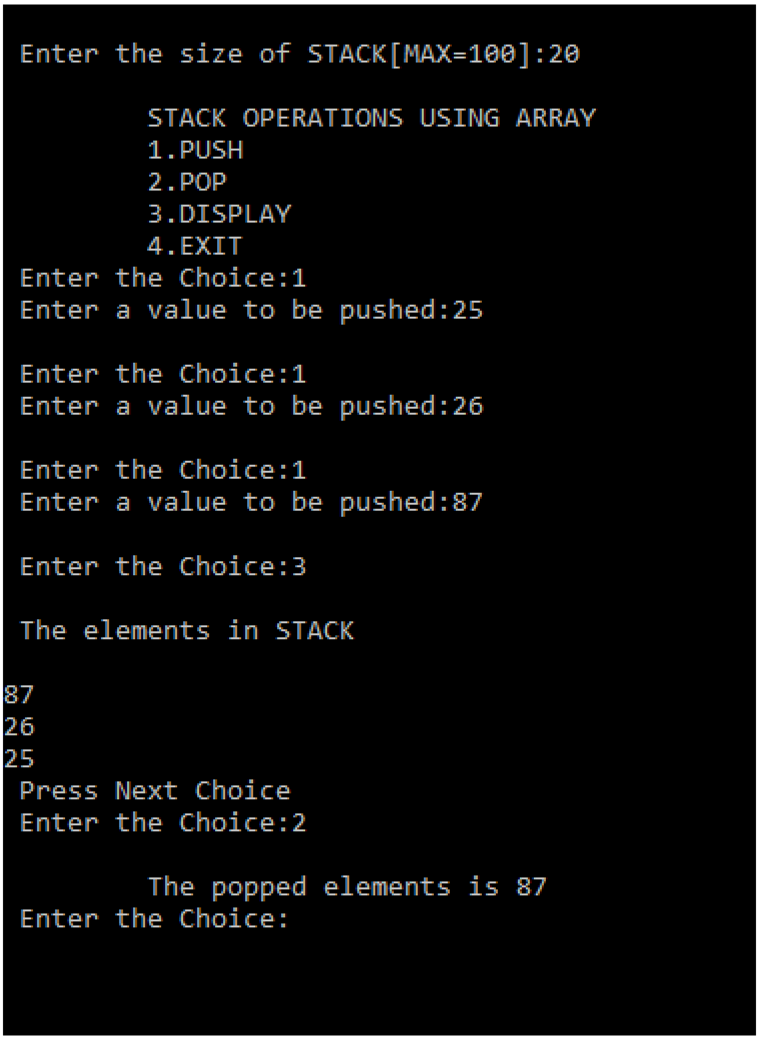
Q. STACK IMPLEMENTATION.

CODE:

```
#include<stdio.h>
int stack[100],choice,n,top,x,i;
void push(void);
void pop(void);
void display(void);
int main()
{
    top=-1;
    printf("\n Enter the size of STACK[MAX=100]:");
    scanf("%d",&n);
    printf("\n\t STACK OPERATIONS USING ARRAY");
    printf("\n\t 1.PUSH\n\t 2.POP\n\t 3.DISPLAY\n\t 4.EXIT");
    do
    {
        printf("\n Enter the Choice:");
        scanf("%d",&choice);
        switch(choice)
        {
            case 1:
            {
                push();
                break;
            }
            case 2:
            {
                pop();
                break;

            }

            case 3:

            {

                display();

                break;

            }

            case 4:

            {

                printf("\n\t EXIT POINT ");

                break;

            }

            default:

            {

                printf ("\n\t Please Enter a Valid Choice(1/2/3/4)");

            }


        }

    }

    while(choice!=4);

    return 0;

}

void push()

{

    if(top>=n-1)

    {

        printf("\n\tSTACK is over flow");


    }
    else

    {

        printf(" Enter a value to be pushed:");

        scanf("%d",&x);

        top++;

        stack[top]=x;

    }

}

void pop()

{

    if(top<=-1)

    {

        printf("\n\t Stack is under flow");

    }

    else

    {

        printf("\n\t The popped elements is %d",stack[top]);

        top--;

    }

}

void display()

{

    if(top>=0)

    {

        printf("\n The elements in STACK \n");

        for(i=top; i>=0; i--)

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

        printf("\n Press Next Choice");

    }
```

```
   else

   {

       printf("\n The STACK is empty");

   }



}
```



```
 Enter the size of STACK[MAX=100]:20

         STACK OPERATIONS USING ARRAY
         1.PUSH
         2.POP
         3.DISPLAY
         4.EXIT
 Enter the Choice:1
 Enter a value to be pushed:25

 Enter the Choice:1
 Enter a value to be pushed:26

 Enter the Choice:1
 Enter a value to be pushed:87

 Enter the Choice:3

 The elements in STACK

87
26
25
 Press Next Choice
 Enter the Choice:2

         The popped elements is 87
 Enter the Choice:
```

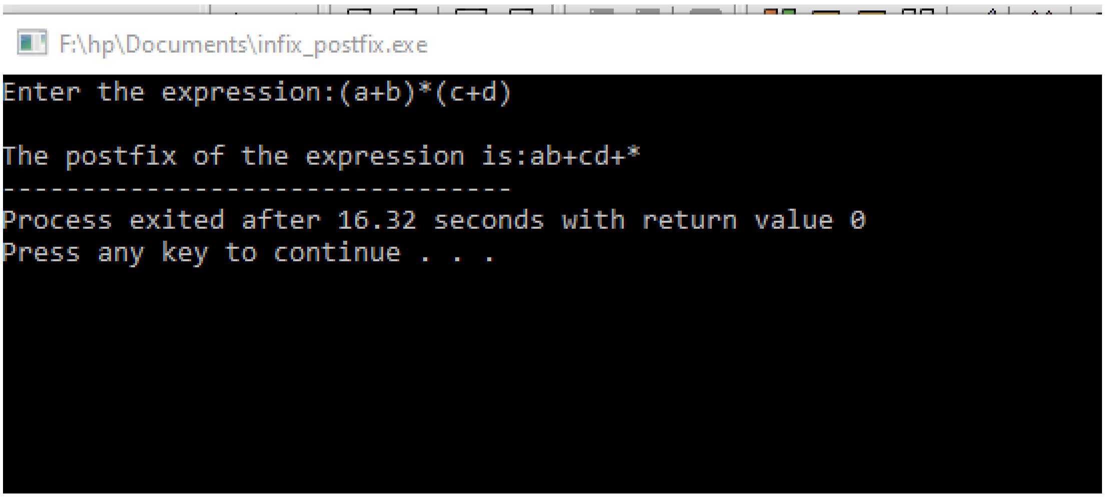
1BM19ET004 **LAB- 2** **AKASH SHRIVASTAVA(3C)**

Q. INFIX TO POSTFIX CONVERSION USING STACK.

```
#include<stdio.h>
#include<string.h>
#include<stdlib.h>
#include<process.h>

int F(char symbol)
{
        switch(symbol)
        {
                case'+':
                case'-': return 2;
                case'*':
                case'/': return 4;
                case'^':
                case'$': return 5;
                case'(': return 0;
                case'#': return -1;
                default: return 8;
        }
}

int G(char symbol)
{
        switch(symbol)
        {
                case'+':
                case'-': return 1;
        case'*':
        case'/': return 3;
        case'^':
        case'$': return 6;
        case'(': return 9;
        case')': return 0;
        case'#': return -1;
        default: return 7;
    }
}

void conversion(char infix[],char postfix[])
{
    int top,i,j;
    char s[50],symbol;
    top=-1;
    s[++top]='#';
    j=0;
    for(i=0;i<strlen(infix);i++)
    {
        symbol=infix[i];
        while(F(s[top])>G(symbol))
        {
            postfix[j]=s[top--];
            j++;
        }
        if(F(s[top])!=G(symbol))
        {
            s[++top]=symbol;
```

```
                }
                else
                top--;
        }
        while(s[top]!='#')
        {
                postfix[j++]=s[top--];
        }
        postfix[j]='\0';
}
int main()
{
        char infix[20];
        char postfix[20];
        printf("Enter the expression:");
        scanf("%s",infix);
        conversion(infix,postfix);
        printf("\nThe postfix of the expression is:%s ",postfix);
        return 0;
}
```

OUTPUT:



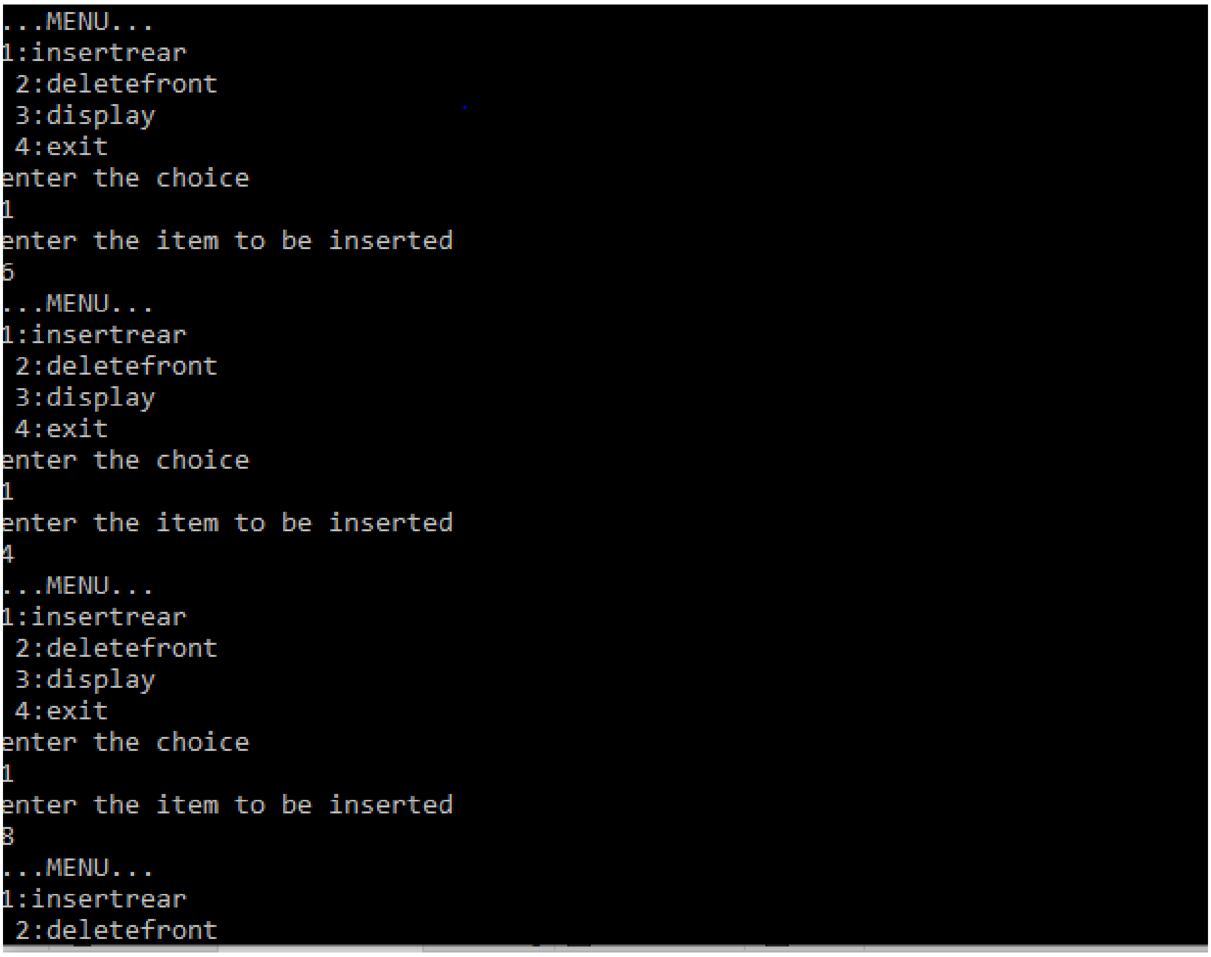
```
Enter the expression:(a+b)*(c+d)

The postfix of the expression is:ab+cd+*
--------------------------------
Process exited after 16.32 seconds with return value 0
Press any key to continue . . .
```

1BM19ET004 **LAB- 3** **AKASH SHRIVASTAVA(3C)**

Q. SIMPLE QUEUE IMPLEMENTATION.

```
#include<stdio.h>

#include<stdlib.h>

#define QUE_SIZE 5

int item,front=0,rear=-1,q[10];

void insertrear()

{if(rear==QUE_SIZE-1)

{

        printf("queue overflow\n");

        return;

}

rear=rear+1;

q[rear]=item;

}int deletefront()

{if (front>rear)

{front=0;

rear=-1;

return -1;

}return q[front++];
```

```
}void displayQ()
{int i;
if (front>rear)
{
        printf("queue is empty\n");
        return;
}
printf("contents of queue\n");
for(i=front;i<=rear;i++)
{
        printf("%d\n",q[i]);
}}
int main()
{
        int choice;
        for(;;)
        {
                printf("...MENU...\n1:insertrear\n 2:deletefront\n 3:display\n 4:exit\n");
                printf("enter the choice\n");
                scanf("%d",&choice);
                switch(choice)
                {
                        case 1:printf("enter the item to be inserted\n");
                        scanf("%d",&item);
                        insertrear ();
                        break;
                        case 2:item=deletefront();
                        if(item==-1)
                        printf("queue is empty\n");
                        else

                        printf("item deleted=%d\n",item);

                        break;

                        case 3:displayQ();

                        break;

                        default:exit (0);



                }



        }
```



```
...MENU...
1:insertrear
 2:deletefront
 3:display
 4:exit
enter the choice
1
enter the item to be inserted
6
...MENU...
1:insertrear
 2:deletefront
 3:display
 4:exit
enter the choice
1
enter the item to be inserted
4
...MENU...
1:insertrear
 2:deletefront
 3:display
 4:exit
enter the choice
1
enter the item to be inserted
3
...MENU...
1:insertrear
 2:deletefront
```

```
}
```

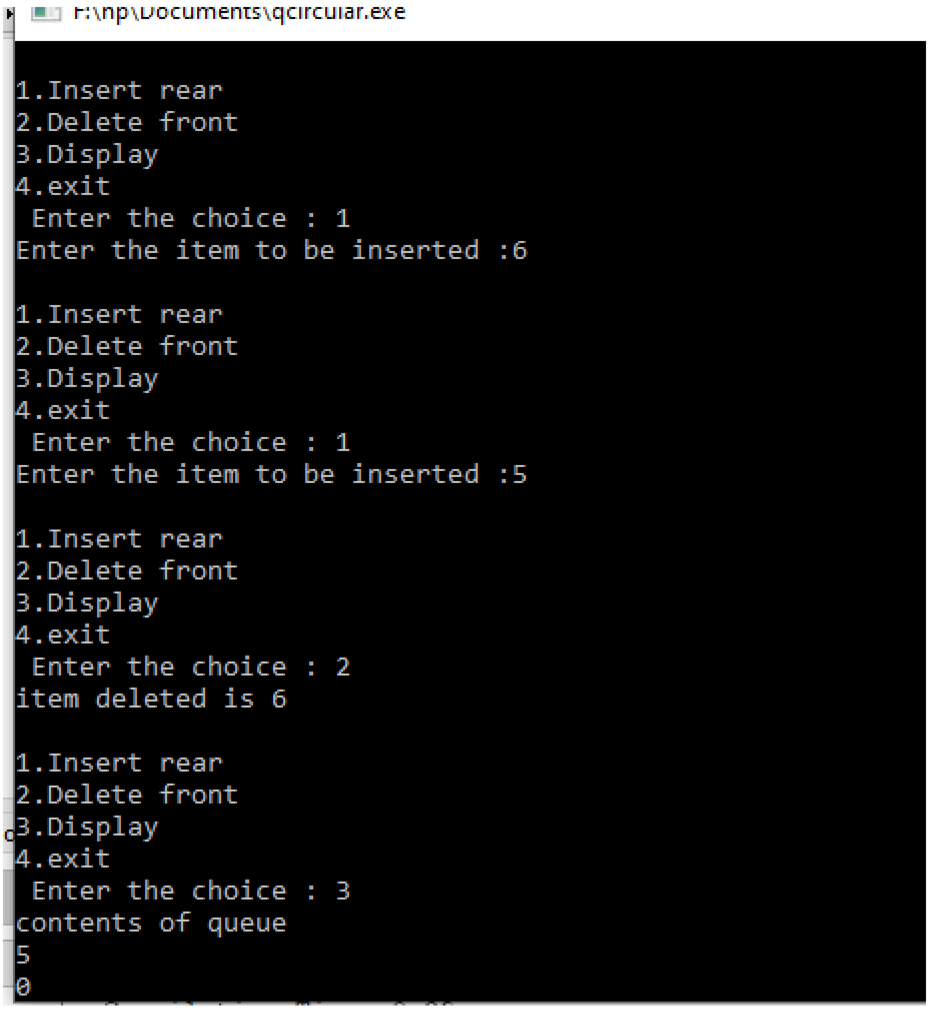
1BM19ET004 LAB- 4 AKASH SHRIVASTAVA(3C)

Q.CIRCULAR QUEUE.

CODE:

```
#include<stdio.h>

#include<stdlib.h>

#include<process.h>

#define que_size 3

int item,front=0,rear=-1,q[que_size],count=0;

void insertrear()

{

        if(count==que_size)

        {

                printf("queue overflow");

                return;

        }

        rear=(rear+1)%que_size;

        q[rear]=item;

        count++;

}

int deletefront()

{

        if(count==0) return -1;

        item = q[front];

        front=(front+1)%que_size;

        count=count-1;

        return item;

}

void displayq()

{
        int i,f;

        if(count==0)

        {

                printf("queue is empty");

                return;

        }

        f=front;

        printf("contents of queue \n");

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

        {

                printf("%d\n",q[f]);

                f=(f+1)%que_size;

        }

}

int main()

{

        int choice;

        for(;;)

        {

                printf("\n1.Insert rear \n2.Delete front \n3.Display \n4.exit \n ");

                printf("Enter the choice : ");

                scanf("%d",&choice);

                switch(choice)

                {

                        case 1:printf("Enter the item to be inserted :");

                            scanf("%d",&item);

                            insertrear();

                            break;

                        case 2:item=deletefront();
```

```
                                        if(item==-1)

                                        printf("queue is empty\n");

                                        else

                                        printf("item deleted is %d \n",item);

                                        break;

                        case 3:displayq();

                                        break;

                        default:exit(0);

                      }

              }


      }
```



F:\hp\Documents\qcircular.exe

```

1.Insert rear
2.Delete front
3.Display
4.exit
 Enter the choice : 1
Enter the item to be inserted :6

1.Insert rear
2.Delete front
3.Display
4.exit
 Enter the choice : 1
Enter the item to be inserted :5

1.Insert rear
2.Delete front
3.Display
4.exit
 Enter the choice : 2
item deleted is 6

1.Insert rear
2.Delete front
3.Display
4.exit
 Enter the choice : 3
contents of queue
5
0
```

1BM19ET004 LAB- 5 , LAB – 6 AND LAB-7 AKASH SHRIVASTAVA(3C)

Q. LAB 5 -> CREATION OF LINKED LIST , INSERTION(FRONT , REAR ,ANY POSITION) , DISPLAY.

LAB-6 -> CREATION LINKED LIST , DELETION(FRONT , REAR ,ANY POSITION) , DISPLAY.

LAB-7-> OPERATIONS ON LINKED LIST→(SORTING , REVERSE , CONCATINATION).

CODE(COMBINED):

```
#include<stdio.h>
#include <stdlib.h>
struct node
{
  int info;
  struct node *link;
};
typedef struct node *NODE;
NODE getnode()
{
NODE x;
x=(NODE)malloc(sizeof(struct node));
if(x==NULL)
 {
  printf("mem full\n");
  exit(0);
 }
 return x;
}
void freenode(NODE x)
{
free(x);
}
NODE insert_front(NODE first,int item)
```

```
{
NODE temp;
temp=getnode();
temp->info=item;
temp->link=NULL;
if(first==NULL)
return temp;
temp->link=first;
first=temp;
return first;
}
NODE delete_front(NODE first)
{
NODE temp;
if(first==NULL)
{
printf("list is empty cannot delete\n");
return first;
}
temp=first;
temp=temp->link;
printf("item deleted at front-end is=%d\n",first->info);
free(first);
return temp;
}
NODE insert_rear(NODE first,int item)
{
NODE temp,cur;
temp=getnode();
```

```
temp->info=item;

temp->link=NULL;

if(first==NULL)

 return temp;

cur=first;

while(cur->link!=NULL)

 cur=cur->link;

cur->link=temp;

return first;

}

NODE delete_rear(NODE first)

{

NODE cur,prev;

if(first==NULL)

{

printf("list is empty cannot delete\n");

return first;

}

if(first->link==NULL)

{

printf("item deleted is %d\n",first->info);

free(first);

return NULL;

}

prev=NULL;

cur=first;

while(cur->link!=NULL)

{

prev=cur;
cur=cur->link;

}

printf("iten deleted at rear-end is %d",cur->info);

free(cur);

prev->link=NULL;

return first;

}

NODE insert_pos(int item,int pos,NODE first)

{

NODE temp;

NODE prev,cur;

int count;

temp=getnode();

temp->info=item;

temp->link=NULL;

if(first==NULL && pos==1)

return temp;

if(first==NULL)

{

 printf("invalid pos\n");

 return first;

}

if(pos==1)

{

temp->link=first;

return temp;

}

count=1;

prev=NULL;
```

```
cur=first;
while(cur!=NULL && count!=pos)
{
 prev=cur;
 cur=cur->link;
 count++;
}
if(count==pos)
{
prev->link=temp;
temp->link=cur;
return first;
}
printf("IP\n");
return first;
}
NODE delete_pos(int pos, NODE first){
    if (first == NULL){
      printf("List empty\n");
      return first;
    }

   NODE temp= first;

    if (pos==1)
    {
        first = temp->link;
        free(temp);
        return first;
    }

    NODE prev;


    for (int i=1; temp!=NULL && i<pos; i++){

        prev=temp;

        temp = temp->link;

    }


    if (temp == NULL || temp->link == NULL){

            printf("Invalid position\n");

            return NULL;

    }

    prev->link=temp->link;

    printf("Element deleted %d\n",temp->info);

    free(temp);

    return first;

}

void display(NODE first)

{

 NODE temp;

 if(first==NULL)

 printf("list empty cannot display items\n");

 for(temp=first;temp!=NULL;temp=temp->link)

  {

  printf("%d\n",temp->info);

  }

}


NODE concat(NODE first,NODE second)
```

```
{
 NODE cur;
 if(first==NULL)
  return second;
 if(second==NULL)
  return first;
 cur=first;
 while(cur->link!=NULL)
  cur=cur->link;
 cur->link=second;
 return first;
}


NODE reverse(NODE first)
 {
 NODE cur,temp;
 cur=NULL;
 while(first!=NULL)
  {
   temp=first;
   first=first->link;
   temp->link=cur;
   cur=temp;
  }
 return cur;
}

NODE order_list(NODE first)
```

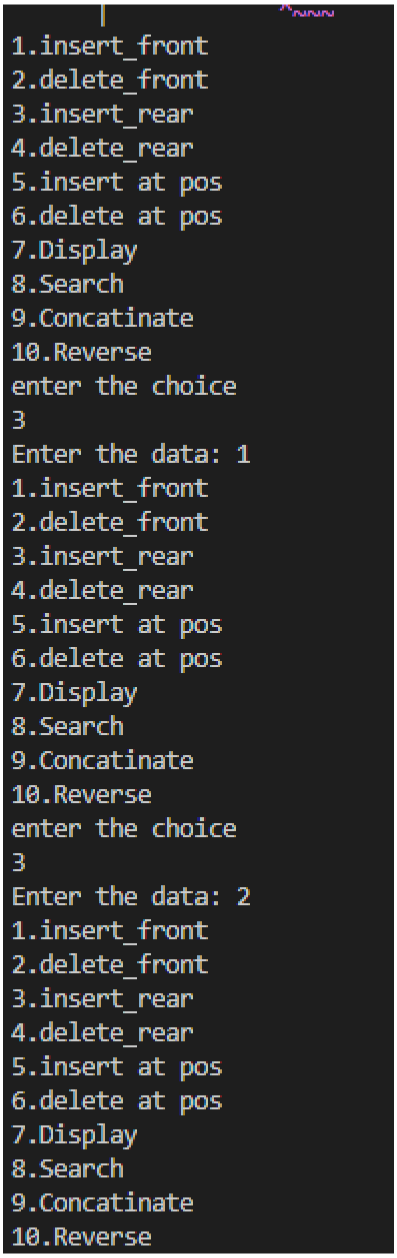
```
{
  int swapped, i;
  NODE ptr1,lptr=NULL;

   if (first == NULL)
   return first;

  do
  {
    swapped = 0;
    ptr1 = first;

    while (ptr1->link != lptr)
    {
      if (ptr1->info > ptr1->link->info)
      {
       int temp = ptr1->info;
       ptr1->info = ptr1->link->info;
       ptr1->link->info = temp;
       swapped = 1;
      }
      ptr1 = ptr1->link;
    }
    lptr = ptr1;
  }
  while (swapped);
  return first;
}
void main()
```

```
{

int item,choice,pos,i,n;

NODE a,b;

NODE first=NULL;


for(;;)

{

printf("1.insert_front\n2.delete_front\n3.insert_rear\n4.delete_rear\n5.insert at pos\n6.delete at
pos\n7.concat\n8.reverse\n9.order list\n10.display\n");

printf("enter the choice\n");

scanf("%d",&choice);

switch(choice)

 {

  case 1:printf("enter the item at front-end\n");

               scanf("%d",&item);

               first=insert_front(first,item);

               break;

  case 2:first=delete_front(first);

               break;

  case 3:printf("enter the item at rear-end\n");

               scanf("%d",&item);

               first=insert_rear(first,item);

               break;

  case 4:first=delete_rear(first);

               break;

  case 5:

  printf("Enter item\n");

  scanf("%d",&item);

  printf("enter the position\n");
```

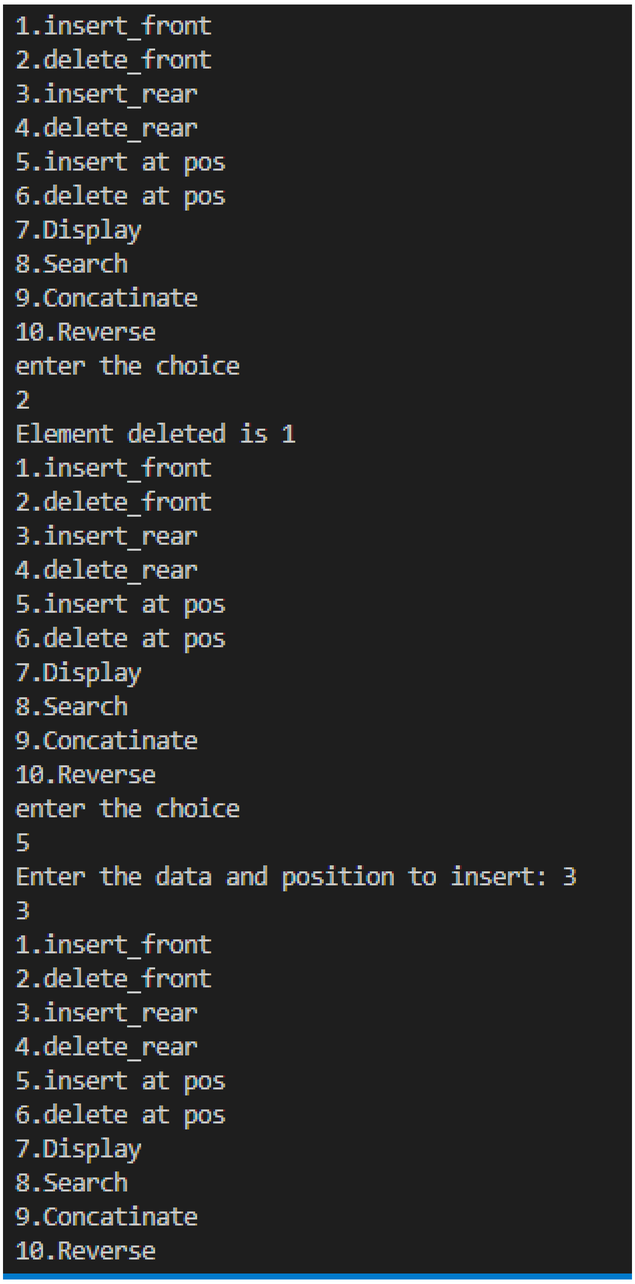
```
                         scanf("%d",&pos);

                         first=insert_pos(item,pos,first);

                         break;

case 6:

printf("Enter posititon of deletion\n");

scanf("%d",&pos);

first=delete_pos(pos,first);

break;

case 7:

printf("enter the no of nodes in 1\n");

                         scanf("%d",&n);

                         a=NULL;

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

                          {

                           printf("enter the item\n");

                           scanf("%d",&item);

                           a=insert_rear(a,item);

                          }

                          printf("enter the no of nodes in 2\n");

                         scanf("%d",&n);

                         b=NULL;

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

                          {

                           printf("enter the item\n");

                           scanf("%d",&item);

                           b=insert_rear(b,item);

                          }

                          a=concat(a,b);

                          display(a);
```

```
                           break;
 case 8:
 first=reverse(first);
                           display(first);
                           break;
 case 9:
 first=order_list(first);
 break;
 case 10:display(first);
              break;
 default:exit(0);
              break;
 }
}
}
```

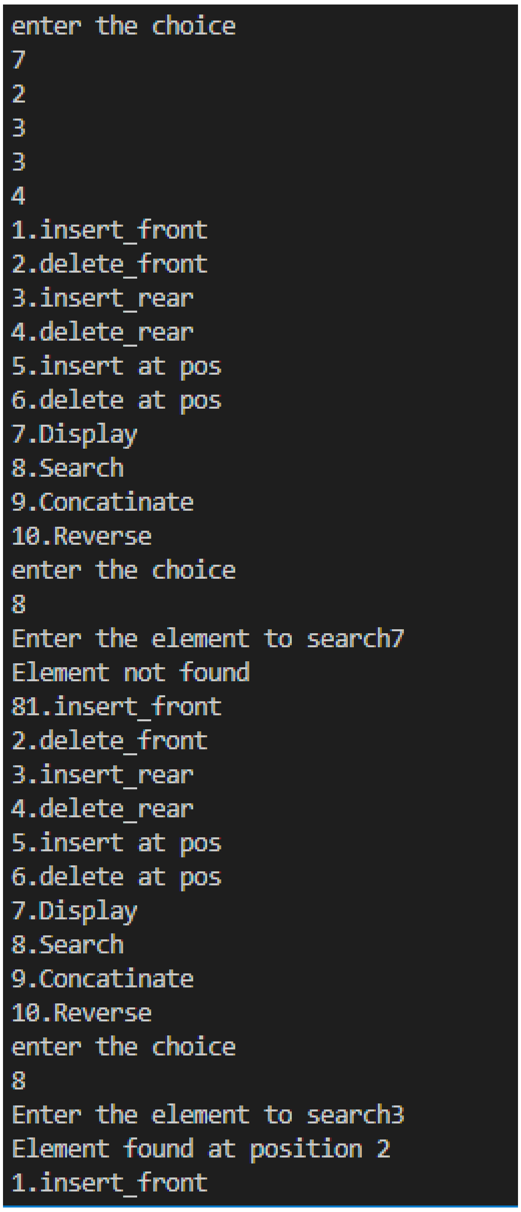
Output:



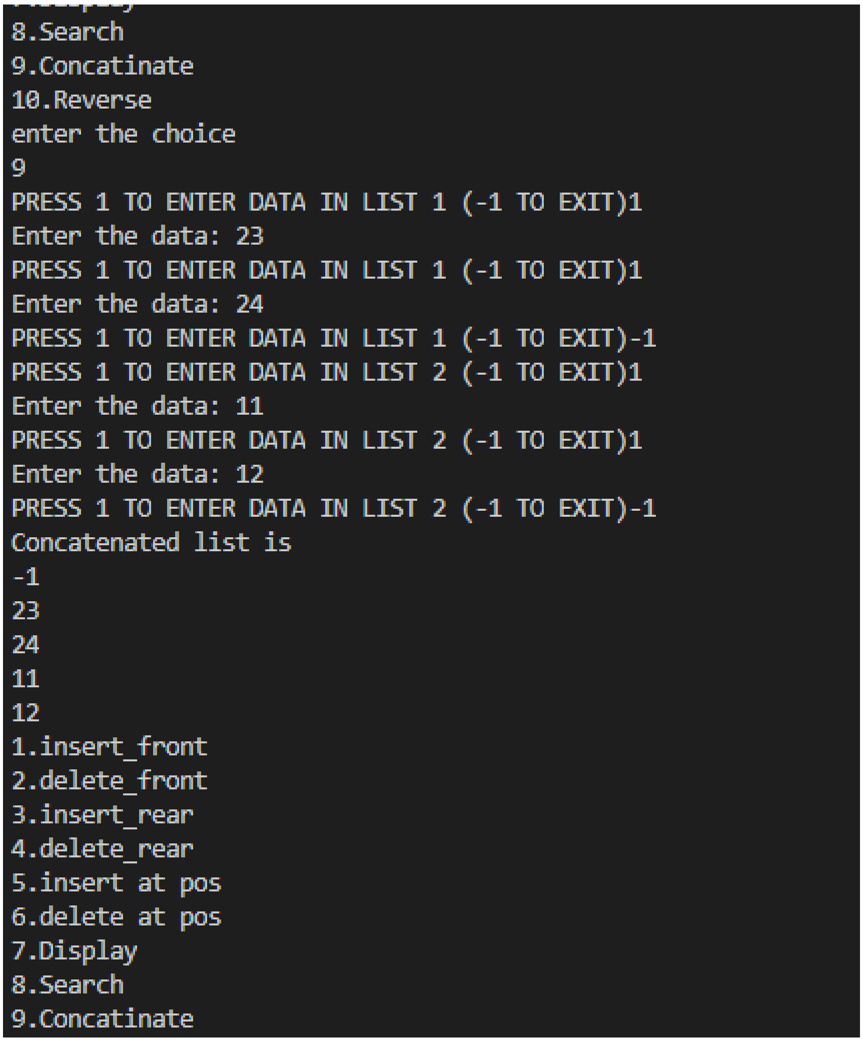
```
1.insert_front
2.delete_front
3.insert_rear
4.delete_rear
5.insert at pos
6.delete at pos
7.Display
8.Search
9.Concatinate
10.Reverse
enter the choice
3
Enter the data: 1
1.insert_front
2.delete_front
3.insert_rear
4.delete_rear
5.insert at pos
6.delete at pos
7.Display
8.Search
9.Concatinate
10.Reverse
enter the choice
3
Enter the data: 2
1.insert_front
2.delete_front
3.insert_rear
4.delete_rear
5.insert at pos
6.delete at pos
7.Display
8.Search
9.Concatinate
10.Reverse
```



```
1.insert_front
2.delete_front
3.insert_rear
4.delete_rear
5.insert at pos
6.delete at pos
7.Display
8.Search
9.Concatinate
10.Reverse
enter the choice
2
Element deleted is 1
1.insert_front
2.delete_front
3.insert_rear
4.delete_rear
5.insert at pos
6.delete at pos
7.Display
8.Search
9.Concatinate
10.Reverse
enter the choice
5
Enter the data and position to insert: 3
3
1.insert_front
2.delete_front
3.insert_rear
4.delete_rear
5.insert at pos
6.delete at pos
7.Display
8.Search
9.Concatinate
10.Reverse
```



```
enter the choice
7
2
3
3
4
1.insert_front
2.delete_front
3.insert_rear
4.delete_rear
5.insert at pos
6.delete at pos
7.Display
8.Search
9.Concatinate
10.Reverse
enter the choice
8
Enter the element to search7
Element not found
81.insert_front
2.delete_front
3.insert_rear
4.delete_rear
5.insert at pos
6.delete at pos
7.Display
8.Search
9.Concatinate
10.Reverse
enter the choice
8
Enter the element to search3
Element found at position 2
1.insert_front
```



```
8.Search
9.Concatinate
10.Reverse
enter the choice
9
PRESS 1 TO ENTER DATA IN LIST 1 (-1 TO EXIT)1
Enter the data: 23
PRESS 1 TO ENTER DATA IN LIST 1 (-1 TO EXIT)1
Enter the data: 24
PRESS 1 TO ENTER DATA IN LIST 1 (-1 TO EXIT)-1
PRESS 1 TO ENTER DATA IN LIST 2 (-1 TO EXIT)1
Enter the data: 11
PRESS 1 TO ENTER DATA IN LIST 2 (-1 TO EXIT)1
Enter the data: 12
PRESS 1 TO ENTER DATA IN LIST 2 (-1 TO EXIT)-1
Concatenated list is
-1
23
24
11
12
1.insert_front
2.delete_front
3.insert_rear
4.delete_rear
5.insert at pos
6.delete at pos
7.Display
8.Search
9.Concatinate
```

1BM19ET004 LAB-8 AKASH SHRIVASTAVA(3C)

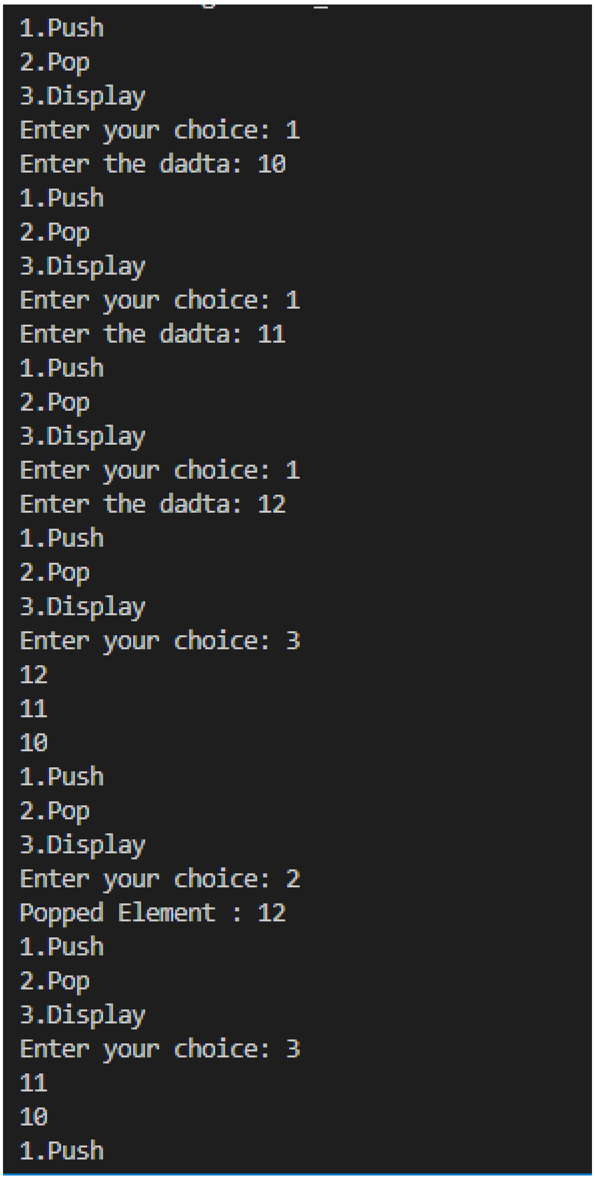
Q. STACK AND QUEUE IMPLEMENTATION USING LINKED LIST.

CODE:

```
#include<stdio.h>

#include<stdlib.h>



struct node

{

   int data;

   struct node* next;

};

typedef struct node* Node;
```

```
Node create()
{
   Node newnode;
   newnode=(Node)malloc(sizeof(struct node));
   return newnode;
}


Node push(Node top)
{
   Node temp=create();
   int x;
   printf("Enter the dadta: ");
   scanf("%d",&x);
   temp->data=x;
   temp->next=top;
   top=temp;
   return top;
}

Node pop(Node top)
{
   Node temp;
   temp=top;
   if(top==NULL)
   printf("Underflow\n");
   else
   {
      printf("Popped Element : %d\n",top->data);
```

```
        top=top->next;
    }

}

void display(Node top)
{
    Node temp;
    temp=top;
    if(top==NULL)
    printf("Empty\n");
    while(temp!=NULL)
    {

        printf("%d\n",temp->data);
        temp=temp->next;
    }

}


int main()
{
    Node top=NULL;
    int choice;
    for(;;)
    {
    printf("1.Push\n2.Pop\n3.Display\n");
    printf("Enter your choice: ");
    scanf("%d",&choice);

    switch (choice)

    {

    case 1: top=push(top);

        break;

    case 2: top=pop(top);

        break;

    case 3: display(top);

    default:

        break;

    }

    }

}
```



```
1.Push
2.Pop
3.Display
Enter your choice: 1
Enter the dadta: 10
1.Push
2.Pop
3.Display
Enter your choice: 1
Enter the dadta: 11
1.Push
2.Pop
3.Display
Enter your choice: 1
Enter the dadta: 12
1.Push
2.Pop
3.Display
Enter your choice: 3
12
11
10
1.Push
2.Pop
3.Display
Enter your choice: 2
Popped Element : 12
1.Push
2.Pop
3.Display
Enter your choice: 3
11
10
1.Push
```

QUEUE IMPLEMENTATION:

CODE:

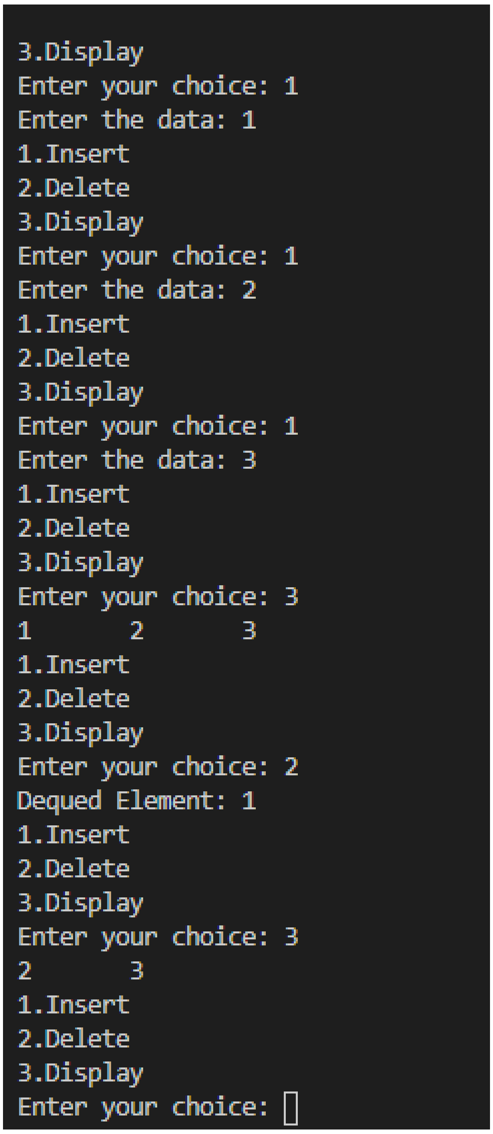
```
#include<stdio.h>

#include<stdlib.h>

struct node

{

   int data;

   struct node* next;

};

typedef struct node* Node;
```

```
Node create()
{
    Node newnode;
    newnode=(Node)malloc(sizeof(struct node));
    return newnode;
}
Node front=NULL;
Node rear=NULL;
void enq()
{
    Node temp=create();
    int x;
    printf("Enter the data: ");
    scanf("%d",&x);
    temp->data=x;
    temp->next=NULL;
    if(front == NULL && rear==NULL)
    {
    front = rear = temp;
    }
    else
    {
        rear->next=temp;
        rear=temp;
    }

}
```

```
void dq()
{
    Node temp;
    temp=front;
    if(front==NULL && rear==NULL)
    {
    printf("Underflow\n");
    return NULL;
    }

    else
    {
        {
            front=front->next;
            printf("Dequed Element: %d\n",temp->data);
            free(temp);
        }
    }
    return front;
}


void display()
{
    Node temp;
    if(front==NULL && rear==NULL)
    printf("Empty\n");
    else
    {
        temp=front;
        while(temp!=NULL)
        {
            printf("%d\t",temp->data);
            temp=temp->next;
        }
    }
    printf("\n");
}



int main()
{
    int choice;
    Node front=NULL;
    Node rear=NULL;
    for(;;)
    {
    printf("1.Insert\n2.Delete\n3.Display\n");
    printf("Enter your choice: ");
    scanf("%d",&choice);
    switch (choice)
    {
    case 1:enq();
        break;
    case 2:dq();
        break;
    case 3:display();
```

```
   default:

      break;

   }

   }

}
```

OUTPUT:



```
3.Display
Enter your choice: 1
Enter the data: 1
1.Insert
2.Delete
3.Display
Enter your choice: 1
Enter the data: 2
1.Insert
2.Delete
3.Display
Enter your choice: 1
Enter the data: 3
1.Insert
2.Delete
3.Display
Enter your choice: 3
1       2       3
1.Insert
2.Delete
3.Display
Enter your choice: 2
Dequed Element: 1
1.Insert
2.Delete
3.Display
Enter your choice: 3
2       3
1.Insert
2.Delete
3.Display
Enter your choice: 
```

1BM19ET004 LAB- 9 AKASH SHRIVASTAVA(3C)

Q. WAP Implement doubly link list with following operations: Create a doubly linked list.

b) Insert front and rear.

c) Delete the node both front and rear d) Display the contents of the list

e) Simple search f) Inserting the node before and after the key node g)Deleting all Occurrences.

CODE:

```
#include<stdio.h>
#include<conio.h>
#include<process.h>
#include<stdlib.h>
struct node
{
         int info;
         struct node *llink;
         struct node *rlink;
         };
typedef struct node *NODE;
NODE getnode()
{
        NODE x;
        x=(NODE)malloc(sizeof(struct node));
        if(x==NULL)
        {
                printf("mem full\n");
                exit(0);
                }
        return x;
        }
```

```
void freenode(NODE x)
{
        free(x);
}
NODE dinsert_front(int item,NODE head)
{
NODE temp,cur;
temp=getnode();
temp->info=item;
cur=head->rlink;
head->rlink=temp;
temp->llink=head;
temp->rlink=cur;
cur->llink=temp;
return head;
}
NODE dinsert_rear(int item,NODE head)
{
NODE temp,cur;
temp=getnode();
temp->info=item;
cur=head->llink;
head->llink=temp;
temp->rlink=head;
temp->llink=cur;
cur->rlink=temp;
return head;
}
NODE ddelete_front(NODE head)
```

```
{

NODE cur,next;

if(head->rlink==head)

{

printf("dq empty\n");

return head;

}

cur=head->rlink;

next=cur->rlink;

head->rlink=next;

next->llink=head;

printf("the node deleted is %d",cur->info);

freenode(cur);

return head;

}

NODE ddelete_rear(NODE head)

{

NODE cur,prev;

if(head->rlink==head)

{

printf("dq empty\n");

return head;

}

cur=head->llink;

prev=cur->llink;

head->llink=prev;

prev->rlink=head;

printf("the node deleted is %d",cur->info);

freenode(cur);
```

```
return head;

}

void display(NODE head)

{

NODE temp;

if(head->rlink==head)

{

printf("dq empty\n");

return;

}

printf("contents of List: \n");

temp=head->rlink;

while(temp!=head)

{

printf("%d  ",temp->info);

temp=temp->rlink;

}

printf("\n");

}

void search(NODE head)

{

        bool flag=false;

        NODE x=head->rlink->llink;//head's address

        NODE temp=head->rlink;

        int item,count=1;

        printf("Enter the element to be searched in the list: ");

        scanf("%d",&item);

        while(temp!=x)

        {
                if(temp->info==item)

                {

                        flag=true;

                        break;

                }

                temp=temp->rlink;

                count++;

        }

        if(flag)

        printf("Element is found in the list at position %d",count);

        else if(temp==x)

        printf("Element not found\n");

}

NODE insert_leftpos(NODE head)

{

NODE temp,cur,prev;

int item;

printf("Enter the item");

scanf("%d",&item);

if(head->rlink==head)

{

printf("list empty\n");

return head;

}

cur=head->rlink;

while(cur!=head)

{

if(item==cur->info)break;

cur=cur->rlink;
```

```
}
if(cur==head)
{
 printf("key not found\n");
 return head;
 }
 prev=cur->llink;
 printf("enter towards left of %d=",item);
 temp=getnode();
 scanf("%d",&temp->info);
 prev->rlink=temp;
 temp->llink=prev;
 cur->llink=temp;
 temp->rlink=cur;
 return head;
}

NODE insert_rightpos(NODE head)
{
		NODE temp,cur,next;
int item;
printf("Enter the item");
scanf("%d",&item);
if(head->rlink==head)
{
printf("list is empty\n");
return head;
}
cur=head->rlink;
```

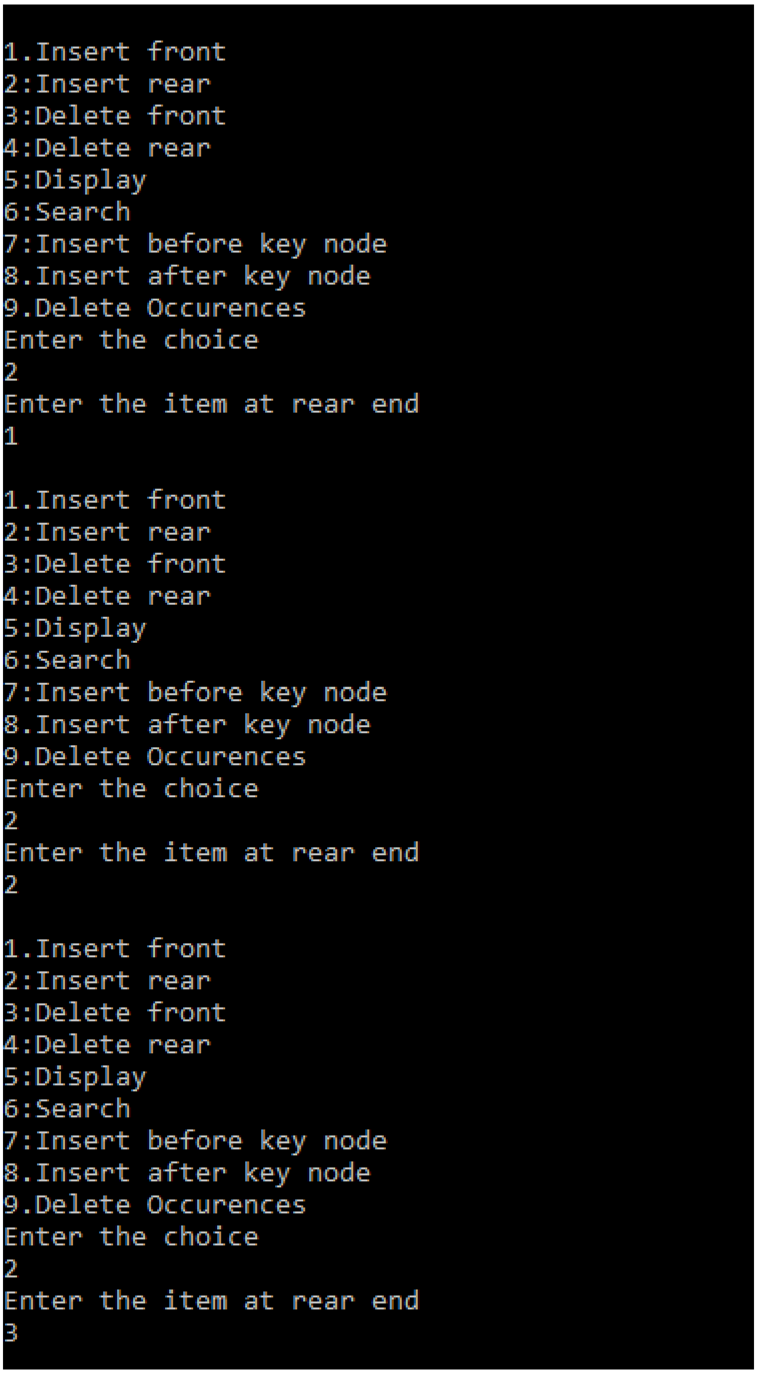
```
while(cur!=head)

{

if(item==cur->info)break;

cur=cur->rlink;

}

if(cur==head)

{

 printf("Key not found\n");

 return head;

 }

 next=cur->rlink;

 printf("enter item towards right of %d=",item);

 temp=getnode();

 scanf("%d",&temp->info);

 next->llink=temp;

 temp->rlink=next;

 cur->rlink=temp;

 temp->llink=cur;

 return head;


}




NODE delete_all_key(NODE head)

{

NODE prev,cur,next;

int item;

printf("Enter the item: ");

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

```
int count;

   if(head->rlink==head)

    {

     printf("Linked list empty\n");

     return head;

     }

count=0;

cur=head->rlink;

while(cur!=head)

{

  if(item!=cur->info)

  cur=cur->rlink;

  else

 {

  count++;

  prev=cur->llink;

  next=cur->rlink;

  prev->rlink=next;

  next->llink=prev;

  freenode(cur);

  cur=next;

 }

}

if(count==0)

  printf("key not found\n");

  else

 printf("key found at %d positions and are deleted\n", count);


return head;
```

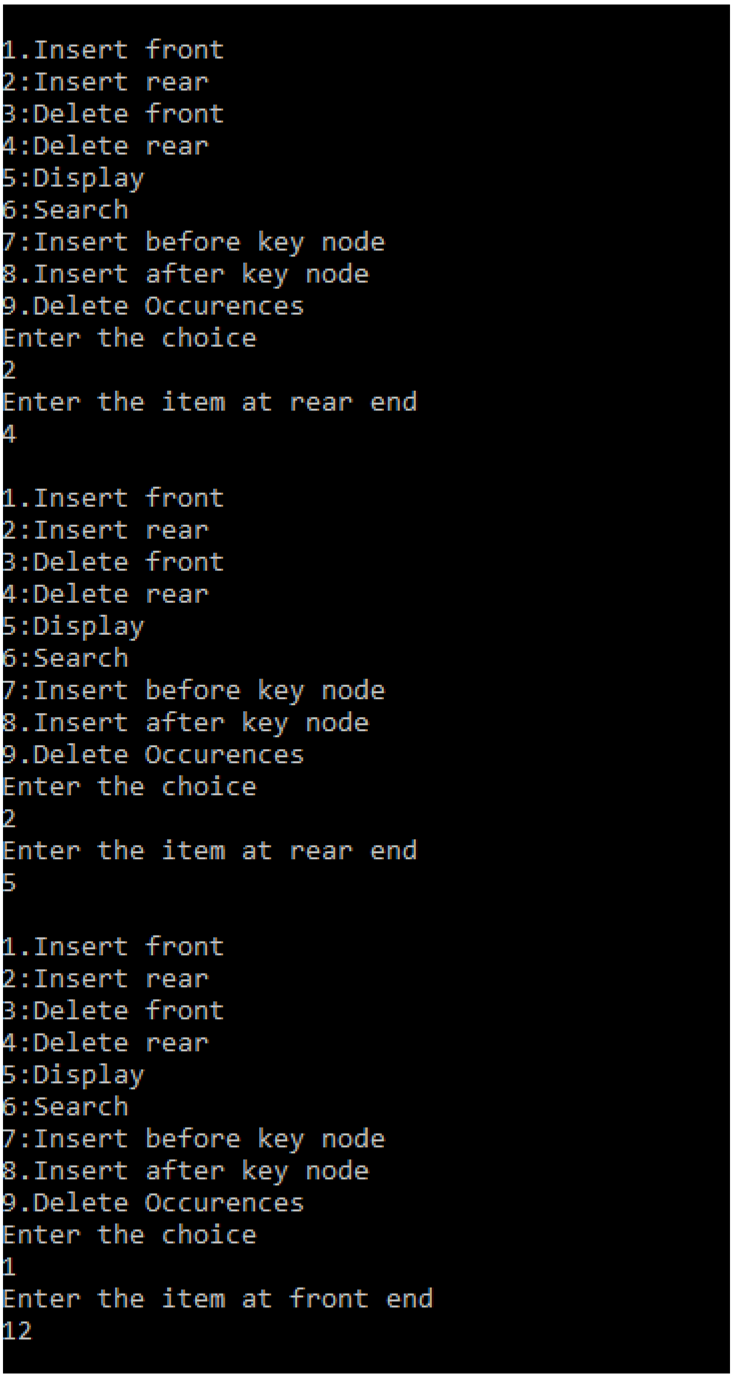
```
}

int main()

{

NODE head,last,y;

int item, choice;

head=getnode();

head->rlink=head;

head->llink=head;

for(;;)

{

        printf("\n1.Insert front\n2:Insert rear\n3:Delete front\n4:Delete rear\n5:Display\n6:Search\n7:Insert before
key node\n8.Insert after key node\n9.Delete Occurences\n");

        printf("Enter the choice\n");

        scanf("%d",&choice);

        switch(choice)

        {

                case 1: printf("Enter the item at front end\n");

                        scanf("%d",&item);

                        last=dinsert_front(item,head);

                        break;

                case 2: printf("Enter the item at rear end\n");

                        scanf("%d",&item);

                        last=dinsert_rear(item,head);

                        break;

                case 3:last=ddelete_front(head);

                        break;

                case 4: last=ddelete_rear(head);

                        break;

                case 5: display(head);
```

```
                    break;

              case 6: search(head);

                 break;

              case 7:y=insert_leftpos(head);

                 break;

              case 8: y=insert_rightpos(head);

                   break;

              case 9:delete_all_key(head);

                         break;

              default:exit(0);

              }

       }

}
```

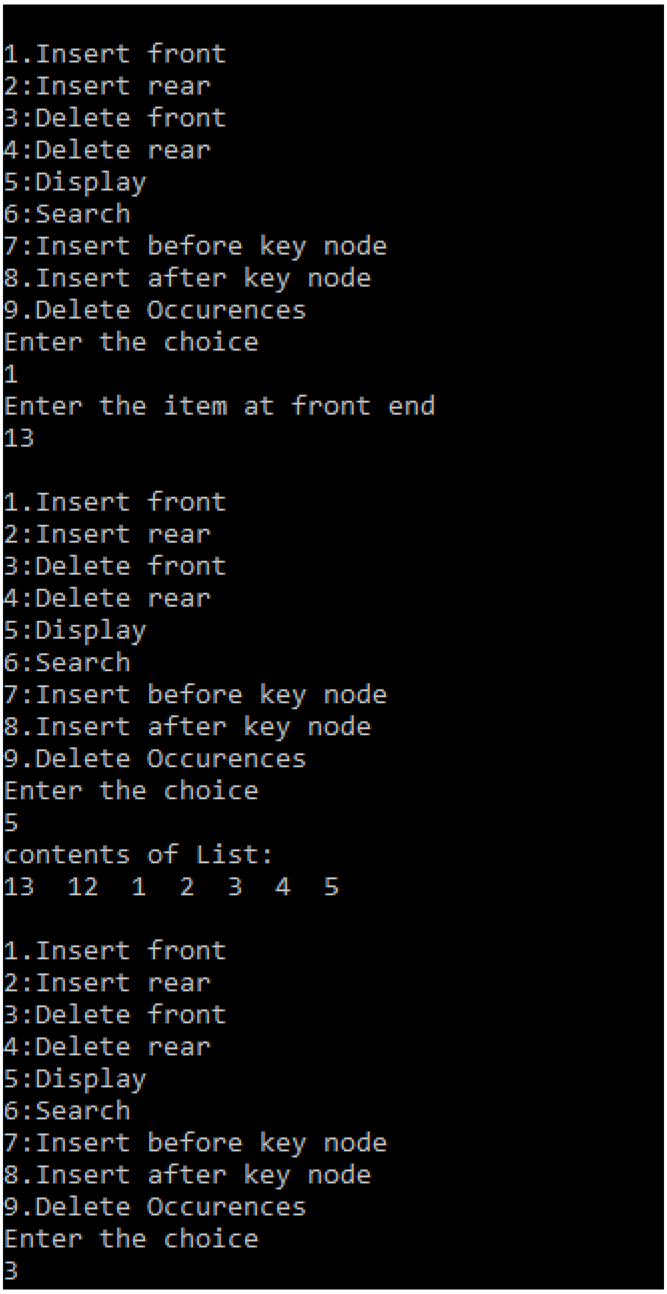
OUTPUT:



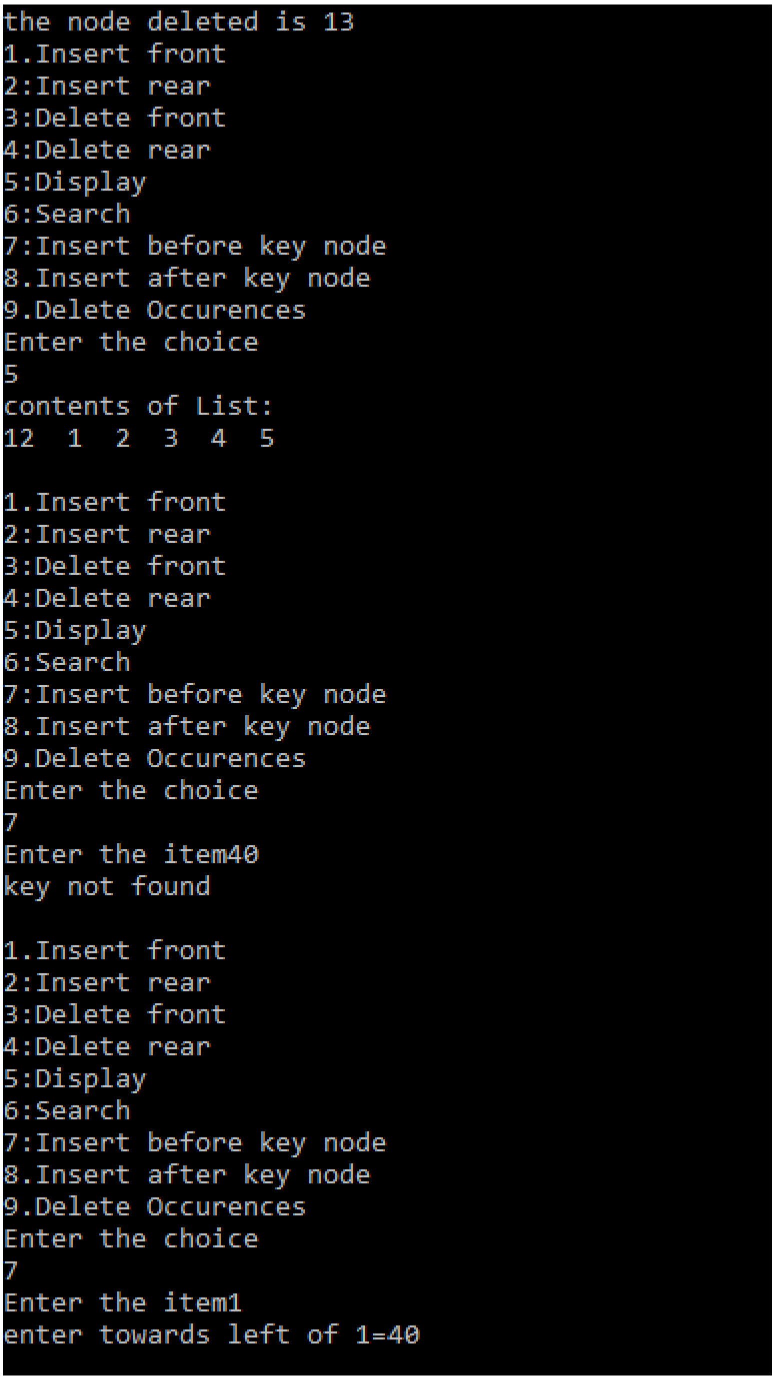
```
1.Insert front
2:Insert rear
3:Delete front
4:Delete rear
5:Display
6:Search
7:Insert before key node
8.Insert after key node
9.Delete Occurences
Enter the choice
2
Enter the item at rear end
1

1.Insert front
2:Insert rear
3:Delete front
4:Delete rear
5:Display
6:Search
7:Insert before key node
8.Insert after key node
9.Delete Occurences
Enter the choice
2
Enter the item at rear end
2

1.Insert front
2:Insert rear
3:Delete front
4:Delete rear
5:Display
6:Search
7:Insert before key node
8.Insert after key node
9.Delete Occurences
Enter the choice
2
Enter the item at rear end
3
```



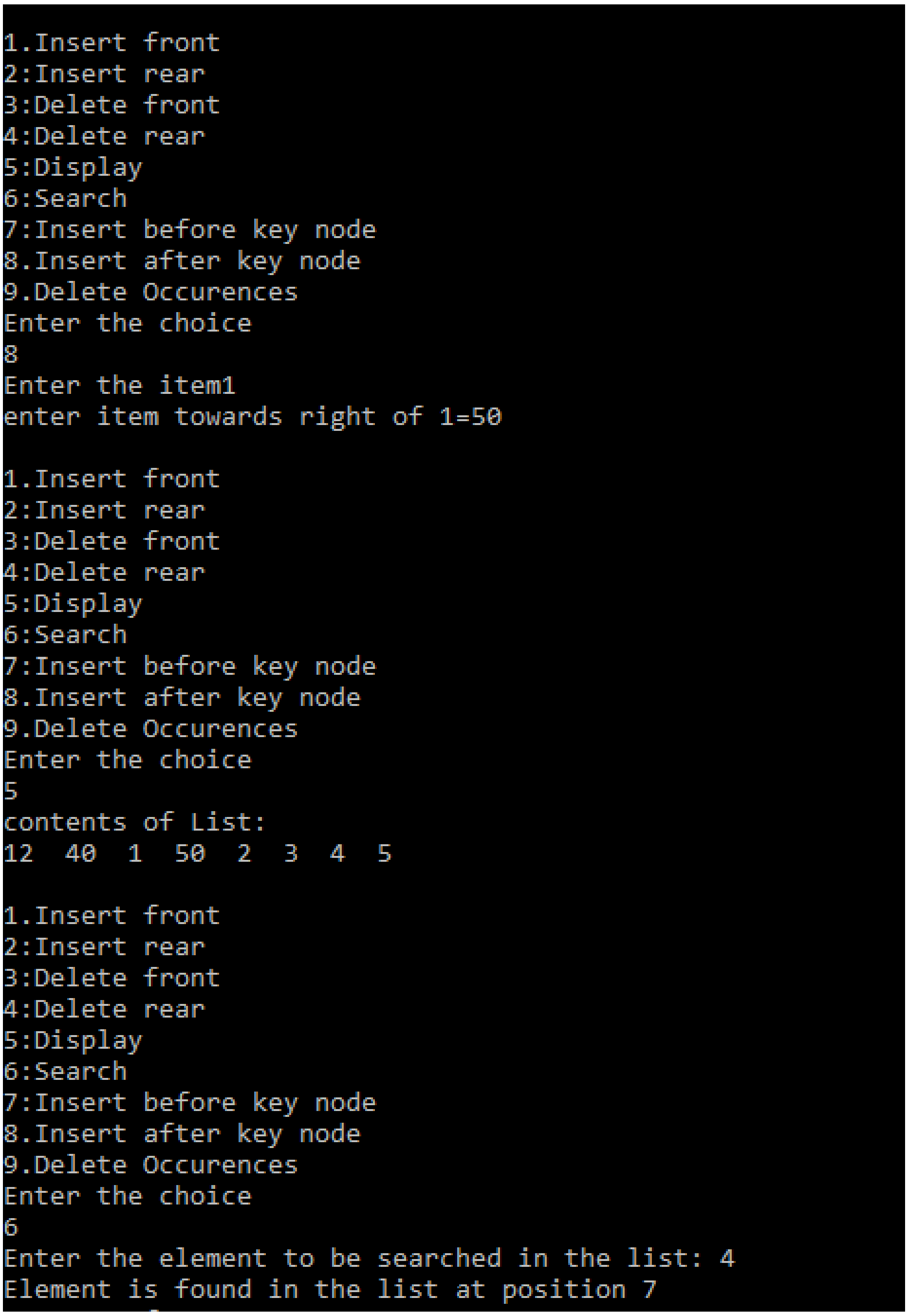
```
1.Insert front
2:Insert rear
3:Delete front
4:Delete rear
5:Display
6:Search
7:Insert before key node
8.Insert after key node
9.Delete Occurences
Enter the choice
2
Enter the item at rear end
4

1.Insert front
2:Insert rear
3:Delete front
4:Delete rear
5:Display
6:Search
7:Insert before key node
8.Insert after key node
9.Delete Occurences
Enter the choice
2
Enter the item at rear end
5

1.Insert front
2:Insert rear
3:Delete front
4:Delete rear
5:Display
6:Search
7:Insert before key node
8.Insert after key node
9.Delete Occurences
Enter the choice
1
Enter the item at front end
12
```



```
1.Insert front
2:Insert rear
3:Delete front
4:Delete rear
5:Display
6:Search
7:Insert before key node
8.Insert after key node
9.Delete Occurences
Enter the choice
1
Enter the item at front end
13

1.Insert front
2:Insert rear
3:Delete front
4:Delete rear
5:Display
6:Search
7:Insert before key node
8.Insert after key node
9.Delete Occurences
Enter the choice
5
contents of List:
13  12  1  2  3  4  5

1.Insert front
2:Insert rear
3:Delete front
4:Delete rear
5:Display
6:Search
7:Insert before key node
8.Insert after key node
9.Delete Occurences
Enter the choice
3
```



```
the node deleted is 13
1.Insert front
2:Insert rear
3:Delete front
4:Delete rear
5:Display
6:Search
7:Insert before key node
8.Insert after key node
9.Delete Occurences
Enter the choice
5
contents of List:
12  1  2  3  4  5

1.Insert front
2:Insert rear
3:Delete front
4:Delete rear
5:Display
6:Search
7:Insert before key node
8.Insert after key node
9.Delete Occurences
Enter the choice
7
Enter the item40
key not found

1.Insert front
2:Insert rear
3:Delete front
4:Delete rear
5:Display
6:Search
7:Insert before key node
8.Insert after key node
9.Delete Occurences
Enter the choice
7
Enter the item1
enter towards left of 1=40
```



```
1.Insert front
2:Insert rear
3:Delete front
4:Delete rear
5:Display
6:Search
7:Insert before key node
8.Insert after key node
9.Delete Occurences
Enter the choice
8
Enter the item1
enter item towards right of 1=50

1.Insert front
2:Insert rear
3:Delete front
4:Delete rear
5:Display
6:Search
7:Insert before key node
8.Insert after key node
9.Delete Occurences
Enter the choice
5
contents of List:
12  40  1  50  2  3  4  5

1.Insert front
2:Insert rear
3:Delete front
4:Delete rear
5:Display
6:Search
7:Insert before key node
8.Insert after key node
9.Delete Occurences
Enter the choice
6
Enter the element to be searched in the list: 4
Element is found in the list at position 7
```

1BM19ET004 LAB- 10 AKASH SHRIVASTAVA(3C)

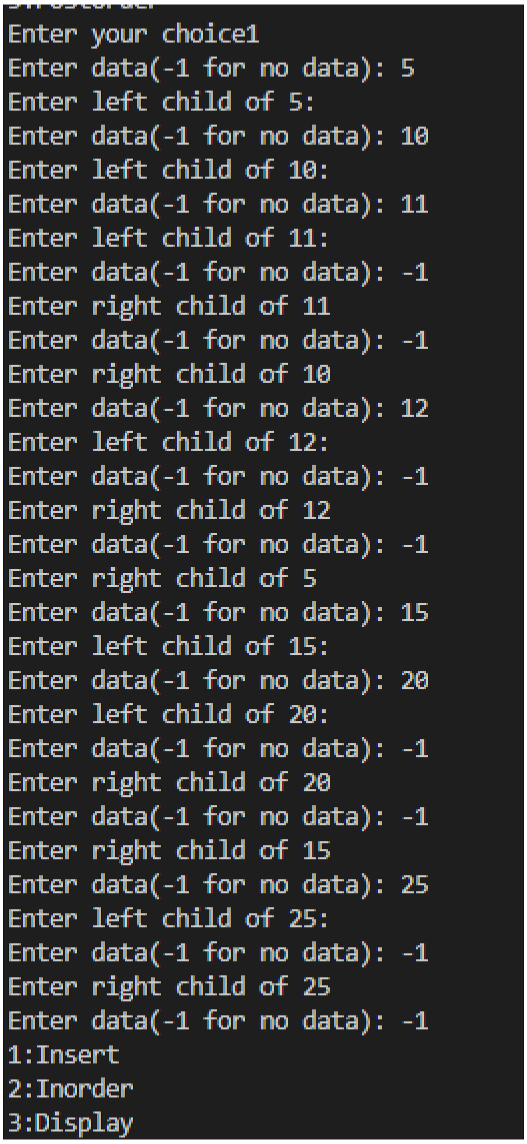
Q. BINARY TREE AND BST.

```
#include<stdio.h>

#include<stdlib.h>

struct Node
```

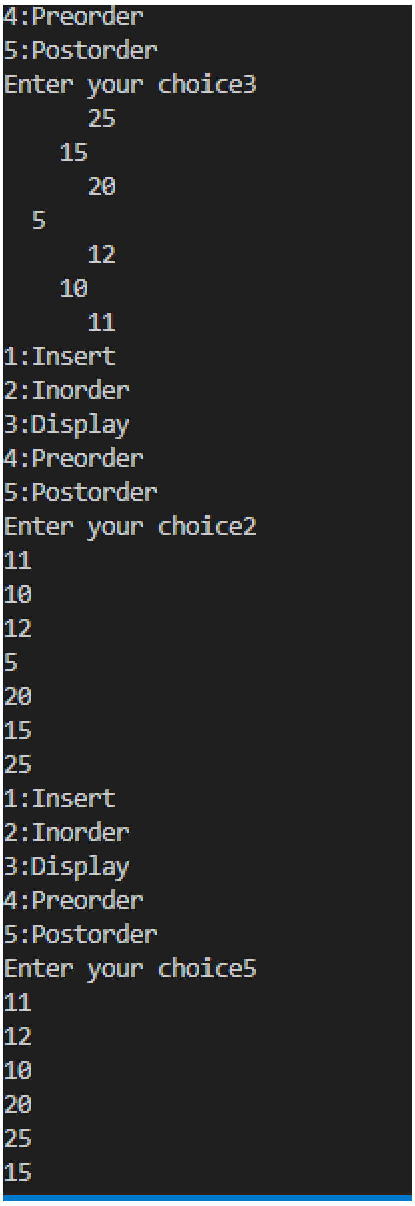
```
{
        struct Node *llink;
        int data;
        struct Node *rlink;
};
typedef struct Node* NODE;

NODE create()
{
        NODE newnode;
        int x;
        newnode=(NODE)malloc(sizeof(struct Node));
        printf("Enter data(-1 for no data): ");
        scanf("%d",&x);
        if(x==-1)
        return 0;
        newnode->data=x;
        printf("Enter left child of %d: \n",x);
        newnode->llink=create();
        printf("Enter right child of %d \n",x);
        newnode->rlink=create();
        return newnode;
}

void inorder(NODE head)
{
        if(head!=0)
        {
                inorder(head->llink);
                printf("%d\t\n",head->data);
                inorder(head->rlink);
        }
}

void preorder(NODE head)
{
        if(head!=0)
        {
                printf("%d\t\n",head->data);
                preorder(head->llink);
                preorder(head->rlink);
        }
}

void postorder(NODE head)
{
        if(head!=0)
        {
                postorder(head->llink);
                postorder(head->rlink);
                printf("%d\t\n",head->data);
        }
}
void display(NODE head,int i)
{
int j;
if(head!=NULL)
{
```

```
display(head->rlink,i+1);

for (j=1;j<=i;j++)

printf("  ");

printf("%d\n",head->data);

display(head->llink,i+1);

}

}


int main()

{

        NODE head=0;

        int ch;

        for(;;)

        {

        printf("1:Insert\n2:Inorder\n3:Display\n4:Preorder\n5:Postorder\n");

        printf("Enter your choice");

        scanf("%d",&ch);

        switch(ch)

        {

                case 1:head=create();

                        break;

                case 2:

                        inorder(head);

                        break;

                case 3:

                   display(head,1);

                         break;

                case 4:

                        preorder(head);
```

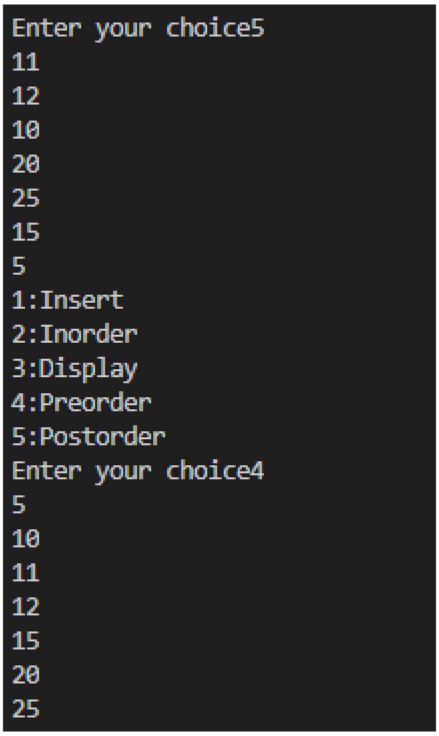
```
                    break;

               case 5:

                 postorder(head);

                    break;

          }

   }

}
```



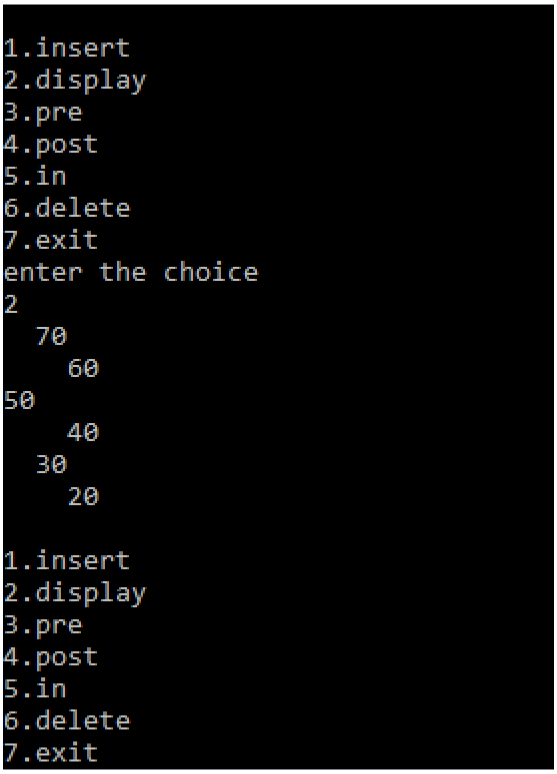
```
Enter your choice1
Enter data(-1 for no data): 5
Enter left child of 5:
Enter data(-1 for no data): 10
Enter left child of 10:
Enter data(-1 for no data): 11
Enter left child of 11:
Enter data(-1 for no data): -1
Enter right child of 11
Enter data(-1 for no data): -1
Enter right child of 10
Enter data(-1 for no data): 12
Enter left child of 12:
Enter data(-1 for no data): -1
Enter right child of 12
Enter data(-1 for no data): -1
Enter right child of 5
Enter data(-1 for no data): 15
Enter left child of 15:
Enter data(-1 for no data): 20
Enter left child of 20:
Enter data(-1 for no data): -1
Enter right child of 20
Enter data(-1 for no data): -1
Enter right child of 15
Enter data(-1 for no data): 25
Enter left child of 25:
Enter data(-1 for no data): -1
Enter right child of 25
Enter data(-1 for no data): -1
1:Insert
2:Inorder
3:Display
```



```
4:Preorder
5:Postorder
Enter your choice3
      25
    15
      20
  5
      12
    10
      11
1:Insert
2:Inorder
3:Display
4:Preorder
5:Postorder
Enter your choice2
11
10
12
5
20
15
25
1:Insert
2:Inorder
3:Display
4:Preorder
5:Postorder
Enter your choice5
11
12
10
20
25
15
```



```
Enter your choice5
11
12
10
20
25
15
5
1:Insert
2:Inorder
3:Display
4:Preorder
5:Postorder
Enter your choice4
5
10
11
12
15
20
25
```



```

1.insert
2.display
3.pre
4.post
5.in
6.delete
7.exit
enter the choice
2
  70
    60
50
    40
  30
    20

1.insert
2.display
3.pre
4.post
5.in
6.delete
7.exit
```

.