); scanf("%c", &c);
} while ((c == 'y') || (c == 'Y'));
}
void bfs(int s, int n){
int p, i;
add(s);
vis[s] = 1;
p = delete
(); if (p !=
0)
printf(" %d", p); while
(p != 0){
```

```
for (i = 1; i <= n; i++) if ((a[p][i] !=
0) && (vis[i] == 0)){
add(i); vis[i]
= 1;
}
p = delete (); if
(p != 0)
printf("%d ", p);
}
for (i = 1; i <= n; i++)
if (vis[i] == 0)
bfs(i, n);
}
void add(int item){
if (rear == 19) printf("Queue
full.."); else{
if (rear == -1){
q[++rear] = item; front++;
}
else
q[++rear] = item;
}
}
int delete (){
int k;
if ((front > rear) || (front == -1)) return
(0);
```

```
else{ k =
q[front++];
return (k);
}
}
void dfs(int s, int n){
int i, k;
push(s); vis[s] = 1;
k = pop(); if (k != 0)
printf(" %d ", k);
while (k != 0){
for (i = 1; i <= n; i++) if ((a[k][i] !=
0) && (vis[i] == 0)){ push(i);
vis[i] = 1;
}
k = pop();
if (k != 0)
printf(" %d ", k);
}
for (i = 1; i <= n; i++) if
(vis[i] == 0)
dfs(i, n);
}
void push(int item){ if
(top == 19)
printf("Stack overflow...");
else
```

```
stack[++top] = item;
}
int pop(){
int k;
if (top == -1) return
(0);
else{ k = stack[top--
]; return (k);
}
}
```

OUTPUT:



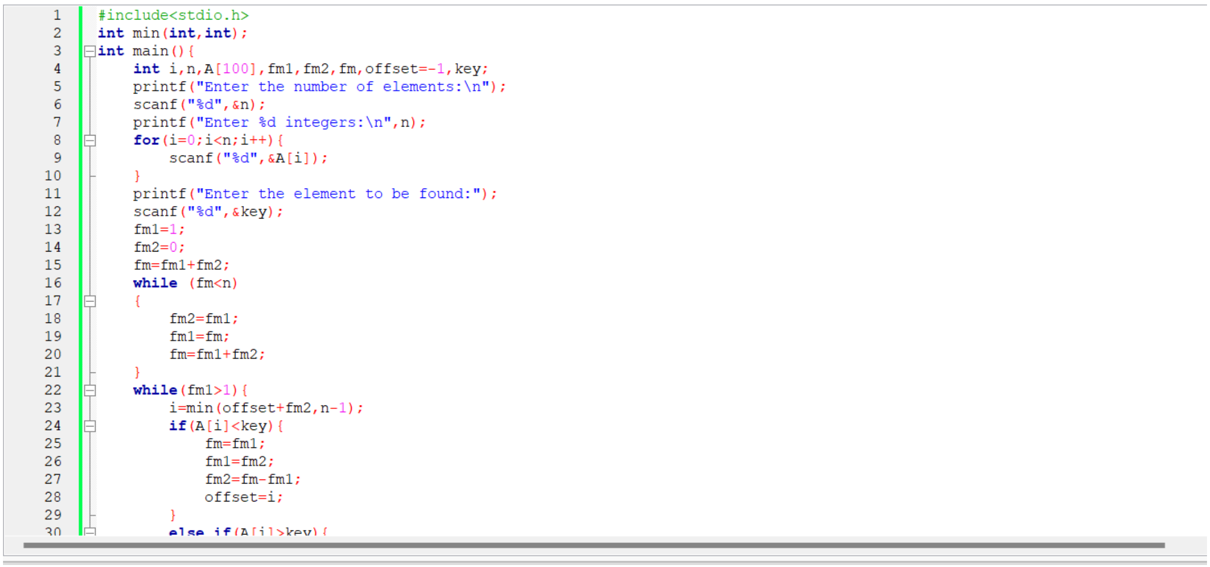
```
Enter the number of vertices: 3
Enter 1 if 1 has a node with 1 else 0: 1
Enter 1 if 1 has a node with 2 else 0: 1
Enter 1 if 1 has a node with 3 else 0: 0
Enter 1 if 2 has a node with 1 else 0: 1
Enter 1 if 2 has a node with 2 else 0: 0
Enter 1 if 2 has a node with 3 else 0: 1
Enter 1 if 3 has a node with 1 else 0: 0
Enter 1 if 3 has a node with 2 else 0: 1
Enter 1 if 3 has a node with 3 else 0: 1
The Adjacency Matrix is:
 1 1 0
 1 0 1
 0 1 1

Menu
1. B.F.S
2. D.F.S
Enter the choice: 2
Enter the source vertex: 1
 1  2  3
Do u want to continue(y/n)? y

Menu
1. B.F.S
2. D.F.S
Enter the choice: 1
Enter the source vertex: 2
 21 3
Do u want to continue(y/n)? n
```

EXPERIMENT 9

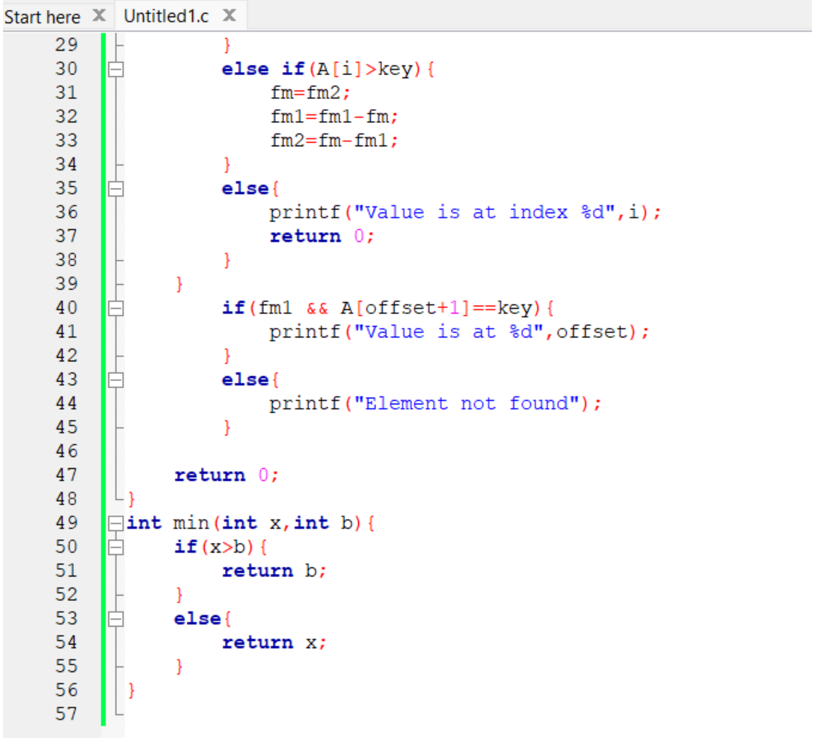
Problem statement 9(a): Write a c program to implement various types of searching techniques

CODE :

FIBONACCI SEARCH

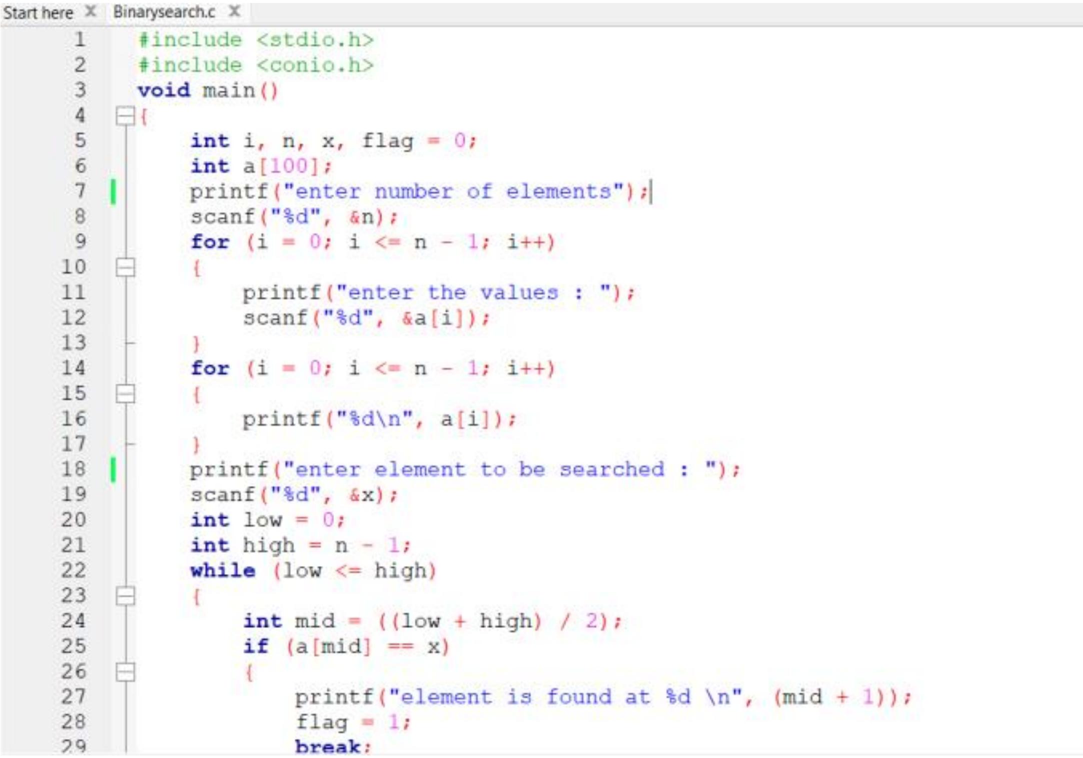


```c
#include<stdio.h>
int min(int,int);
int main(){
    int i,n,A[100],fm1,fm2,fm,offset=-1,key;
    printf("Enter the number of elements:\n");
    scanf("%d",&n);
    printf("Enter %d integers:\n",n);
    for(i=0;i<n;i++){
        scanf("%d",&A[i]);
    }
    printf("Enter the element to be found:");
    scanf("%d",&key);
    fm1=1;
    fm2=0;
    fm=fm1+fm2;
    while (fm<n)
    {
        fm2=fm1;
        fm1=fm;
        fm=fm1+fm2;
    }
    while(fm1>1){
        i=min(offset+fm2,n-1);
        if(A[i]<key){
            fm=fm1;
            fm1=fm2;
            fm2=fm-fm1;
            offset=i;
        }
        else if(A[i]>key){
```

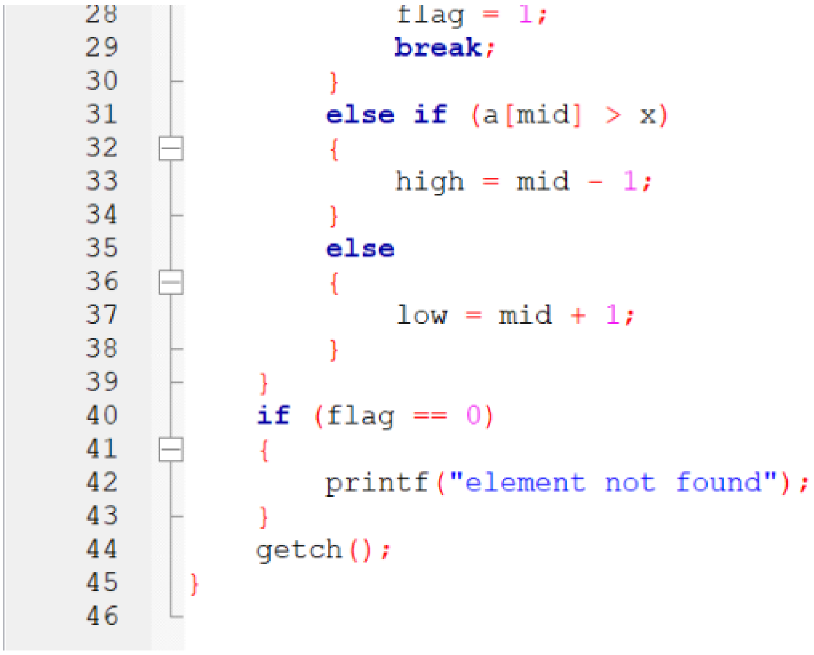


```c
        }
        else if(A[i]>key){
            fm=fm2;
            fm1=fm1-fm;
            fm2=fm-fm1;
        }
        else{
            printf("Value is at index %d",i);
            return 0;
        }
    }
        if(fm1 && A[offset+1]==key){
            printf("Value is at %d",offset);
        }
        else{
            printf("Element not found");
        }

    return 0;
}
int min(int x,int b){
    if(x>b){
        return b;
    }
    else{
        return x;
    }
}
```

BINARY SEARCH



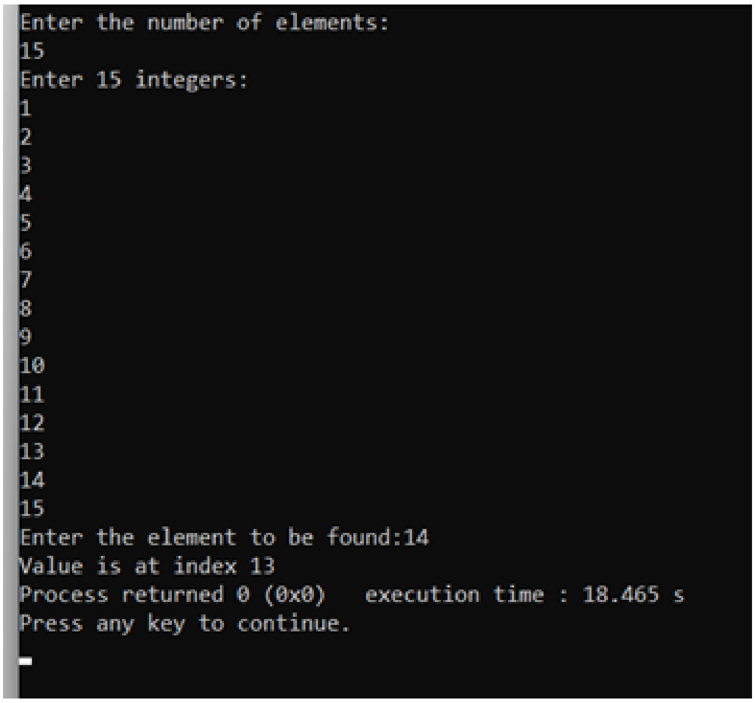
```c
#include <stdio.h>
#include <conio.h>
void main()
{
    int i, n, x, flag = 0;
    int a[100];
    printf("enter number of elements");
    scanf("%d", &n);
    for (i = 0; i <= n - 1; i++)
    {
        printf("enter the values : ");
        scanf("%d", &a[i]);
    }
    for (i = 0; i <= n - 1; i++)
    {
        printf("%d\n", a[i]);
    }
    printf("enter element to be searched : ");
    scanf("%d", &x);
    int low = 0;
    int high = n - 1;
    while (low <= high)
    {
        int mid = ((low + high) / 2);
        if (a[mid] == x)
        {
            printf("element is found at %d \n", (mid + 1));
            flag = 1;
            break;
```



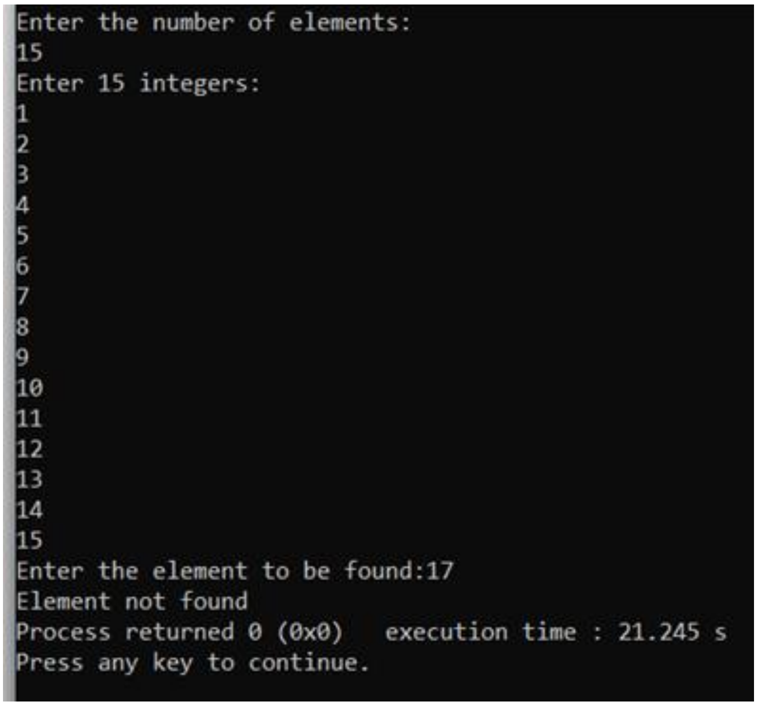
```c
            flag = 1;
            break;
        }
        else if (a[mid] > x)
        {
            high = mid - 1;
        }
        else
        {
            low = mid + 1;
        }
    }
    if (flag == 0)
    {
        printf("element not found");
    }
    getch();
}
```

OUTPUT:

FIBONACCI SEARCH:

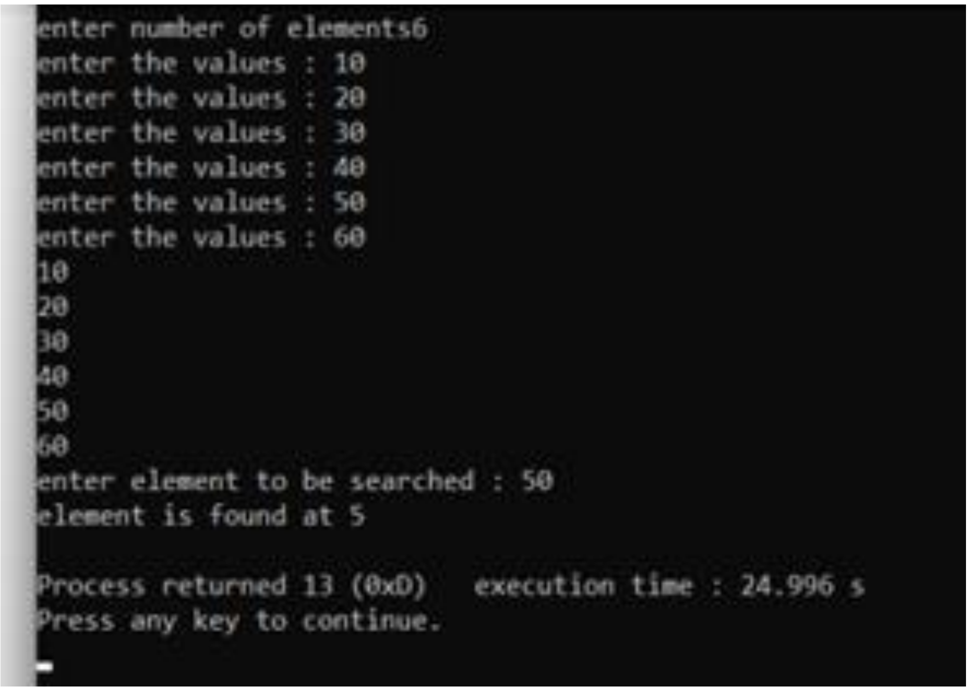


```
Enter the number of elements:
15
Enter 15 integers:
1
2
3
4
5
6
7
8
9
10
11
12
13
14
15
Enter the element to be found:14
Value is at index 13
Process returned 0 (0x0)   execution time : 18.465 s
Press any key to continue.
```

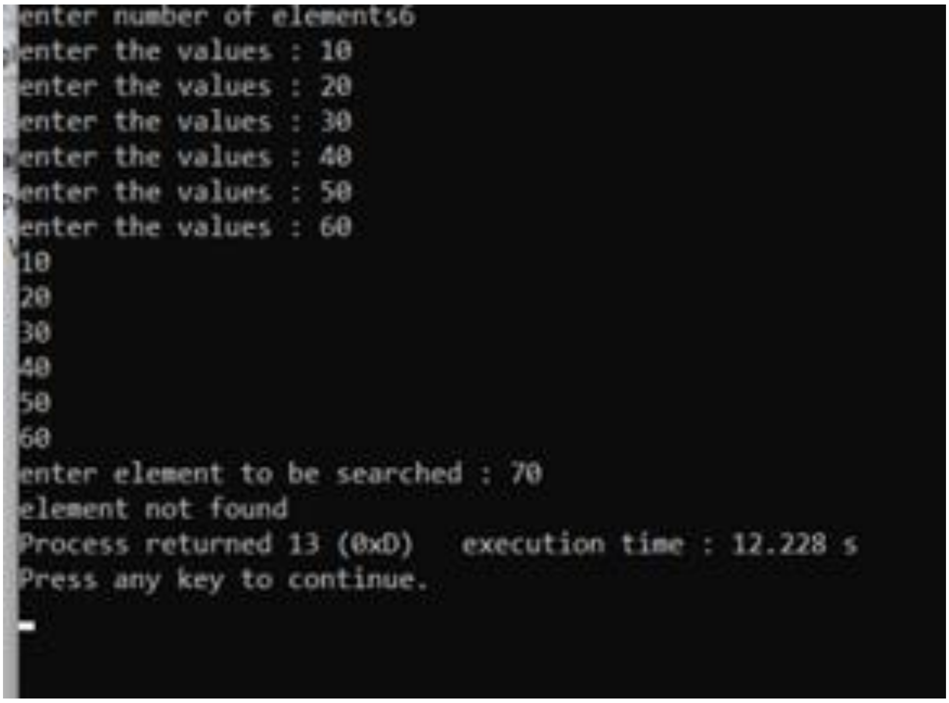


```
Enter the number of elements:
15
Enter 15 integers:
1
2
3
4
5
6
7
8
9
10
11
12
13
14
15
Enter the element to be found:17
Element not found
Process returned 0 (0x0)   execution time : 21.245 s
Press any key to continue.
```

BINARY SEARCH



```
enter number of elements6
enter the values : 10
enter the values : 20
enter the values : 30
enter the values : 40
enter the values : 50
enter the values : 60
10
20
30
40
50
60
enter element to be searched : 50
element is found at 5

Process returned 13 (0xD)   execution time : 24.996 s
Press any key to continue.
```

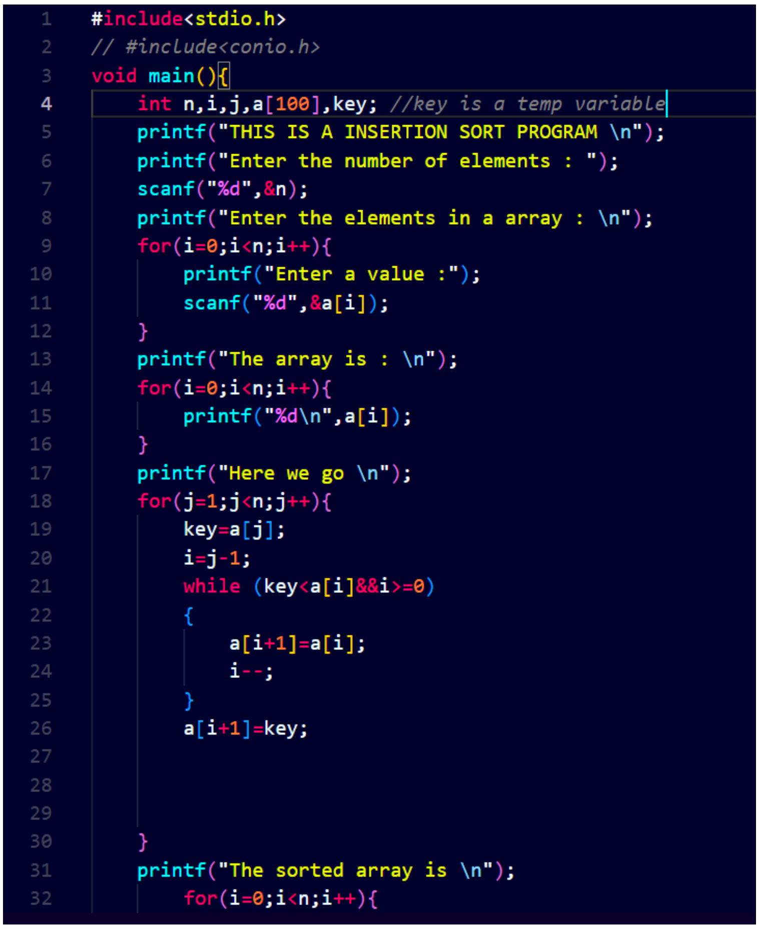


```
enter number of elements6
enter the values : 10
enter the values : 20
enter the values : 30
enter the values : 40
enter the values : 50
enter the values : 60
10
20
30
40
50
60
enter element to be searched : 70
element not found
Process returned 13 (0xD)   execution time : 12.228 s
Press any key to continue.
```

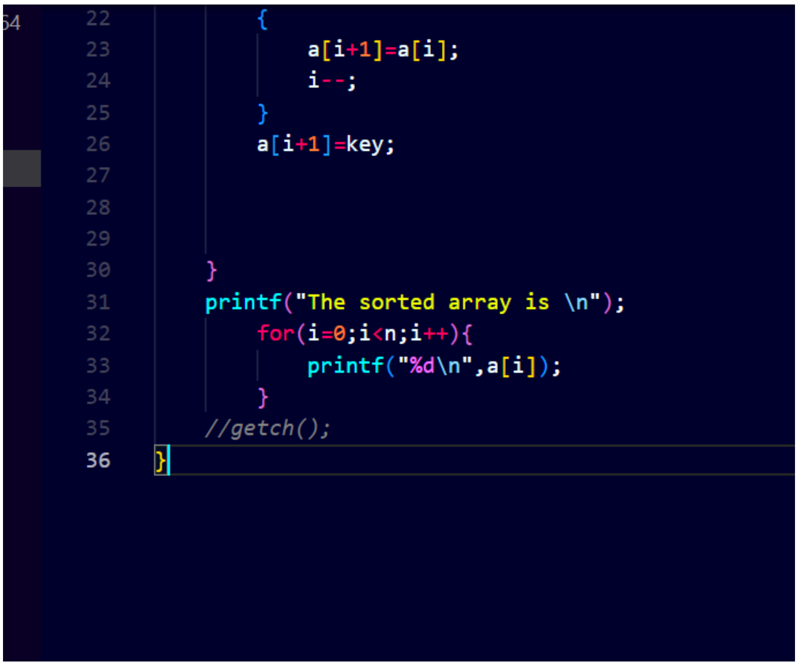
Problem statement 9(b): Write a c program to implement various types of sorting techniques

CODE :

INSERTION SORT

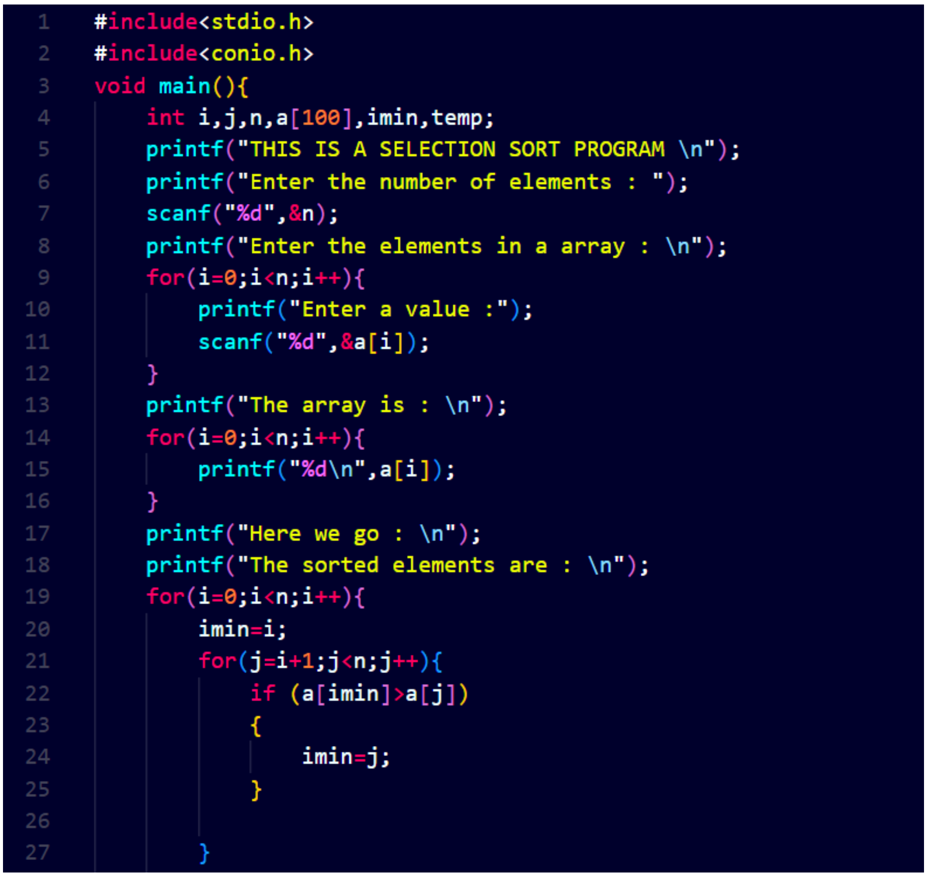


```c
#include<stdio.h>
// #include<conio.h>
void main(){
    int n,i,j,a[100],key; //key is a temp variable
    printf("THIS IS A INSERTION SORT PROGRAM \n");
    printf("Enter the number of elements : ");
    scanf("%d",&n);
    printf("Enter the elements in a array : \n");
    for(i=0;i<n;i++){
        printf("Enter a value :");
        scanf("%d",&a[i]);
    }
    printf("The array is : \n");
    for(i=0;i<n;i++){
        printf("%d\n",a[i]);
    }
    printf("Here we go \n");
    for(j=1;j<n;j++){
        key=a[j];
        i=j-1;
        while (key<a[i]&&i>=0)
        {
            a[i+1]=a[i];
            i--;
        }
        a[i+1]=key;



    }
    printf("The sorted array is \n");
        for(i=0;i<n;i++){
```

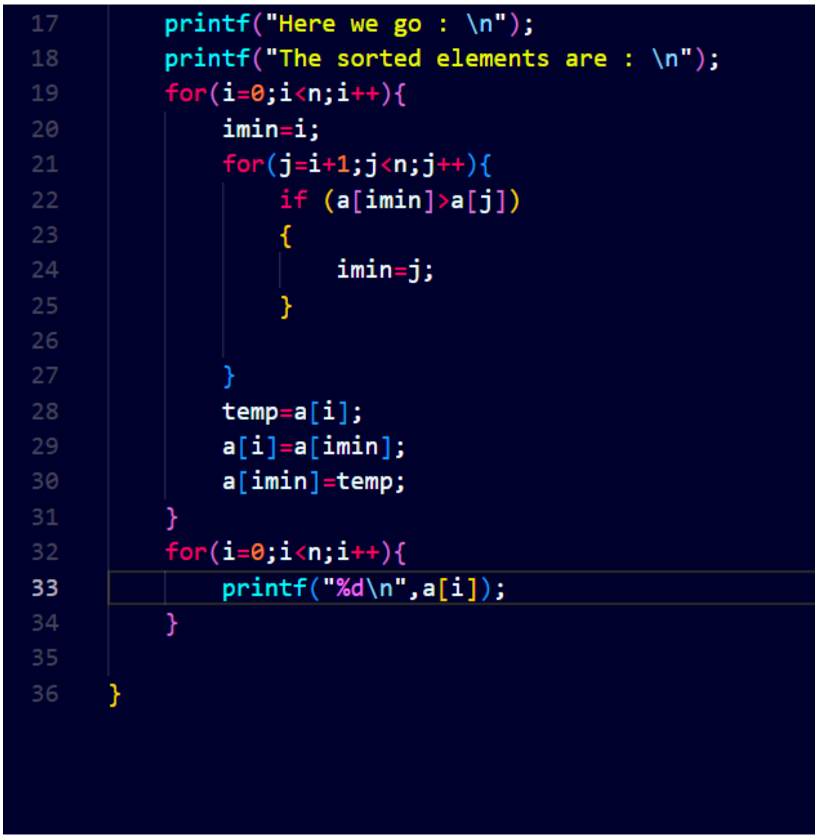


```c
        {
            a[i+1]=a[i];
            i--;
        }
        a[i+1]=key;



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

SELECTION SORT:



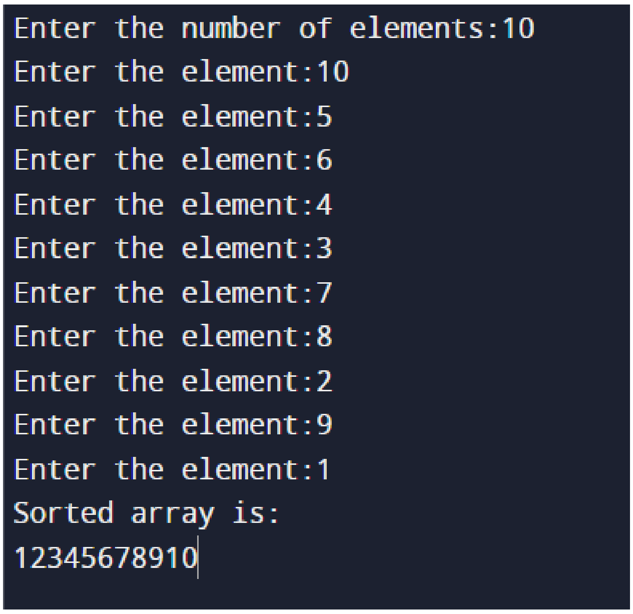
```c
#include<stdio.h>
#include<conio.h>
void main(){
    int i,j,n,a[100],imin,temp;
    printf("THIS IS A SELECTION SORT PROGRAM \n");
    printf("Enter the number of elements : ");
    scanf("%d",&n);
    printf("Enter the elements in a array : \n");
    for(i=0;i<n;i++){
        printf("Enter a value :");
        scanf("%d",&a[i]);
    }
    printf("The array is : \n");
    for(i=0;i<n;i++){
        printf("%d\n",a[i]);
    }
    printf("Here we go : \n");
    printf("The sorted elements are : \n");
    for(i=0;i<n;i++){
        imin=i;
        for(j=i+1;j<n;j++){
            if (a[imin]>a[j])
            {
                imin=j;
            }

        }
```



```c
    printf("Here we go : \n");
    printf("The sorted elements are : \n");
    for(i=0;i<n;i++){
        imin=i;
        for(j=i+1;j<n;j++){
            if (a[imin]>a[j])
            {
                imin=j;
            }

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

}
```

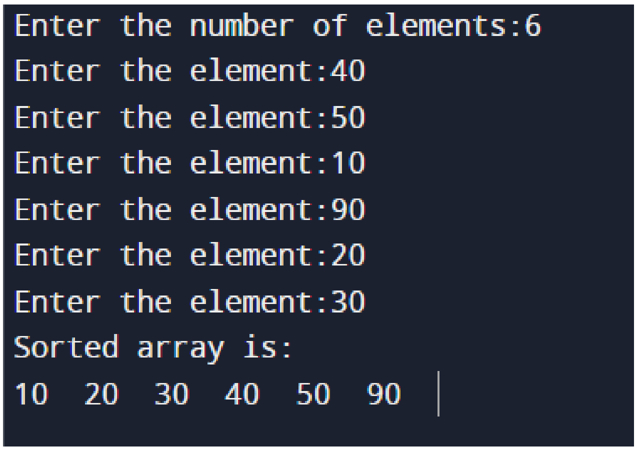
OUTPUT :

INSERTION SORT



```
Enter the number of elements:10
Enter the element:10
Enter the element:5
Enter the element:6
Enter the element:4
Enter the element:3
Enter the element:7
Enter the element:8
Enter the element:2
Enter the element:9
Enter the element:1
Sorted array is:
12345678910
```

SELECTION SORT:



```
Enter the number of elements:6
Enter the element:40
Enter the element:50
Enter the element:10
Enter the element:90
Enter the element:20
Enter the element:30
Sorted array is:
10  20  30  40  50  90
```

EXPERIMENT 10

Problem statement 10: Write a c program to implement various

Hashing techniques

CODE:

LINEAR PROBING

```
#include<stdio.h>
int ht[10], i ,found=0, key;

void insert_val()
{
   int val, f=0;
   printf("\nEnter the element to be inserted:");
   scanf("%d",&val);
   key = ( val%10 );
   if( ht[key]== -1)
   {
      ht[key] = val;
   }
   else
   {
      for( i=1; i<10; i++)
      {
         key = ( val%10 );
         key=(key+i)%10;
         if( ht[key]== -1)
         {
            ht[key]=val;
            break;
         }
      }
   }
}

void display()
{
   for(i=0; i<10; i++)
   {
      printf("\t%d", ht[i]);
   }
}
```

```
void search_val()
{
   int val, flag=0;
   printf("\nEnter the element to be searched:");
   scanf("%d",&val);
   key = ( val %10 );
   if( ht[key]== val)
   {
      flag = 1;
   }
   else
   {
      for( i=1; i<10; i++)
      {
         key=(key+i)%10;
         if( ht[key]== val)
         {
            flag = 1;
            break;
         }
      }
   }
   if( flag == 1)
   {
      found=1;
      printf("\n The item searched was found at position %d!", key+1);
   }
   else
   {
      key=-1;
      printf("\n The item searched was not found in the hash table");
   }
}

void delete_val()
{
   search_val();
   if(found==1)
   {
      if( key != -1)
      {
         printf("\n The element deleted id %d", ht[key]);
```

```
            ht[ key ] =-1;
        }
    }
}

void main()
{
    int choice;
    for( i=0; i<10; i++)
    {
        ht[i]=-1;
    }
    do
    {
        printf("Choices
are:\n1.Insert\n2.Search\n3.Delete\n4.Display\n5.Exit\nEnter your choice:");
        scanf("%d",&choice);
        switch(choice)
        {
            case 1:
            insert_val();
            break;
            case 2:
            search_val();
            break;
            case 3:
            delete_val();
            break;
            case 4:
            display();
            break;
            default:
            printf("Invalid choice\n");
        }
    } while (choice!=5);
}
```

QUADRATIC PROBING

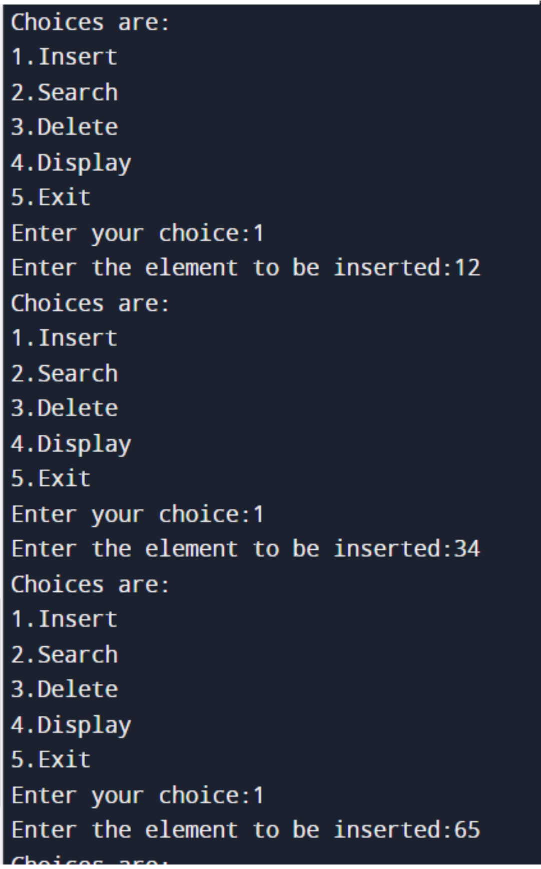
```
#include<stdio.h>
int ht[10], i ,found=0, key;
int c1=1,c2=3;

void insert_val()
{
   int val, f=0;
   printf("\nEnter the element to be inserted:");
   scanf("%d",&val);
   key = ( val%10 );
   if( ht[key]== -1)
   {
      ht[key] = val;
   }
   else
   {
      for( i=1; i<10; i++)
      {
         key = ( val%10 );
         key=(key+i*c1+i*i*c2)%10;
         if( ht[key]== -1)
         {
            ht[key]=val;
            break;
         }
      }
   }
}

void display()
{
   for(i=0; i<10; i++)
   {
      printf("\t%d", ht[i]);
   }
}

void search_val()
{
   int val, flag=0;
   printf("\nEnter the element to be searched:");
```

```
    scanf("%d",&val);
    key = ( val %10 );
    if( ht[key]== val)
    {
        flag = 1;
    }
    else
    {
        for( i=1; i<10; i++)
        {
            key = ( val %10 );
            key=(key+i*c1+i*i*c2)%10;
            if( ht[key]== val)
            {
                flag = 1;
                break;
            }
        }
    }
    if( flag == 1)
    {
        found=1;
        printf("\n The item searched was found at position %d!", key+1);
    }
    else
    {
        key=-1;
        printf("\n The item searched was not found in the hash table");
    }
}

void delete_val()
{
    search_val();
    if(found==1)
    {
        if( key != -1)
        {
            printf("\n The element deleted id %d", ht[key]);
            ht[ key ] =-1;
        }
    }
}
```

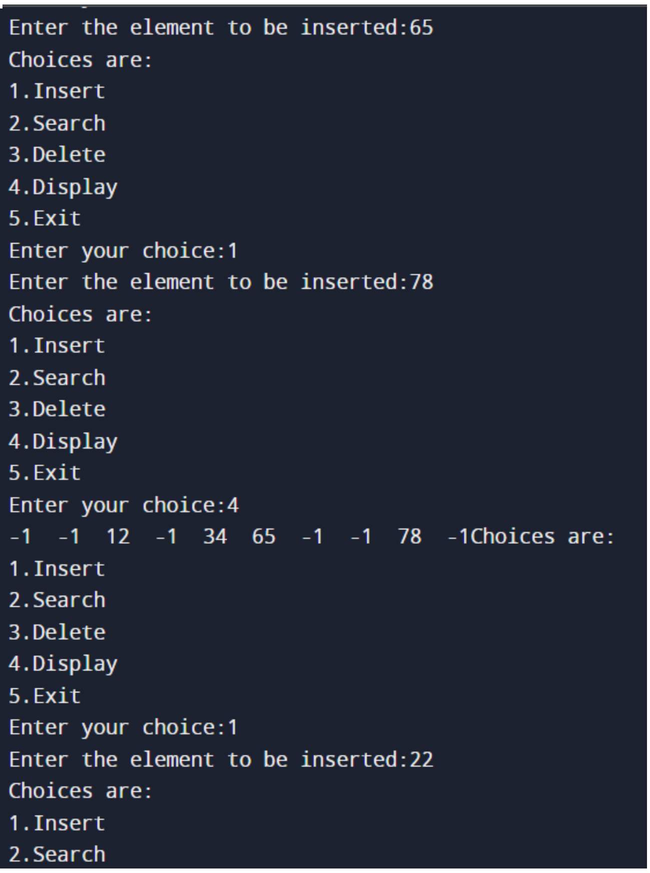
```
void main()
{
    int choice;
    for( i=0; i<10; i++)
    {
        ht[i]=-1;
    }
    do
    {
        printf("Choices are:\n1.Insert\n2.Search\n3.Delete\n4.Display\n5.Exit\nEnter your choice:");
        scanf("%d",&choice);
        switch(choice)
        {
            case 1:
            insert_val();
            break;
            case 2:
            search_val();
            break;
            case 3:
            delete_val();
            break;
            case 4:
            display();
            break;
            default:
            printf("Invalid choice\n");
        }
    } while (choice!=5);
}
```

OUTPUT:

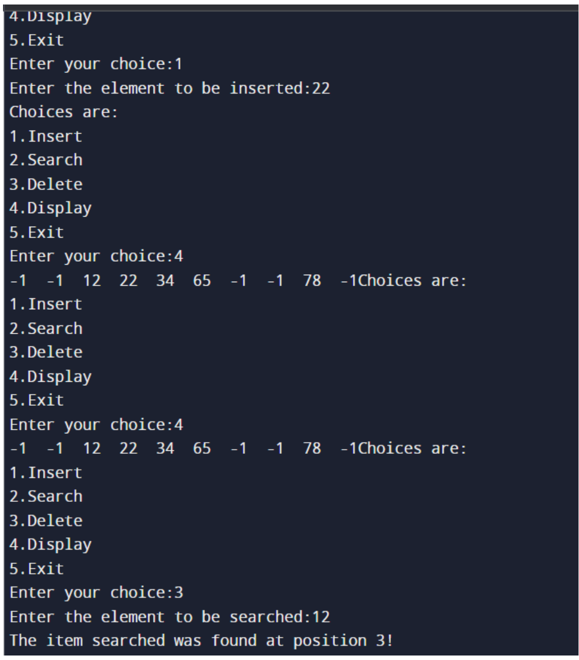
LINEAR PROBING



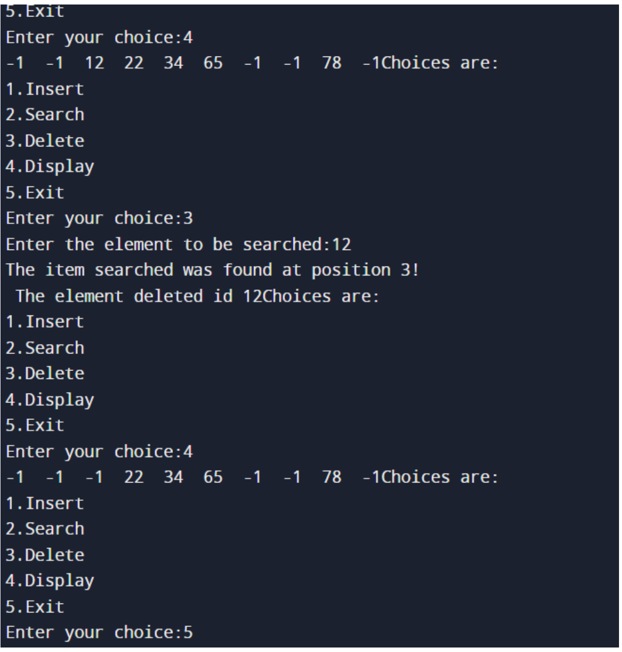
```
Choices are:
1.Insert
2.Search
3.Delete
4.Display
5.Exit
Enter your choice:1
Enter the element to be inserted:12
Choices are:
1.Insert
2.Search
3.Delete
4.Display
5.Exit
Enter your choice:1
Enter the element to be inserted:34
Choices are:
1.Insert
2.Search
3.Delete
4.Display
5.Exit
Enter your choice:1
Enter the element to be inserted:65
Choices are:
```



```
Enter the element to be inserted:65
Choices are:
1.Insert
2.Search
3.Delete
4.Display
5.Exit
Enter your choice:1
Enter the element to be inserted:78
Choices are:
1.Insert
2.Search
3.Delete
4.Display
5.Exit
Enter your choice:4
-1  -1  12  -1  34  65  -1  -1  78  -1Choices are:
1.Insert
2.Search
3.Delete
4.Display
5.Exit
Enter your choice:1
Enter the element to be inserted:22
Choices are:
1.Insert
2.Search
```

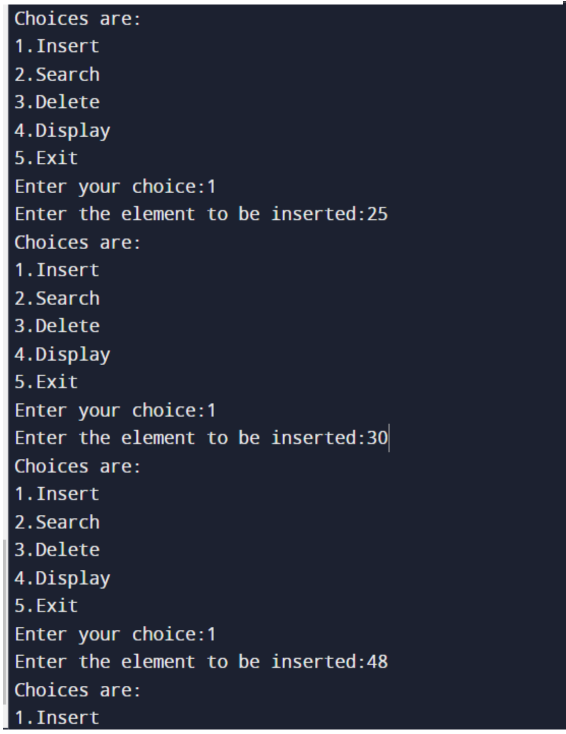


```
4.Display
5.Exit
Enter your choice:1
Enter the element to be inserted:22
Choices are:
1.Insert
2.Search
3.Delete
4.Display
5.Exit
Enter your choice:4
-1  -1  12  22  34  65  -1  -1  78  -1Choices are:
1.Insert
2.Search
3.Delete
4.Display
5.Exit
Enter your choice:4
-1  -1  12  22  34  65  -1  -1  78  -1Choices are:
1.Insert
2.Search
3.Delete
4.Display
5.Exit
Enter your choice:3
Enter the element to be searched:12
The item searched was found at position 3!
```

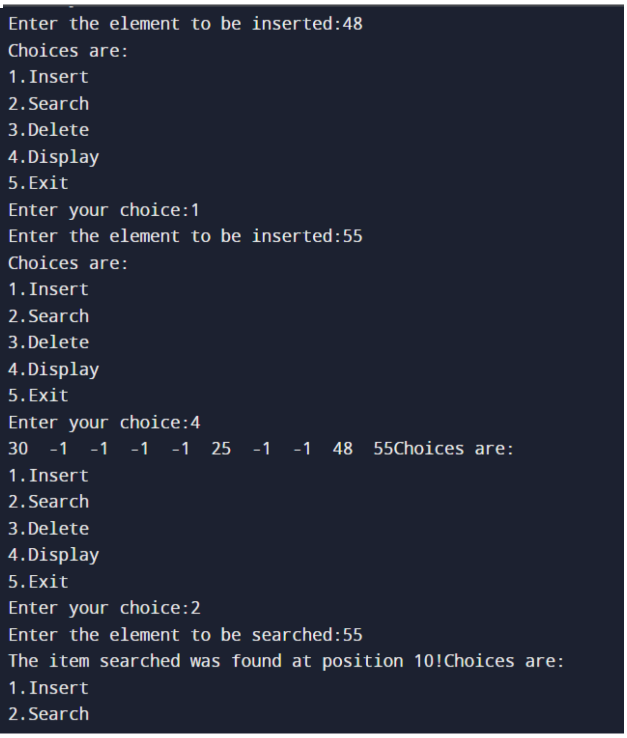


```
5.Exit
Enter your choice:4
-1  -1  12  22  34  65  -1  -1  78  -1Choices are:
1.Insert
2.Search
3.Delete
4.Display
5.Exit
Enter your choice:3
Enter the element to be searched:12
The item searched was found at position 3!
 The element deleted id 12Choices are:
1.Insert
2.Search
3.Delete
4.Display
5.Exit
Enter your choice:4
-1  -1  -1  22  34  65  -1  -1  78  -1Choices are:
1.Insert
2.Search
3.Delete
4.Display
5.Exit
Enter your choice:5
```

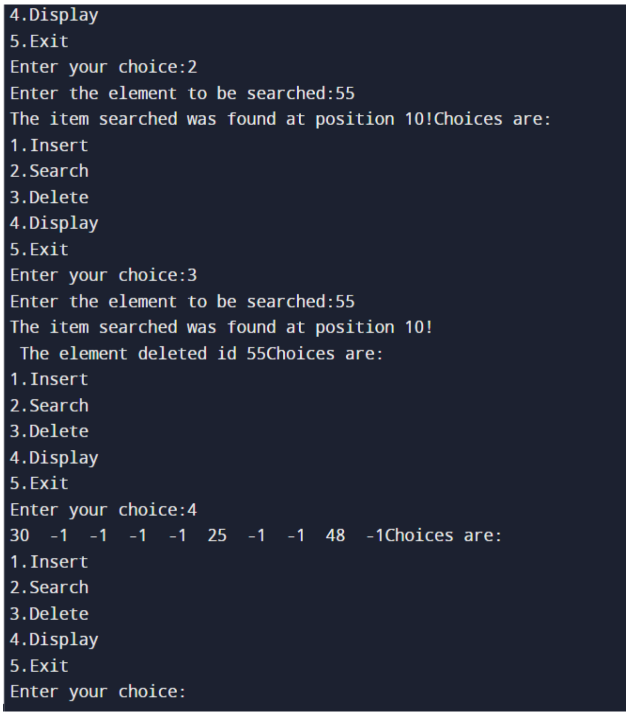
QUADRATIC PROBING



```
Choices are:
1.Insert
2.Search
3.Delete
4.Display
5.Exit
Enter your choice:1
Enter the element to be inserted:25
Choices are:
1.Insert
2.Search
3.Delete
4.Display
5.Exit
Enter your choice:1
Enter the element to be inserted:30
Choices are:
1.Insert
2.Search
3.Delete
4.Display
5.Exit
Enter your choice:1
Enter the element to be inserted:48
Choices are:
1.Insert
```



```
Enter the element to be inserted:48
Choices are:
1.Insert
2.Search
3.Delete
4.Display
5.Exit
Enter your choice:1
Enter the element to be inserted:55
Choices are:
1.Insert
2.Search
3.Delete
4.Display
5.Exit
Enter your choice:4
30  -1  -1  -1  -1  25  -1  -1  48  55Choices are:
1.Insert
2.Search
3.Delete
4.Display
5.Exit
Enter your choice:2
Enter the element to be searched:55
The item searched was found at position 10!Choices are:
1.Insert
2.Search
```



```
4.Display
5.Exit
Enter your choice:2
Enter the element to be searched:55
The item searched was found at position 10!Choices are:
1.Insert
2.Search
3.Delete
4.Display
5.Exit
Enter your choice:3
Enter the element to be searched:55
The item searched was found at position 10!
 The element deleted id 55Choices are:
1.Insert
2.Search
3.Delete
4.Display
5.Exit
Enter your choice:4
30  -1  -1  -1  -1  25  -1  -1  48  -1Choices are:
1.Insert
2.Search
3.Delete
4.Display
5.Exit
Enter your choice:
```