VISVESVARAYA TECHNOLOGICAL UNIVERSITY

"JnanaSangama", Belgaum -590014, Karnataka.

DATA STRUCTURE LAB RECORD

Submitted by

SAMARTH C SHETTY (1BM19CS141)

Under the Guidance of

Prof. Lohith JJ
Assistant Professor, BMSCE

in partial fulfillment for the award of the degree of
BACHELOR OF ENGINEERING
in
COMPUTER SCIENCE AND ENGINEERING

B.M.S. COLLEGE OF ENGINEERING
(Autonomous Institution under VTU)
BENGALURU-560019

Sep-2020 to Jan-2021

B. M. S. College of Engineering,

Bull Temple Road, Bangalore 560019

(Affiliated To Visvesvaraya Technological University, Belgaum)

Department of Computer Science and Engineering

CERTIFICATE

This is to certify that the LAB RECORD carried out by **SAMARTH C SHETTY (1BM19CS141)** who is the bonafide students of **B. M. S. College of Engineering.** It is in partial fulfillment for the award of **Bachelor of Engineering in Computer Science and Engineering** of the Visveswaraiah Technological University, Belgaum during the year 2020-2021. The lab report has been approved as it satisfies the academic requirements in respect of **DATA STRUCTURE LAB RECORD (19CS3PCDST)** work prescribed for the said degree.

Signature of the Guide
Prof. Prof. Lohith J J
Assistant Professor
BMSCE, Bengaluru

Signature of the HOD
Dr. Umadevi V
Associate Prof.& Head, Dept. of CSE
BMSCE, Bengaluru

External Viva

Name of the Examiner | Signature with date

1._____ _____

2. _____ _____

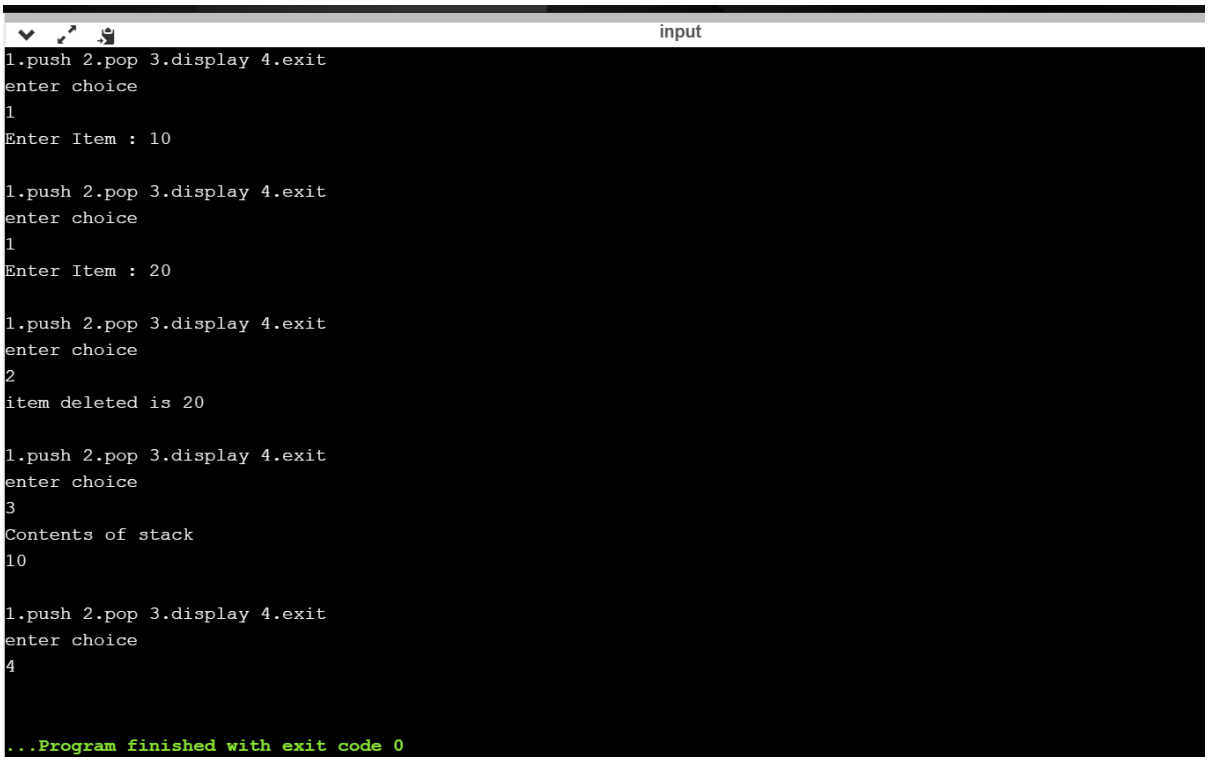
LAB PROGRAM 1

```
#include <stdio.h>
#include <conio.h>
#define stack_size 5
int top=-1;
int s[10];
int item;
void push()
{
   if(top==stack_size-1)
   {
      printf("stack overflow");
      return;
   }
   else
   {
      top=top+1;
      s[top]=item;

   }
}
int pop()
{
   if(top==-1)
   {
      return -1;
   }
   else
```

```
    {
        return s[top--];
    }
}
void display()
{
    int i;
    if(top==-1)
    {
        printf("Stack empty");
        return;
    }
    else
    {
    printf("Contents of stack \n");
    for(i=top;i>=0;i--)
    {
        printf("%d\n",s[i]);
    }
    }
}

int main()
{
    int item_deleted;
    int choice;
    for(;;)
    {
        printf("\n1.push 2.pop 3.display 4.exit \n");
```

```
        printf("enter choice \n");
        scanf("%d",&choice);
        switch(choice)
        {
           case 1:printf("Enter Item : ");
                  scanf("%d",&item);
                  push();
                  break;
           case 2:item_deleted=pop();
                  if(item_deleted==-1)
                  {
                    printf("stack empty \n");
                  }
                  else
                  {
                    printf("item deleted is %d\n",item_deleted);
                  }
                  break;
           case 3:display();
                  break;
           default:exit(0);

        }
     }
     getch();
```

OUTPUT:



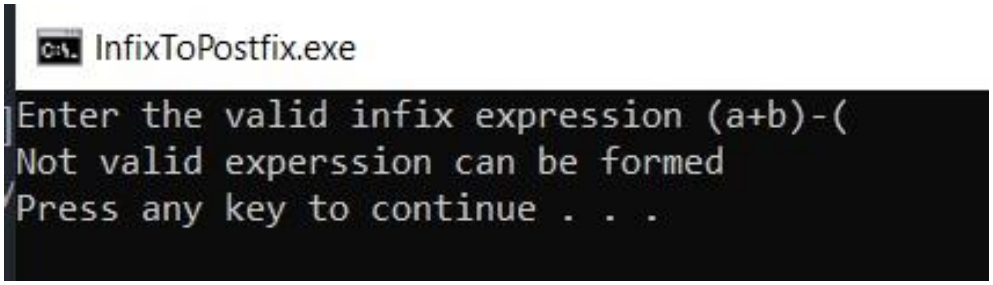
LAB PROGRAM 2

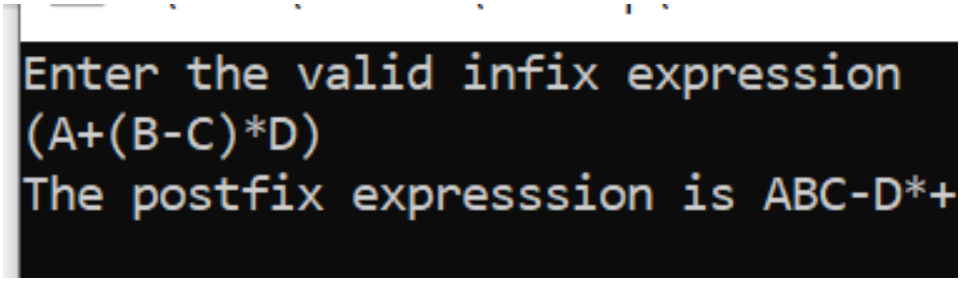
```
//LAB 2 INFIX TO POSTFIX CONVERSION
#include<stdio.h>
#include<string.h>
#include<stdlib.h>
int F(char symbol)
{
switch(symbol)
{
 case'+':
 case'-':return 2;
 case'*':
 case'/':return 4;
 case'^':
```

```
 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;
 default:return 7;
 }
}


int infix_postfix(char infix[],char postfix[])
{
int top,i,j,d=0,f=0;
char s[30],symbol;
top=-1;
```

```
s[++top]='#';
j=0;

for(i=0;i<strlen(infix);i++)
 {
   if(infix[i]=='('){
   d++;}
   else if (infix[i]==')')
   f++;
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';
return (d+f);
```

```
}


void main()
{int a;
char infix[20];
char postfix[20];
printf("Enter the valid infix expression ");
scanf("%s",infix);
a= infix_postfix(infix , postfix );
if((strlen(postfix)+a)!=strlen(infix))
printf("Not valid experssion can be formed \n");
else
printf("The postfix expression is :\t%s\n",postfix);

}
```





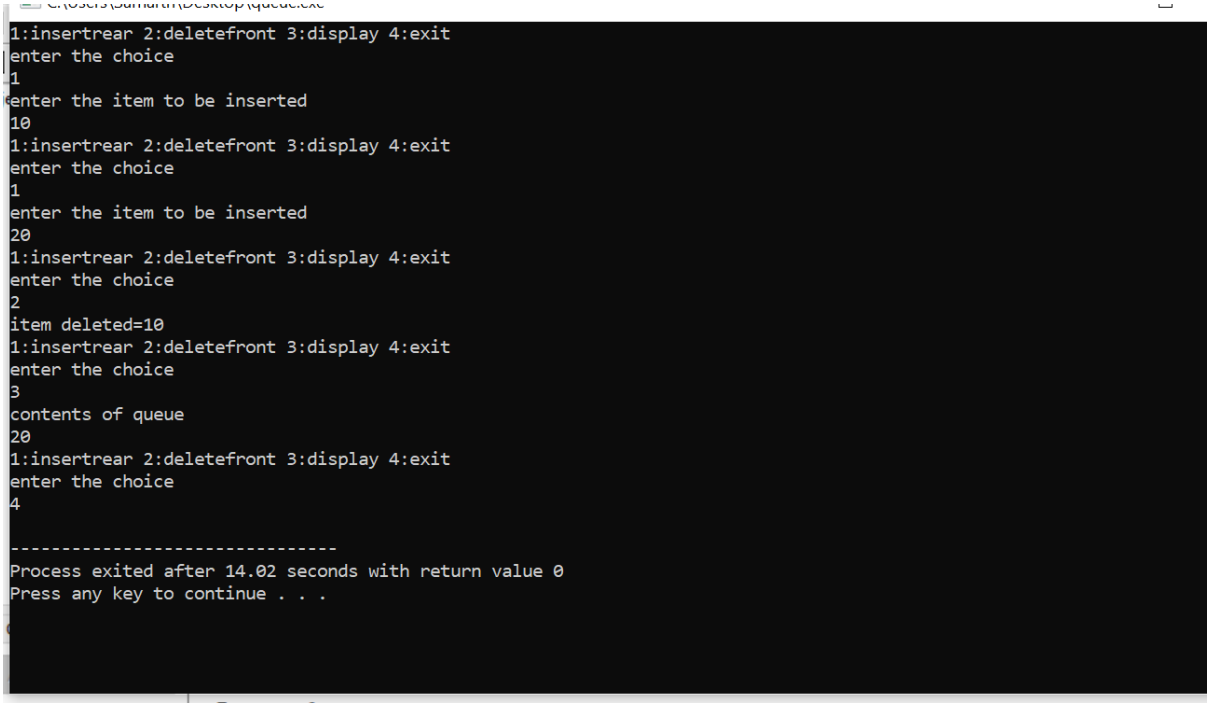
LAB PROGRAM 3

```
#include<stdio.h>
#include<stdlib.h>
#define QUE_SIZE 3
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("1:insertrear 2:deletefront 3:display 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);
```

```
        }
}
```

OUTPUT



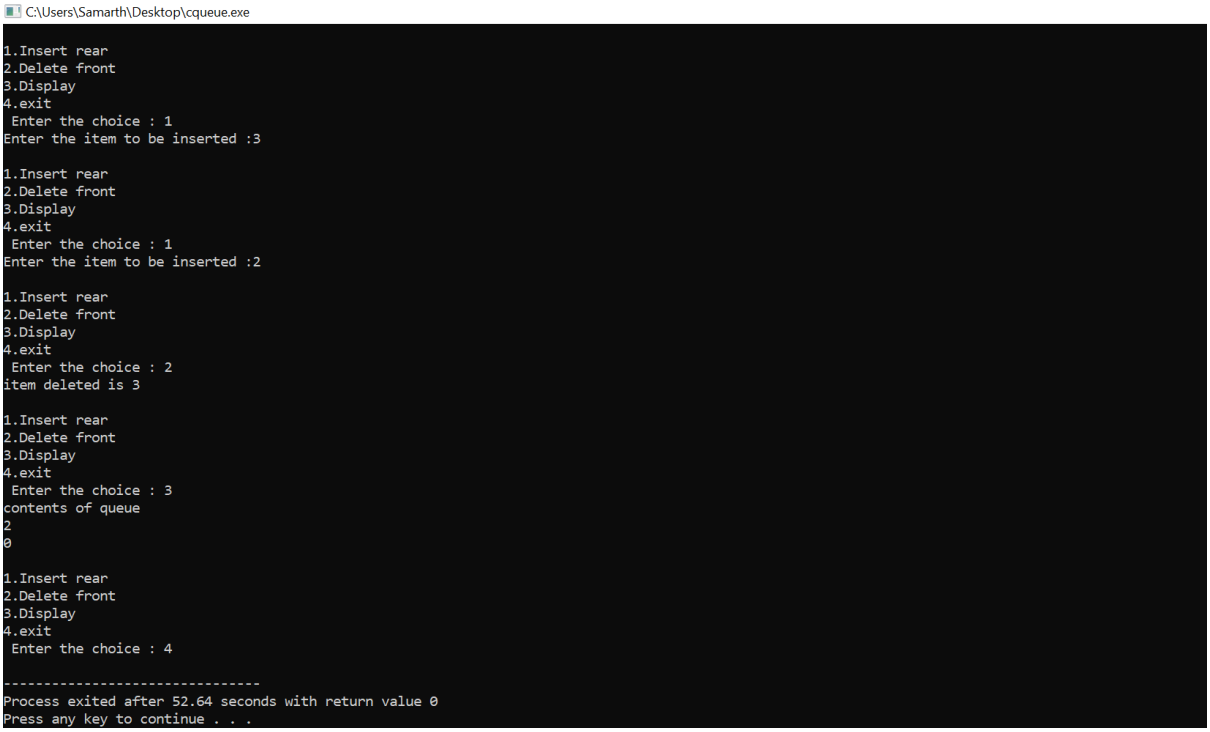
LAB PROGRAM 4

```
#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;
        }
```

```
}
void 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);
                }
        }
        getch();
```

OUTPUT



LAB PROGRAM 5

```
#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("Memory 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 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 insert_pos(int item,int pos,NODE first){
NODE temp,cur,prev;
int count;
temp=getnode();
temp->info=item;
temp->link=NULL;
if(first==NULL&&pos==1){
return temp;
}
if(first==NULL){
printf("Invalid position\n");
return first;
}
if(pos==1){
temp->link=first;
first=temp;
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("Invalid position\n");
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);
}
}
void main()
{
int item,choice,key,pos;
int count=0;
NODE first=NULL;
for(;;){
printf("\n1:Insert rear\n2:Insert front\n3:Insert info position\n4:Display list\n5:Exit\n");
printf("Enter the choice: ");
scanf("%d",&choice);
switch(choice){
case 1:printf("Enter the item at rear end\n");
scanf("%d",&item);
```

```
first=insert_rear(first,item);
break;
case 2:printf("\nEnter the item at front end\n");
scanf("%d",&item);
first=insert_front(first,item);
break;
case 3:printf("Enter the item to be inserted at given position\n");
scanf("%d",&item);
printf("Enter the position\n");
scanf("%d",&pos);
first=insert_pos(item,pos,first);
break;
case 4:display(first);
break;
default:exit(0);
break;
}
}
}
```

OUTPUT:

LAB PROGRAM 6

```
#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("Memory full\n");
exit(0);
```

```
}
return x;
}
void freenode(NODE x){
free(x);
}

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;
```

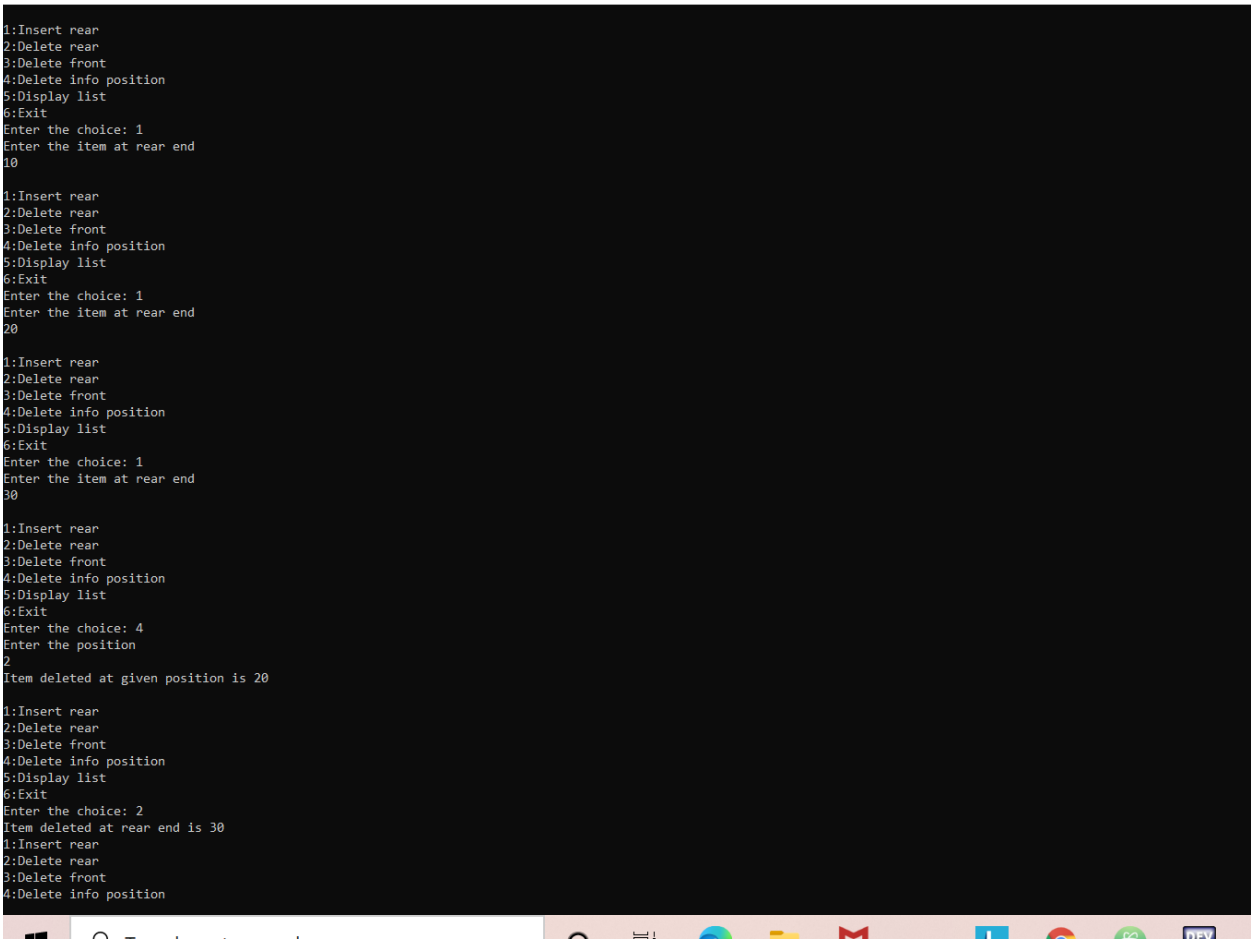
```
cur->link=temp;
return first;
}
NODE delete_rear(NODE first){
NODE cur,prev;
if(first==NULL){
printf("List is empty cannot delete\n");
return first;
}
if(first->link==NULL){
printf("Item deleted is %d\n",first->info);
free(first);
return NULL;
}
prev=NULL;
cur=first;
while(cur->link!=NULL){
prev=cur;
cur=cur->link;
}
printf("Item deleted at rear end is %d",cur->info);
free(cur);
prev->link=NULL;
return first;
}

NODE delete_pos(int pos,NODE first){
NODE cur;
NODE prev;
```

```
int count,flag=0;
if(first==NULL || pos<0){
printf("Invalid position\n");
return NULL;
}
if(pos==1){
cur=first;
first=first->link;
freenode(cur);
return first;
}
prev=NULL;
cur=first;
count=1;
while(cur!=NULL){
if(count==pos){
flag=1;
break;
}
count++;
prev=cur;
cur=cur->link;
}
if(flag==0){
printf("Invalid position\n");
return first;
}
printf("Item deleted at given position is %d\n",cur->info);
prev->link=cur->link;
```

```
freenode(cur);
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);
}
}
void main()
{
int item,choice,key,pos;
int count=0;
NODE first=NULL;
for(;;){
printf("\n1:Insert rear\n2:Delete rear\n3:Delete front\n4:Delete info position\n5:Display
list\n6:Exit\n");
printf("Enter the choice: ");
scanf("%d",&choice);
switch(choice){
case 1:printf("Enter the item at rear end\n");
scanf("%d",&item);
first=insert_rear(first,item);
break;
case 2:first=delete_rear(first);
break;
case 3:first=delete_front(first);
```

```
break;
case 4:printf("Enter the position\n");
scanf("%d",&pos);
first=delete_pos(pos,first);
break;
case 5:display(first);
break;
default:exit(0);
break;
}
}
}
```

OUTPUT



LAB PROGRAM 7

```
#include<stdio.h>
#include<conio.h>
#include<stdlib.h>
#include<process.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;
}
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_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;
}

void display(NODE first)
{
 NODE temp;
 if(first==NULL)
  printf("list empty \n");
```

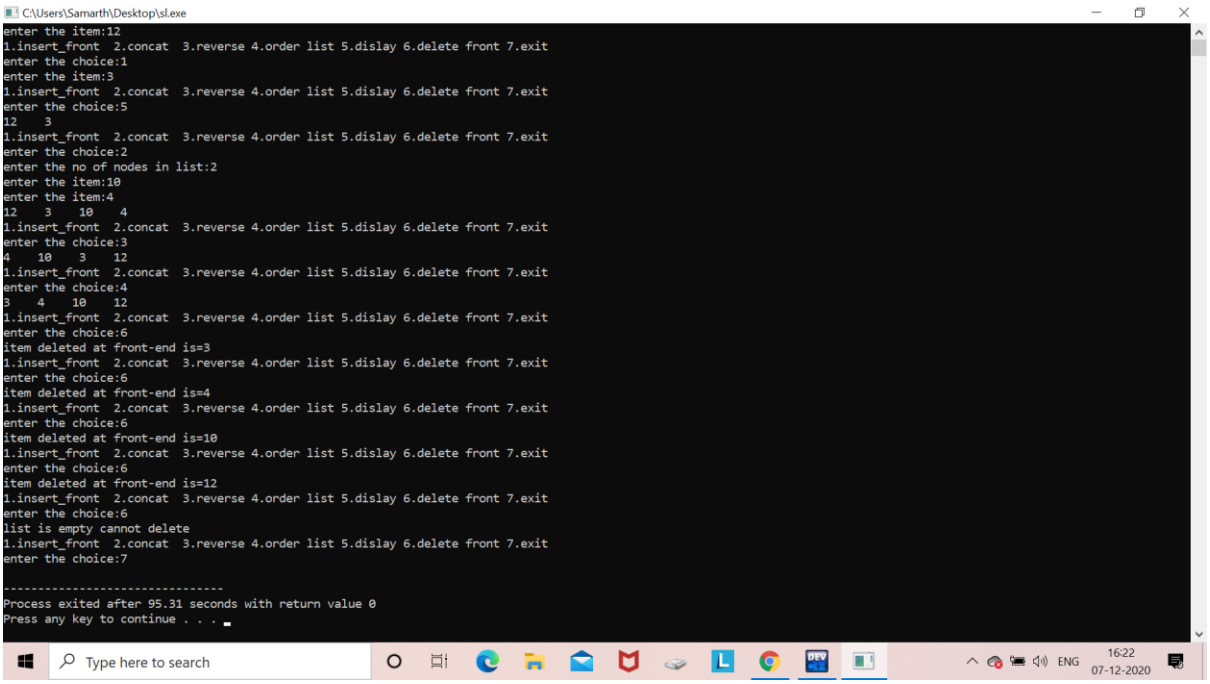
```
 for(temp=first;temp!=NULL;temp=temp->link)
  {
  printf("%d    ",temp->info);
  }
  printf("\n");
}
NODE concat(NODE first,NODE second)
{
 NODE cur;
 if(first==NULL)
  return second;
 if(second==NULL)
  return first;
 cur=first;
 while(cur->link!=NULL)
  cur=cur->link;
 cur->link=second;
 return 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 sortList(NODE first) {
      NODE current = first, index = NULL;
      int temp;

      if(first == NULL) {
           printf("list is empty.");
         return current;
      }
      else {
         while(current != NULL) {

            index = current->link;

            while(index != NULL) {

               if(current->info > index->info) {
                  temp = current->info;
                  current->info = index->info;
                  index->info = temp;
               }
               index = index->link;
            }
            current = current->link;
         }
                              return current;
      }
```

```
   }

int main()
{
int item,choice,pos,i,n;
NODE first=NULL,a,b;
for(;;)
{
printf("1.insert_front  2.concat  3.reverse 4.order list 5.dislay 6.delete front 7.exit\n");
printf("enter the choice:");
scanf("%d",&choice);
switch(choice)
 {
  case 1:printf("enter the item:");
         scanf("%d",&item);
         first=insert_rear(first,item);
         break;
  case 2:printf("enter the no of nodes in list:");
         scanf("%d",&n);
         a=NULL;
         for(i=0;i<n;i++)
          {
           printf("enter the item:");
           scanf("%d",&item);
           a=insert_rear(a,item);
          }
          first=concat(first,a);
          display(first);
         break;
```

```
case 3:first=reverse(first);
      display(first);
      break;
case 4:sortList(first);
                  display(first);
      break;
case 5:display(first);
      break;
case 6:first=delete_front(first);
   break;
default:exit(0);
}
}
return 0;
}
```

OUTPUT



LAB PROGRAM 8

```
#include<stdio.h>
#include<stdlib.h>
#include<math.h>
#include<conio.h>
#include<process.h>
struct node{
struct node *link;
 int info;
};
typedef struct node *NODE;
NODE freenode(NODE x){
free(x);
}
NODE getnode(){
NODE x = (NODE)malloc(sizeof(struct node));
if(x==NULL){
printf("Memory is full\n");
exit(0);
}
return x;
}
NODE insertfront(NODE first,int item){
NODE temp =getnode();
temp->info = item;
temp->link = NULL;
if(first == NULL){
return temp;
```

```
}
temp->link = first;
first = temp;
return first;
}
NODE deletefront(NODE first){
if(first ==NULL){
printf("Stack is Empty\n");
return first;
}
NODE temp = first;
first = first->link;
33
printf("item POPED = %d\n",temp->info);
freenode(temp);
return first;
}
NODE deleterear(NODE first){
NODE prev,curr;
if(first == NULL){
printf("Queue Empty\n");
return first;
}
if(first->link == NULL){
printf("item Delete at rear end is: %d\n",first->info);
free(first);
return NULL;
}
curr = first;
```

```
prev = NULL;
while(curr->link != NULL){
prev = curr;
curr = curr->link ;
}
prev->link = NULL;
printf("item delete from Queue is = %d\n",curr->info);
freenode(curr);
return first;
}
void display(NODE first){
NODE temp;
for(temp=first;temp!=NULL;temp=temp->link){
printf("%d\n",temp->info);
}}
int main(){
34
int item,choice;
NODE first =NULL,first2 =NULL;
for(;;){
printf("1:PUSH item to Stack 2:POP from stack 3:Display Stack 4:Insert Queue 5:Delete Queue
6:Display Queue 6:Exit : \n ");
printf("Enter The Choice: \t");
scanf("%d",&choice);
switch(choice){
case 1 : printf("Enter item:\t");
scanf("%d",&item);
first= insertfront(first,item); break;
case 2 :first=deletefront(first);break;
```

case 3 : if(first==NULL)

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

else display(first); break;

case 4: printf("Enter item:\t");

scanf("%d",&item);

first2 = insertfront(first2,item);break;

case 5: first2 = deleterear(first2);

break;

case 6 : if(first2 ==NULL)

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

else display(first2);break;

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

OUTPUT

```
 Enter The Choice:     1
nter item:     23
:PUSH item to Stack  2:POP from stack  3:Display Stack  4:Insert Queue  5:Delete Queue   6:Display Queue  6:Exit :
 Enter The Choice:     1
nter item:     45
:PUSH item to Stack  2:POP from stack  3:Display Stack  4:Insert Queue  5:Delete Queue   6:Display Queue  6:Exit :
 Enter The Choice:     3
5
5
:PUSH item to Stack  2:POP from stack  3:Display Stack  4:Insert Queue  5:Delete Queue   6:Display Queue  6:Exit :
 Enter The Choice:     2
tem POPED = 45
:PUSH item to Stack  2:POP from stack  3:Display Stack  4:Insert Queue  5:Delete Queue   6:Display Queue  6:Exit :
 Enter The Choice:     4
nter item:     200
:PUSH item to Stack  2:POP from stack  3:Display Stack  4:Insert Queue  5:Delete Queue   6:Display Queue  6:Exit :
 Enter The Choice:     4
nter item:     200
:PUSH item to Stack  2:POP from stack  3:Display Stack  4:Insert Queue  5:Delete Queue   6:Display Queue  6:Exit :
 Enter The Choice:     4
nter item:     800
:PUSH item to Stack  2:POP from stack  3:Display Stack  4:Insert Queue  5:Delete Queue   6:Display Queue  6:Exit :
 Enter The Choice:     6
00
00
00
:PUSH item to Stack  2:POP from stack  3:Display Stack  4:Insert Queue  5:Delete Queue   6:Display Queue  6:Exit :
 Enter The Choice:     5
tem delete from Queue is = 200
:PUSH item to Stack  2:POP from stack  3:Display Stack  4:Insert Queue  5:Delete Queue   6:Display Queue  6:Exit :
 Enter The Choice:     2
tem POPED = 23
:PUSH item to Stack  2:POP from stack  3:Display Stack  4:Insert Queue  5:Delete Queue   6:Display Queue  6:Exit :
 Enter The Choice:     5
tem delete from Queue is = 200
:PUSH item to Stack  2:POP from stack  3:Display Stack  4:Insert Queue  5:Delete Queue   6:Display Queue  6:Exit :
 Enter The Choice:     6
00
:PUSH item to Stack  2:POP from stack  3:Display Stack  4:Insert Queue  5:Delete Queue   6:Display Queue  6:Exit :
 Enter The Choice:     5
tem Delete at rear end is: 800
:PUSH item to Stack  2:POP from stack  3:Display Stack  4:Insert Queue  5:Delete Queue   6:Display Queue  6:Exit :
 Enter The Choice:     2
tack is Empty
:PUSH item to Stack  2:POP from stack  3:Display Stack  4:Insert Queue  5:Delete Queue   6:Display Queue  6:Exit :
 Enter The Choice:     6
ueue empty cannot display items
:PUSH item to Stack  2:POP from stack  3:Display Stack  4:Insert Queue  5:Delete Queue   6:Display Queue  6:Exit :
 Enter The Choice:     7
ress any key to continue . . .
```

LAB PROGRAM 9

```
#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_leftpos(int item,NODE head ,int pos){
 NODE temp,cur,perv;temp=getnode();temp->info=item;
 int i=1;
 cur=head->rlink;
 perv=NULL;
 while(i<pos && cur!=head){
   perv =cur;
 cur=cur->rlink;i++;
 }
 if(cur==head)
 {
  printf("POSITION not found\n");
  return head;
  }
 perv ->rlink=temp;
 temp->rlink=cur;
 temp->llink=perv;
 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 dq\n");
temp=head->rlink;
while(temp!=head)
{
```

```
printf("%d \t",temp->info);
temp=temp->rlink;
}
printf("\n");
}
void main()
{
NODE head,last;
int item,pos, choice;
head=getnode();
head->rlink=head;
head->llink=head;

for(;;)
{
        printf("\n1:insert front\t2:insert rear\t3:delete front\t4:delete rear\t5:display\t6:left-side-insert\t7:exit\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:  printf("enter the item at left side pos to entered\n");
                        scanf("%d",&item);
      printf("POSITION\t");
                        scanf("%d",&pos);
                        last=dinsert_leftpos(item,head,pos);
                        break;
                default:exit(0);
                }
        }
getch();
}
```

OUTPUT

```
1:insert front  2:insert rear   3:delete front  4:delete rear   5:display      6:left-side-insert     7:exit
enter the choice
1
enter the item at front end
10

1:insert front  2:insert rear   3:delete front  4:delete rear   5:display      6:left-side-insert     7:exit
enter the choice
2
enter the item at rear end
20

1:insert front  2:insert rear   3:delete front  4:delete rear   5:display      6:left-side-insert     7:exit
enter the choice
2
enter the item at rear end
30

1:insert front  2:insert rear   3:delete front  4:delete rear   5:display      6:left-side-insert     7:exit
enter the choice
6
enter the item at left side pos to entered
15
POSITION        2

1:insert front  2:insert rear   3:delete front  4:delete rear   5:display      6:left-side-insert     7:exit
enter the choice
5
contents of dq
10      15      20      30

1:insert front  2:insert rear   3:delete front  4:delete rear   5:display      6:left-side-insert     7:exit
enter the choice
3
the node deleted is 10
1:insert front  2:insert rear   3:delete front  4:delete rear   5:display      6:left-side-insert     7:exit
enter the choice
4
the node deleted is 30
1:insert front  2:insert rear   3:delete front  4:delete rear   5:display      6:left-side-insert     7:exit
```

LAB PROGRAM 10

```
#include<stdio.h>
#include<conio.h>
#include<process.h>
struct node
 {
  int info;
  struct node *rlink;
  struct node *llink;
 };
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(NODE root,int item)
{
```

```
NODE temp,cur,prev;
temp=getnode();
temp->rlink=NULL;
temp->llink=NULL;
temp->info=item;
if(root==NULL)
 return temp;
prev=NULL;
cur=root;
while(cur!=NULL)
{
prev=cur;
cur=(item<cur->info)?cur->llink:cur->rlink;
}
if(item<prev->info)
 prev->llink=temp;
else
 prev->rlink=temp;
return root;
}
void display(NODE root,int i)
{
int j;
if(root!=NULL)
 {
  display(root->rlink,i+1);
  for(j=0;j<i;j++)
              printf("  ");
   printf("%d\n",root->info);
```

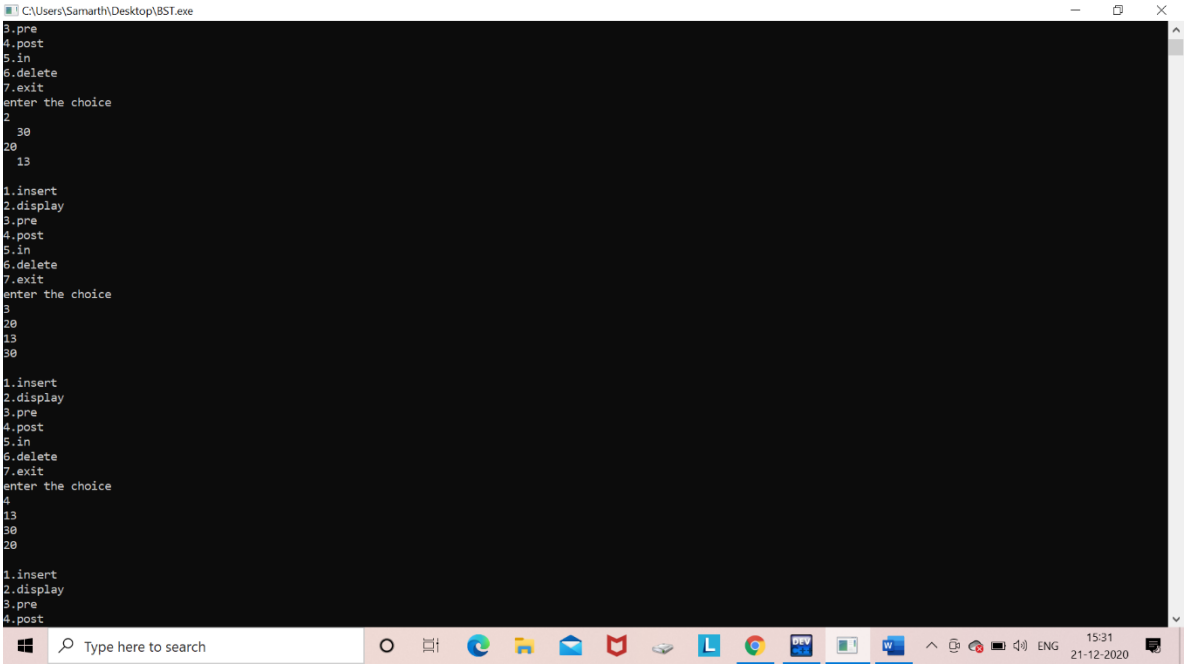
```
            display(root->llink,i+1);
 }
}
NODE delete(NODE root,int item)
{
NODE cur,parent,q,suc;
if(root==NULL)
{
printf("empty\n");
return root;
}
parent=NULL;
cur=root;
while(cur!=NULL&&item!=cur->info)
{
parent=cur;
cur=(item<cur->info)?cur->llink:cur->rlink;
}
if(cur==NULL)
{
 printf("not found\n");
 return root;
}
if(cur->llink==NULL)
 q=cur->rlink;
else if(cur->rlink==NULL)
 q=cur->llink;
else
 {
```

```
 suc=cur->rlink;
 while(suc->llink!=NULL)
  suc=suc->llink;
 suc->llink=cur->llink;
 q=cur->rlink;
 }
 if(parent==NULL)
  return q;
 if(cur==parent->llink)
  parent->llink=q;
 else
  parent->rlink=q;
 freenode(cur);
 return root;
 }

void preorder(NODE root)
{
if(root!=NULL)
 {
  printf("%d\n",root->info);
  preorder(root->llink);
  preorder(root->rlink);
  }
 }
void postorder(NODE root)
{
if(root!=NULL)
 {
```

```
 postorder(root->llink);
 postorder(root->rlink);
 printf("%d\n",root->info);
 }
}
void inorder(NODE root)
{
if(root!=NULL)
 {

 inorder(root->llink);
 printf("%d\n",root->info);
 inorder(root->rlink);
 }
}
void main()
{
int item,choice;
NODE root=NULL;
for(;;)
{
printf("\n1.insert\n2.display\n3.pre\n4.post\n5.in\n6.delete\n7.exit\n");
printf("enter the choice\n");
scanf("%d",&choice);
switch(choice)
 {
  case 1:printf("enter the item\n");
                        scanf("%d",&item);
```

```
                    root=insert(root,item);
                    break;
case 2:display(root,0);
                    break;
case 3:preorder(root);
                    break;
case 4:postorder(root);
                    break;
case 5:inorder(root);
                    break;
case 6:printf("enter the item\n");
                    scanf("%d",&item);
                    root=delete(root,item);
                    break;
default:exit(0);
                     break;
          }
         }
}
```

OUTPUT


C:\Users\Samarth\Desktop\BST.exe
3.pre
4.post
5.in
6.delete
7.exit
enter the choice
2
30
20
13
1.insert
2.display
3.pre
4.post
5.in
6.delete
7.exit
enter the choice
3
20
13
30
1.insert
2.display
3.pre
4.post
5.in
6.delete
7.exit
enter the choice
4
13
30
20
1.insert
2.display
3.pre
4.post
Type here to search
ENG
15:31
21-12-2020