VISVESVARAYA TECHNOLOGICAL UNIVERSITY

"JnanaSangama", Belgaum -590014, Karnataka.

DATA STRUCTURE LAB RECORD

Submitted by

Piyush Dubey(1BM19ET033)
CSE-C

Under the Guidance of

Prof. SHEETAL VA
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 **Piyush Dubey (1BM19ET033)** 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. Sheelal VA
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. _____ _____

Question 1:

Write a program to simulate the working of stack using an array with the following :
a) Push b) Pop c) Display
The program should print appropriate messages for stack overflow, stack underflow

Code:

```
#include <stdio.h>
#define stacksize 10
int stack[50];
int p=-1;

void push(int data)
{
    if (p==49)
        printf("Stack Overflow \n");
    else{
        p+=1;
        stack[p]=data;
    }
}

void pop(){
    if (p==-1)
        printf("Stack Underflow \n");
    else {
        printf("Element deleted: %d \n", stack[p]);
        p-=1;
    }
}

void display(){
    printf("Elements \n");
    for (int i=0;i<=p;i++)
        printf("%d ",stack[i]);
        printf("\n");
}
int main(){
    int n,x;
    for (int i=0;i==0;){
    printf("Enter choice:\n1.Push 2.Pop 3.Display 4.Stop\n");
    scanf("%d",&n);
    switch (n){
    case 1:
        scanf("%d",&x);
        push(x);
```

```
        break;
    case 2:
        pop();
        break;
    case 3:
        display();
        break;
    case 4:
        exit(0);
        break;
    default:
        printf("Wrong Choice\n");
        break;
    }
    }
    return 0;
}
```

Output:

```
Select "D:\Prg data\Stack implementation\bin\Debug\Stack implementation.exe"
1
10
Enter choice:
1.Push 2.Pop 3.Display 4.Stop
1
15
Enter choice:
1.Push 2.Pop 3.Display 4.Stop
1
5
Enter choice:
1.Push 2.Pop 3.Display 4.Stop
1
20
Enter choice:
1.Push 2.Pop 3.Display 4.Stop
3
Elements
10 15 5 20
Enter choice:
1.Push 2.Pop 3.Display 4.Stop
2
Element deleted: 20
Enter choice:
1.Push 2.Pop 3.Display 4.Stop
2
Element deleted: 5
Enter choice:
1.Push 2.Pop 3.Display 4.Stop
2
Element deleted: 15
Enter choice:
1.Push 2.Pop 3.Display 4.Stop
2
Element deleted: 10
Enter choice:
1.Push 2.Pop 3.Display 4.Stop
3
Elements

Enter choice:
1.Push 2.Pop 3.Display 4.Stop
2
Stack Underflow
Enter choice:
1.Push 2.Pop 3.Display 4.Stop
```

Question 2:
WAP to convert a given valid parenthesized infix arithmetic expression to postfix expression. The expression consists of single character operands and the binary operators
+ (plus), - (minus), * (multiply) and / (divide)

Code:

```
#include <stdio.h>
#include <stdlib.h>
#include <process.h>
#include <string.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 Expression: ");
    scanf("%s",infix);
    conversion(infix, postfix);
    printf("The post fix expression: %s",postfix);
    return 0;
}
```

Output:

```
PROBLEMS    OUTPUT    DEBUG CONSOLE    TERMINAL

Windows PowerShell
Copyright (C) Microsoft Corporation. All rights reserved.

Try the new cross-platform PowerShell https://aka.ms/pscore6

PS D:\Prg data> cd "d:\Prg data\C&C++\Stack implementation\" ; if ($?) { gcc inf_to_post.c -o inf_to_post } ; if ($?) { .\inf_to_post }
Enter Expression: A+B*(C-D)+R/T*F
The post fix expression: ABCD-*+RT/F*+
PS D:\Prg data\C&C++\Stack implementation>
```

Question 3:

WAP to simulate the working of a queue of integers using an array. Provide the following operations

a) Insert b) Delete c) Display

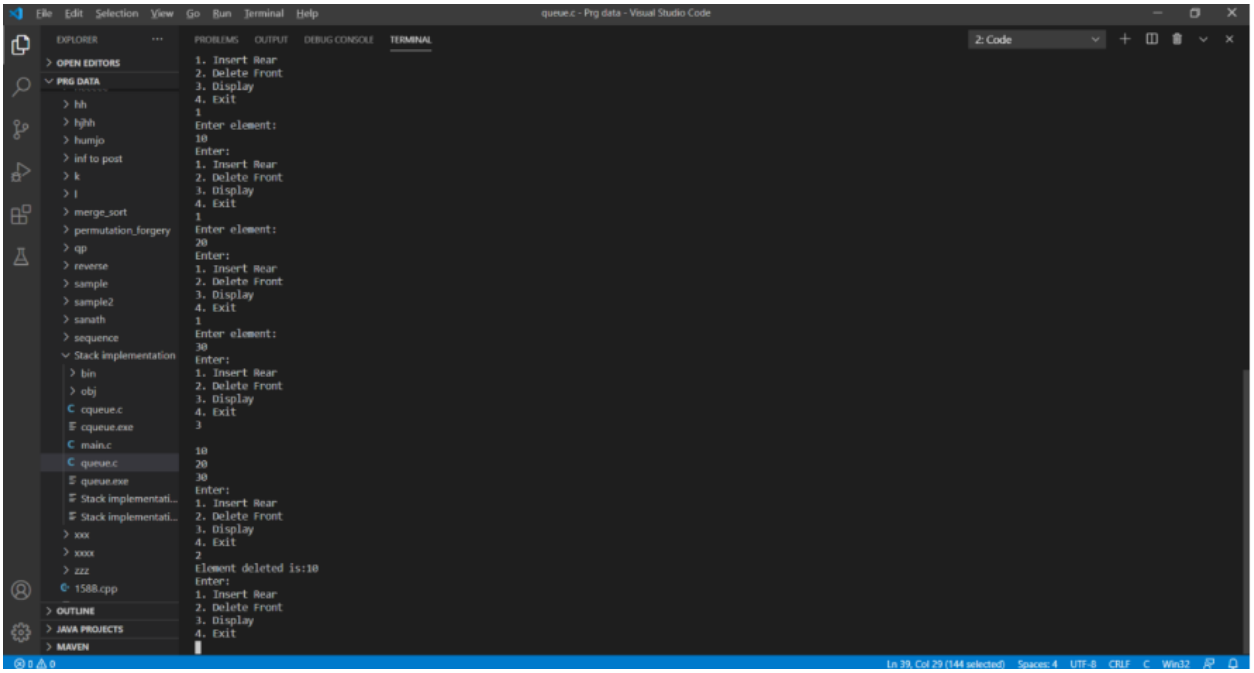
The program should print appropriate messages for queue empty and queue overflow conditions

Code:

```c
#include <stdio.h>
#include <stdlib.h>
#define qsize 5


int item, f=0, r=-1, q[10];

void insertrear(){
    if(r==qsize-1){
        printf("Queue overflow\n");
        return;
    }
    q[++r]=item;
}

int deletefront(){
    if(f>r){
        f=0;
        r=-1;
        return -1;
    }
return q[f++];
}

void display(){
    if (f>r){
```

```
        printf("Queue is empty\n");
        return;
    }
    printf("\n");
    for(int i=f;i<=r;i++){
        printf("%d\n",q[i]);
    }
}
int main(){
    int choice;
    for(;;){
        printf("Enter:\n1. Insert Rear\n2. Delete Front\n3. Display\n4. Exit\n");
        scanf("%d",&choice);
        switch (choice){
            case 1: printf("Enter element:\n");
            scanf("%d", &item);
            insertrear();
            break;
            case 2: item=deletefront();
            if(item==-1)
            printf("Queue empty\n");
            else
            printf("Element deleted is:%d\n",item);
            break;
            case 3: display();
            break;
            case 4: exit(0);
            default: printf("Wrong choice\n");
        }
    }
    return 0;
}
```

Output:



Question 4:

WAP to simulate the working of a circular queue of integers using an array. Provide the following operations.

a) Insert b) Delete c) Display

The program should print appropriate messages for queue empty and queue overflow conditions

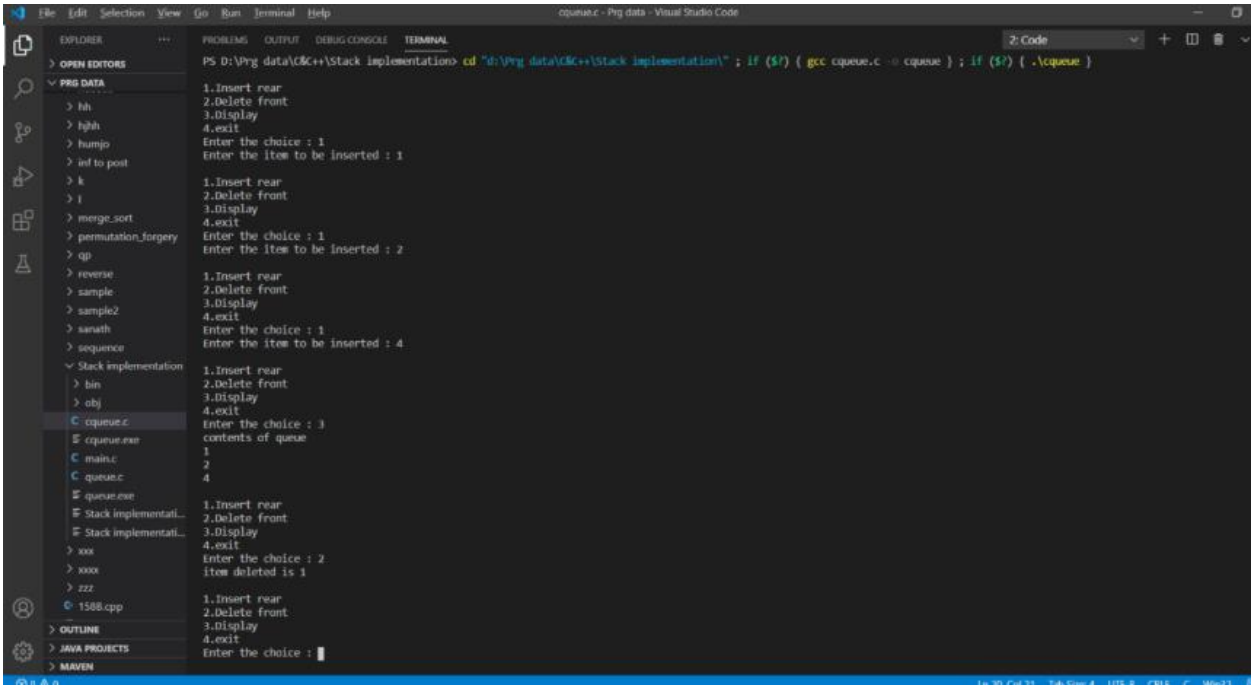
Code:

```
#include<stdio.h>
#include<stdlib.h>
#include<process.h>
#define que_size 5
int item,front=0,rear=-1,q[que_size],count=0;
void insertrear()
{
    if(count==que_size)
    {
        printf("queue overflow\n");
        return;
    }
    rear=(rear+1)%que_size;
    q[rear]=item;
    count++;
}
```

```c
int deletefront()
{
    if(count==0) return -1;
    item = q[front];
    front=(front+1)%que_size;
    count-=1;
    return item;
}
void displayq()
{
    int i,f;
    if(count==0)
    {
        printf("queue is empty\n");
        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;
```

```
            case 4: exit(0);
            default:printf("WRONG CHOICE!");
        }
    }
}
```

Output:



Question 5 and 6:

WAP to Implement Singly Linked List with following operations
a) Create a linked list. b) Insertion of a node at first position, at any position and at end of list. c) Display the contents of the linked list. d)Deletion of first element, specified element and last element in the list. e) Concatenate. f) Reverse. g) sort.

Code:

```
#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:

```
PS D:\Prg data> cd "d:\Prg data\C&C++\Stack implementation\" ; if ($?) { gcc SLL.c -o SLL } ; if ($?) { .\SLL }
1.insert_front
2.delete_front
3.insert_rear
4.delete_rear
5.insert at pos
6.delete at pos
7.concat
8.reverse
9.order list
10.display
enter the choice
1
enter the item at front-end
10
1.insert_front
2.delete_front
3.insert_rear
4.delete_rear
5.insert at pos
6.delete at pos
7.concat
8.reverse
9.order list
10.display
enter the choice
1
enter the item at front-end
20
1.insert_front
2.delete_front
3.insert_rear
4.delete_rear
5.insert at pos
6.delete at pos
7.concat
8.reverse
9.order list
10.display
enter the choice
1
enter the item at front-end
30
1.insert_front
```

```
30
1.insert_front
2.delete_front
3.insert_rear
4.delete_rear
5.insert at pos
6.delete at pos
7.concat
8.reverse
9.order list
10.display
enter the choice
10
30
20
10
1.insert_front
2.delete_front
3.insert_rear
4.delete_rear
5.insert at pos
6.delete at pos
7.concat
8.reverse
9.order list
10.display
enter the choice
5
Enter item
2
enter the position
60
IP
1.insert_front
2.delete_front
3.insert_rear
4.delete_rear
5.insert at pos
6.delete at pos
7.concat
8.reverse
9.order list
10.display
enter the choice
10
```

```
enter the choice
5
Enter item
2
enter the position
60
IP
1.insert_front
2.delete_front
3.insert_rear
4.delete_rear
5.insert at pos
6.delete at pos
7.concat
8.reverse
9.order list
10.display
enter the choice
10
30
20
10
1.insert_front
2.delete_front
3.insert_rear
4.delete_rear
5.insert at pos
6.delete at pos
7.concat
8.reverse
9.order list
10.display
enter the choice
5
Enter item
50
enter the position
2
1.insert_front
2.delete_front
3.insert_rear
4.delete_rear
5.insert at pos
6.delete at pos
7.concat
```

```
8.reverse
9.order list
10.display
enter the choice
10
30
50
20
10
1.insert_front
2.delete_front
3.insert_rear
4.delete_rear
5.insert at pos
6.delete at pos
7.concat
8.reverse
9.order list
10.display
enter the choice
8
10
20
50
30
1.insert_front
2.delete_front
3.insert_rear
4.delete_rear
5.insert at pos
6.delete at pos
7.concat
8.reverse
9.order list
10.display
enter the choice
3
enter the item at rear-end
25
1.insert_front
2.delete_front
3.insert_rear
4.delete_rear
5.insert at pos
6.delete at pos
```

```
PROBLEMS    OUTPUT    DEBUG CONSOLE    TERMINAL

10.display
enter the choice
3
enter the item at rear-end
25
1.insert_front
2.delete_front
3.insert_rear
4.delete_rear
5.insert at pos
6.delete at pos
7.concat
8.reverse
9.order list
10.display
enter the choice
3
enter the item at rear-end
70
1.insert_front
2.delete_front
3.insert_rear
4.delete_rear
5.insert at pos
6.delete at pos
7.concat
8.reverse
9.order list
10.display
enter the choice
10
10
20
50
30
25
70
1.insert_front
2.delete_front
3.insert_rear
4.delete_rear
5.insert at pos
6.delete at pos
7.concat
8.reverse
```

```
5.insert at pos
6.delete at pos
7.concat
8.reverse
9.order list
10.display
enter the choice
10
10
20
25
30
50
70
1.insert_front
2.delete_front
3.insert_rear
4.delete_rear
5.insert at pos
6.delete at pos
7.concat
8.reverse
9.order list
10.display
enter the choice
8
70
50
30
25
20
10
1.insert_front
2.delete_front
3.insert_rear
4.delete_rear
5.insert at pos
6.delete at pos
7.concat
8.reverse
9.order list
10.display
```

```
4.delete_rear
5.insert at pos
6.delete at pos
7.concat
8.reverse
9.order list
10.display
enter the choice
7
enter the no of nodes in 1
3
enter the item
10
enter the item
20
enter the item
30
enter the no of nodes in 2
2
enter the item
15
enter the item
25
10
20
30
15
25
1.insert_front
2.delete_front
3.insert_rear
4.delete_rear
5.insert at pos
6.delete at pos
7.concat
8.reverse
9.order list
10.display
enter the choice
```

Question 7:

WAP to Implement Singly Linked List with following operations
a) Create a linked list. b) Insertion of a node at first position, at any position and at end of list. c) Display the contents of the linked list. d) count e) search. f) sort.

Code:

```
#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;
}

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_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 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 count(NODE first){
    NODE temp;
    temp=first;
    int c=0;
    while(temp!=NULL){
    temp=temp->link;
    c++;
    }
    printf("Number of elements: %d\n",c);
```

```
}

void list_search(NODE first, int key){
    NODE temp;
    temp=first;
    int c=0,f=0;
    while(temp!=NULL){
        c++;
        if(temp->info==key){
            printf("Search successful, element position: %d\n",c);
            f=1;break;
        }
        temp=temp->link;
    }
    if(f==0)
    printf("Search Unsuccessful!\n");
}
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);
  }
}
int main(){
    int item,choice,pos,i,n;
    NODE first=NULL;
    for(;;)
        {
        printf("1.insert-
front\n2.delete_rear\n3.display\n4.count items\n5.search\n6.order\nAny other key
to 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);
     first=insert_front(first,item);
     break;
  case 2:first=delete_rear(first);
     break;
  case 3:display(first);
```

```
    break;
  case 4:count(first);
  break;
  case 5:printf("Enter element to be searched: ");
  scanf("%d",&item);
  list_search(first,item);
  break;
  case 6:
  first=order_list(first);
  break;
 default:exit(0);
 }
}
}
```

Output:

```
1
enter the item at front-end
10
1.insert-front
2.delete_rear
3.display
4.count items
5.search
6.order
Any other key to exit
enter the choice
1
enter the item at front-end
30
1.insert-front
2.delete_rear
3.display
4.count items
5.search
6.order
Any other key to exit
enter the choice
1
enter the item at front-end
40
1.insert-front
2.delete_rear
3.display
4.count items
5.search
6.order
Any other key to exit
enter the choice
3
40
30
10
20
1.insert-front
2.delete_rear
3.display
4.count items
5.search
6.order
Any other key to exit
1.insert-front
2.delete_rear
3.display
4.count items
5.search
6.order
Any other key to exit
enter the choice
4
Number of elements: 4
1.insert-front
2.delete_rear
3.display
4.count items
5.search
6.order
Any other key to exit
enter the choice
5
Enter element to be searched: 10000
Search Unsuccessful!
1.insert-front
2.delete_rear
3.display
4.count items
5.search
6.order
Any other key to exit
enter the choice
6
1.insert-front
2.delete_rear
3.display
4.count items
5.search
6.order
Any other key to exit
enter the choice
3
10
20
30
40
1.insert-front
2.delete_rear
```

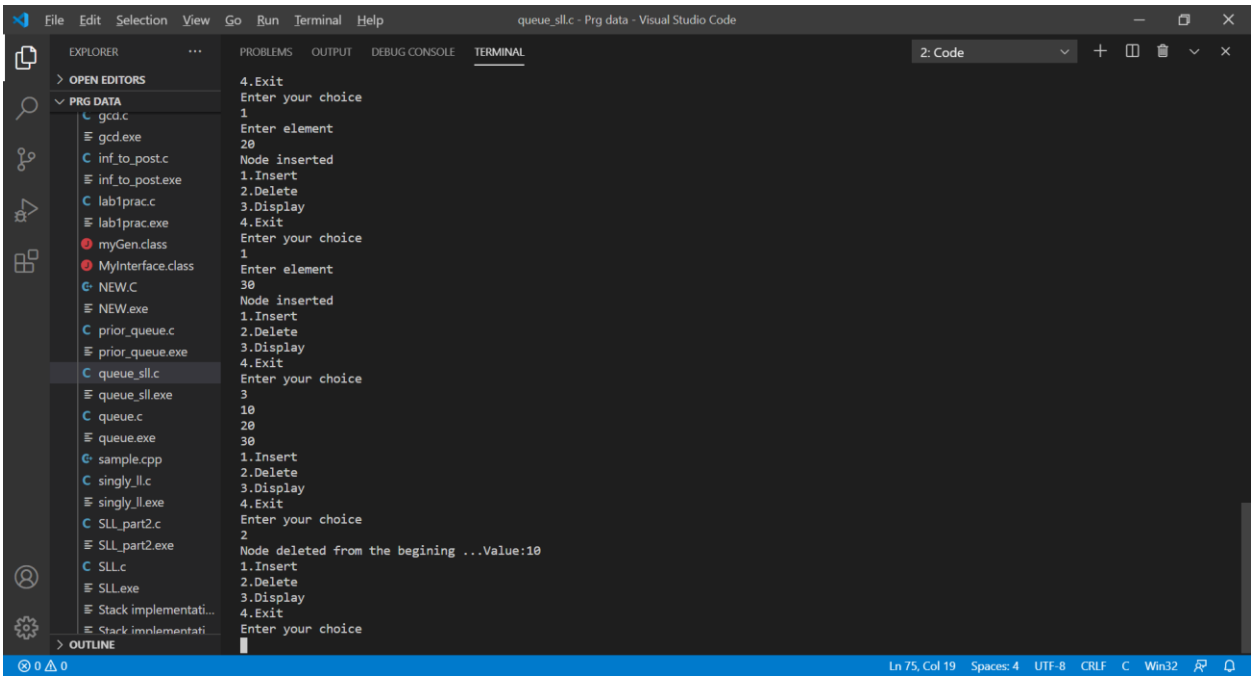
Question 8:
WAP to implement Stack & Queues using Linked Representation.

Code queue:

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

struct node *head=NULL;
int item;

void Enqueue(int item)
{
    struct node *ptr,*temp;

    ptr = (struct node*)malloc(sizeof(struct node));
    ptr->data = item; ptr -> next = NULL;
        if(head == NULL)
        {
            head = ptr;
            printf("Node inserted\n");
        }
        else
        {
            temp = head;
            while (temp -> next != NULL)
            {
                temp = temp -> next;
            }
            temp->next = ptr;
            printf("Node inserted\n");
           }
    }
  void Dequeue()
    {
        struct node *ptr;
        if(head == NULL)
        {
            printf("List is empty\n");
        }
        else
```

```
        {
            ptr = head;
            head = ptr->next;
            printf("Node deleted from the begining ...Value:%d\n",ptr->data);
            free(ptr);
        }
    }
void display(){
    struct node *temp;
    if(head == NULL)
        printf("Queue is empty\n");
        temp=head;
        while(temp!=NULL)
        {
            printf("%d\n",temp->data);
            temp=temp->next;
        }
}
void main(){
    int choice;
    while(1){
        printf("1.Insert\n2.Delete\n3.Display\n4.Exit\nEnter your choice\n");
        scanf("%d",&choice);
        switch(choice){
            case 1: printf("Enter element\n");
            scanf("%d",&item);
            Enqueue(item);
            break;
            case 2:
            Dequeue();
            break;
            case 3:
            display();
            break;
            case 4: exit(0);
            default : printf ("Wrong Choice!!!\n");
        }
    }
}
```

Output queue:



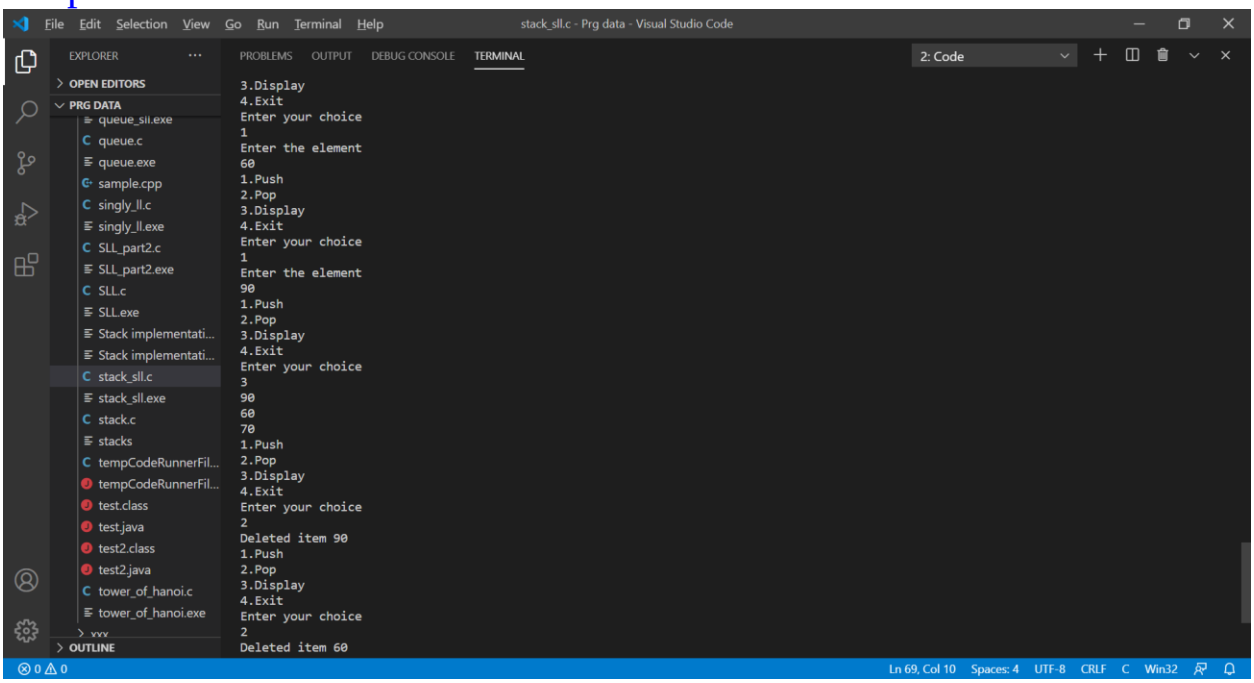
Code stack:

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

struct node *top=NULL;

void push()
{
    struct node *new_node;
    new_node=(struct node *)malloc(sizeof(struct node));
    printf("Enter the element\n");
    scanf("%d",&new_node->data);
    new_node->next=NULL;
    if(top==NULL)
    {
        top=new_node;
    }
    else
    {
```

```
        new_node->next=top;
        top=new_node;
    }

}
  void pop()
    {
        if(top==NULL)
        printf("Stack is empty\n");
        else
        {
            printf("Deleted item %d\n",top->data);
            top=top->next;
        }
    }
void display(){
    struct node *temp;
    if(top == NULL)
        printf("Stack is empty\n");
    else
    {
        temp=top;
        while(temp!=NULL)
        {
            printf("%d\n",temp->data);
            temp=temp->next;
        }
    }
}
void main(){
    int choice;
    while(1){
        printf("1.Push\n2.Pop\n3.Display\n4.Exit\nEnter your choice\n");
        scanf("%d",&choice);
        switch(choice){
            case 1: push();
            break;
            case 2:
            pop();
            break;
            case 3:
            display();
            break;
            case 4: exit(0);
            default : printf ("Wrong Choice!!!\n");
        }
```

```
    }
}
```

Output stack:



Question 9:

WAP Implement doubly link list with primitive operations
a) Create a doubly linked list. b) Insert a new node to the left of the node. b) Delete the node based on a specific value. c) Display the contents of the list

Code:

```
#include <stdio.h>
#include <stdlib.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("Memory full\n");
        exit(0);
    }
    return x;
}
NODE dinsert_rear(int item, NODE head){
    NODE temp,cur;
    temp=getnode();
    temp->info=item;
    cur=head->llink;
    temp->llink=cur;
    cur->rlink=temp;
    head->llink=temp;
    temp->rlink=head;
    return head;
}
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 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 item deleted is %d\n",cur->info);
    free(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 item deleted is %d\n",cur->info);
    free(cur);
    return head;
}
NODE lsearch(NODE head, int key, int z){
    NODE cur,prev,temp;
    int f=0,c=1;
    if(head->rlink==head)
    {
        printf("list empty\n");
        return head;
    }
    cur=head->rlink;
    while(cur!=head)
    {
        if(cur->info==key){
            f=1;
            break;
        }
        cur=cur->rlink;
        c++;
    }
    if(f==1 && z==0) {
    printf("Search successful, found at index %d\n",c);
    return head;
    }
    if(f==1 && z==1){
    prev=cur->llink;
    printf("enter towards left of %d=",key);
    temp=getnode();
    scanf("%d",&temp->info);
    prev->rlink=temp;
    temp->llink=prev;
    cur->llink=temp;
    temp->rlink=cur;
```

```
    return head;
    }
    if(f==1 && z==2){
    prev=cur;
    cur=cur->rlink;
    printf("enter towards right of %d=",key);
    temp=getnode();
    scanf("%d",&temp->info);
    prev->rlink=temp;
    temp->llink=prev;
    cur->llink=temp;
    temp->rlink=cur;
    return head;
    }
    printf("Search unsuccessful\n");
}
NODE delete_all_key(int item,NODE head)
{
NODE prev,cur,next;
int count;
   if(head->rlink==head)
    {
     printf("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;
  free(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;
}

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 ",temp->info);
        temp=temp->rlink;
    }
    printf("\n");
}
void main(){
    NODE head, last;
    int item, choice;
    head=getnode();
    head->rlink=head;
    head->llink=head;
    for(;;){
        printf("Enter choice:\n1. Insert Front\n2. Delete front\n3. Insert rear\n
4. Delete rear\n5. Simple search\n6. Insert left of key\n7. Insert right of key\n
8. Delete all occurunces of key\n9. Display\n--- Any other key to exit ---\n");
        scanf("%d",&choice);
        switch(choice){
            case 1: printf("Enter the item at front end\n");
            scanf("%d",&item);
            head=dinsert_front(item,head);
            break;
        case 3: printf("enter the item at rear end\n");
            scanf("%d",&item);
            head=dinsert_rear(item,head);
            break;
        case 2:
            head=ddelete_front(head);
            break;
        case 4:
            head=ddelete_rear(head);
```

```
            break;
        case 5:printf("Enter key\n");
        scanf("%d",&item);
        head=lsearch(head,item,0);
        break;
        case 6:printf("Enter key\n");
        scanf("%d",&item);
        head=lsearch(head,item,1);
        break;
        case 7:printf("Enter key\n");
        scanf("%d",&item);
        head=lsearch(head,item,2);
        break;
        case 8: printf("Enter key\n");
        scanf("%d",&item);
        head=delete_all_key(item,head);
        break;
        case 9: display(head);
            break;
        default:exit(0);

        }

    }

}
```

Output:

```
--- Any other key to exit ---
6
Enter key
20
enter towards left of 20=45
Enter choice:
1. Insert front
2. Delete front
3. Insert rear
4. Delete rear
5. Simple search
6. Insert left of key
7. Insert right of key
8. Delete all occurunces of key
9. Display
--- Any other key to exit ---
7
Enter key
20
enter towards right of 20=99
Enter choice:
1. Insert front
2. Delete front
3. Insert rear
4. Delete rear
5. Simple search
6. Insert left of key
8. Delete all occurunces of key
9. Display
--- Any other key to exit ---
9
contents of dq
30 45 20 99 10
Enter choice:
1. Insert front
2. Delete front
3. Insert rear
```

```
4. Delete rear
5. Simple search
6. Insert left of key
7. Insert right of key
8. Delete all occurunces of key
9. Display
--- Any other key to exit ---
8
Enter key
30
key found at 3 positions and are deleted
Enter choice:
1. Insert front
2. Delete front
3. Insert rear
4. Delete rear
5. Simple search
6. Insert left of key
7. Insert right of key
8. Delete all occurunces of key
9. Display
--- Any other key to exit ---
9
contents of dq
20 10
Enter choice:
1. Insert front
2. Delete front
3. Insert rear
4. Delete rear
5. Simple search
6. Insert left of key
7. Insert right of key
8. Delete all occurunces of key
9. Display
--- Any other key to exit ---
10
```

Question 10:

Write a program

a) To construct a binary Search tree.

b) To traverse the tree using all the methods i.e., in-order, preorder and post order

c) To display the elements in the tree.

Code:

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

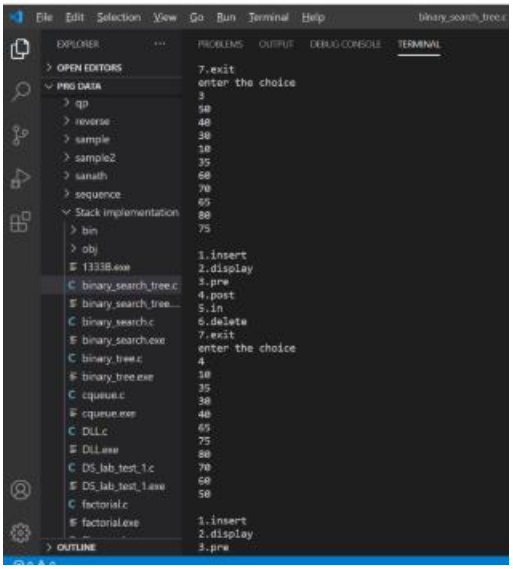
```
{
prev=cur;
cur=(item<cur->info)?cur->llink:cur->rlink;
}
if(item<prev->info)
 prev->llink=temp;
else
 prev->rlink=temp;
return root;
}
void display(NODE root,int i)
{
int j;
if(root!=NULL)
 {
  display(root->rlink,i+1);
  for(j=0;j<i;j++)
    printf("  ");
   printf("%d\n",root->info);
   display(root->llink,i+1);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

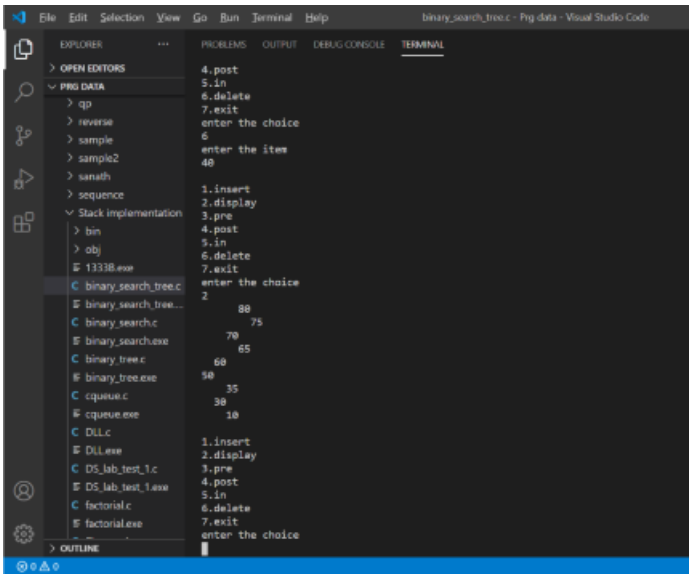
```
else
 {
 suc=cur->rlink;
 while(suc->llink!=NULL)
  suc=suc->llink;
 suc->llink=cur->llink;
 q=cur->rlink;
 }
 if(parent==NULL)
  return q;
 if(cur==parent->llink)
  parent->llink=q;
 else
  parent->rlink=q;
 freenode(cur);
 return root;
 }

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:


enter the choice
1
enter the item
65
1.insert
2.display
3.pre
4.post
5.in
6.delete
7.exit
enter the choice
3
enter the item
10
1.insert
2.display
3.pre
4.post
5.in
6.delete
7.exit
enter the choice
2
80
75
70
65
60
50
40
35
30
10


7.exit
enter the choice
3
50
40
30
10
35
60
70
65
80
75
1.insert
2.display
3.pre
4.post
5.in
6.delete
7.exit
enter the choice
4
10
35
30
40
65
75
80
70
60
50
1.insert
2.display
3.pre


4.post
5.in
6.delete
7.exit
enter the choice
6
enter the item
40
1.insert
2.display
3.pre
4.post
5.in
6.delete
7.exit
enter the choice
2
80
75
70
65
60
50
35
30
10
1.insert
2.display
3.pre
4.post
5.in
6.delete
7.exit
enter the choice