LAB:-8(Doubley Linked List HRank)

Q :- Hackerrank Solutions

```
//*****************This program is written by Manan Jain(211B173)******************//
SinglyLinkedListNode* insertNodeAtTail(SinglyLinkedListNode* head, int data) {
    SinglyLinkedListNode *new_node= new SinglyLinkedListNode(data);
    SinglyLinkedListNode *temp=head;
    new_node->next=NULL;
    if(temp==NULL)
        return new_node;
    while(temp->next!=NULL){
        temp=temp->next;
    }
    temp->next=new_node;
    return head;}

void printLinkedList(SinglyLinkedListNode* head) {
     while(head!=NULL){
          cout<<head->data<<endl;
          head=head->next;}}

SinglyLinkedListNode* insertNodeAtHead(SinglyLinkedListNode* llist, int data) {
     SinglyLinkedListNode * new_node=new SinglyLinkedListNode(data);
     new_node->next=llist;
     return new_node;}

void reversePrint(SinglyLinkedListNode* llist) {
     if(llist==NULL){
          return;
     }
     reversePrint(llist->next);
     cout<<llist->data<<endl;}

bool has_cycle(SinglyLinkedListNode* head) {
     SinglyLinkedListNode * temp=head;
     bool ans=false;
     while(temp!=NULL){
          if(temp->data!=NULL){
               temp->data=NULL;
               temp=temp->next; }
          else{
               ans=true;
               break;
          }     }
     return ans;}
```

Q :- Write a menu driven program for implementing doubly linked list

```
//*****************This program is written by Manan Jain(211B173)******************//

#include <stdio.h>
#include <stdlib.h>
struct node {
          int info;
          struct node *prev, *next;
};
struct node* start = NULL;
void traverse()
```

Manan Jain 211B173

```
{
        if (start == NULL) {
                printf("\nList is empty\n");
                return;
        }
        struct node* temp;
        temp = start;
        while (temp != NULL) {
                printf("Data = %d\n", temp->info);
                temp = temp->next;
        }

void insertAtFront()
{
        int data;
        struct node* temp;
        temp = (struct node*)malloc(sizeof(struct node));
        printf("\nEnter number to be inserted: ");
        scanf("%d", &data);
        temp->info = data;
        temp->prev = NULL;
        temp->next = start;
        start = temp;
}
void insertAtEnd()
{
        int data;
        struct node *temp, *trav;
        temp = (struct node*)malloc(sizeof(struct node));
        temp->prev = NULL;
        temp->next = NULL;
        printf("\nEnter number to be inserted: ");
        scanf("%d", &data);
        temp->info = data;
        temp->next = NULL;
        trav = start;
        if (start == NULL) {

                start = temp;
        }
        else {
                while (trav->next != NULL)
                        trav = trav->next;
                temp->prev = trav;
                trav->next = temp;
        }
}
void insertAtPosition()
{
        int data, pos, i = 1;
        struct node *temp, *newnode;
        newnode = malloc(sizeof(struct node));
        newnode->next = NULL;
        newnode->prev = NULL;
        printf("\nEnter position : ");
        scanf("%d", &pos);
        if (start == NULL) {
                start = newnode;
                newnode->prev = NULL;
                newnode->next = NULL;
```

Manan Jain 211B173

```
    }
    else if (pos == 1) {
    insertAtFront();
    }
    else {
    printf("\nEnter number to be inserted: ");
    scanf("%d", &data);
    newnode->info = data;
    temp = start;
        while (i < pos - 1) {
            temp = temp->next;
            i++;
        }
        newnode->next = temp->next;
        newnode->prev = temp;
        temp->next = newnode;
        temp->next->prev = newnode;
    }
}
void deleteFirst()
{
    struct node* temp;
    if (start == NULL)
        printf("\nList is empty\n");
    else {
        temp = start;
        start = start->next;
        if (start != NULL)
            start->prev = NULL;
        free(temp);
    }
}
void deleteEnd()
{
    struct node* temp;
    if (start == NULL)
        printf("\nList is empty\n");
    temp = start;
    while (temp->next != NULL)
        temp = temp->next;
    if (start->next == NULL)
        start = NULL;
    else {
        temp->prev->next = NULL;
        free(temp);
    }
}
void deletePosition()
{int pos, i = 1;
    struct node *temp, *position;
    temp = start;
    if (start == NULL)
        printf("\nList is empty\n");
    else {
        printf("\nEnter position : ");
        scanf("%d", &pos);
        if (pos == 1) {
            deleteFirst();
            if (start != NULL) {
                start->prev = NULL;
```

Manan Jain 211B173

```
            }
            free(position);
            return;
        }
        while (i < pos - 1) {
            temp = temp->next;
            i++;
        }
        position = temp->next;
        if (position->next != NULL)
            position->next->prev = temp;
        temp->next = position->next;
        free(position);
    }}
int main()
{
    int choice;
    while (1) {
        printf("\n\t1 To see list\n");
        printf("\t2 For insertion at"
            " starting\n");
        printf("\t3 For insertion at"
            " end\n");
        printf("\t4 For insertion at "
            "any position\n");
        printf("\t5 For deletion of "
            "first element\n");
        printf("\t6 For deletion of "
            "last element\n");
        printf("\t7 For deletion of "
            "element at any position\n");
        printf("\t8 To exit\n");
        printf("\nEnter Choice :\n");
        scanf("%d", &choice);
        switch (choice) {
        case 1:
            traverse();
            break;
        case 2:
            insertAtFront();
            break;
        case 3:
            insertAtEnd();
            break;
        case 4:
            insertAtPosition();
            break;
        case 5:
            deleteFirst();
            break;
        case 6:
            deleteEnd();
            break;
        case 7:
            deletePosition();
            break;
        case 8:
            exit(1);
            break;
        default:
```

Manan Jain 211B173

```
            printf("Incorrect Choice. Try Again \n");
            continue;
        }}return 0;}
```

Q :- WAP to count the number of nodes in circular linked list if only start pointer of circular linked list is given

```
//*****************This program is written by Manan Jain(211B173)******************//
#include<iostream>
using namespace std;
struct Node {
    int data;
    Node* next;
    Node(int x)
    {
        data = x;
        next = NULL;
    }
};
int countNodes(Node* head)
{
    Node* temp = head;
    int result = 0;
    if (head != NULL) {
        do {
            temp = temp->next;
            result++;
        } while (temp != head);}
    return result;}
int main()
{
    /* Initialize lists as empty */
    Node* head = NULL;
    head = push(head, 12);
    head = push(head, 56);
    head = push(head, 2);
    head = push(head, 11);
    cout << countNodes(head);
    return 0;
}
```

Manan Jain 211B173

LAB:-9(STACK)

Q :- Write a menu-driven program to implement stack using array

```
//******************This program is written by Manan Jain(211B173)*******************//
#include<iostream>
using namespace std;
class STACK
{
public:
    int top;
    int arr[5];
    STACK()
    {top=-1;  }
    bool isEmpty()
    {
        if(top<4)
            return true;
        else
            return false;}
    void push(int val)
    {
        if(isEmpty())
        {
            arr[top+1]=val;
            top=top+1;
        }
        else
            cout<<"Stack Overflow!"<<endl;
    }
    void pop()
    {
        int p=0;
        p=arr[top];
        if(top==-1)
            cout<<"Stack Underflow!"<<endl;
        else
            {
                top=top-1;
                cout<<"Popped Value : "<<p<<endl;
            }
    }

    void display()
    {
        cout<<"STACK : "<<endl;
        for(int i=top;i>=0;i--)
        {
            cout<<arr[i]<<endl;
        }
        cout<<endl;
    }
};

int main()
{
    STACK s1;
    int choice;
```

Manan Jain 211B173

```
    int data;
    while(1)
        {
            cout<<"***Stack Menu**"<<endl;
            cout<<"1.push()"<<endl;
            cout<<"2.pop()"<<endl;
            cout<<"3.display()"<<endl;
            cout<<"4.Exit"<<endl;
            cout<<"Enter Choice(1-4):";
            cin>>choice;
            switch(choice)
            {
            case 1:
                cout<<"Enter Element to push : ";
                cin>>data;
                s1.push(data);
                cout<<endl;
                break;
            case 2:
                s1.pop();
                break;
            case 3:
                s1.display();
                break;
            case 4:
                break;
            default:
                cout<<"Enter Valid Choice"<<endl;
                break;
            }
            if(choice==4)
                break;
        }
    return 0;
}
```

Q :- Write a menu-driven program to using array implement Linked List

//*****************This program is written by Manan Jain(211B173)*******************//

```
#include<iostream>
using namespace std;

class Node
{
public:
    Node* NEXT;
    int info;

    Node()
    {
        NEXT=NULL;
        info=0;
    }

    Node(int val)
    {
        NEXT=NULL;
        info=val;
    }
```

Manan Jain 211B173

```
};
class STACK
{
  public:
     Node* start;
     STACK()
     {
        start=NULL;
     }
     Node* CreateS()
     {
        return start;
     }

     void push(Node** top,int data)
     {
        Node *new_node= new Node(data);
        if(new_node==NULL)
        {
           cout<<"STACK OVERFLOW"<<endl;
        }
        else
          {


        if(*top==NULL)
        {
           *top=new_node;
           new_node->NEXT=NULL;
        }
        else
        {
           new_node->NEXT=*top;
           *top=new_node;
        }
          }

     }

     void pop(Node** top)
     {
        Node* temp;
        temp=*top;

        if(temp==NULL)
        {
           cout<<"STACK UNDERFLOW"<<endl;
        }
        else
        {
           int p;
           p=temp->info;
           cout<<"Popped Value : "<<p<<endl;
           *top=temp->NEXT;
           delete temp;
        }
     }
```

Manan Jain 211B173

```
     void display(Node** top)
     {
         Node* temp=*top;
         cout<<"STACK : "<<endl;
         while(temp!=NULL)
         {
             cout<<temp->info<<endl;
             temp=temp->NEXT;
         }
         cout<<endl;
     }
};
int main()
{
    STACK *s1 = new STACK();
    Node* top=s1->CreateS();
    int choice;
    int data;
    while(1)
       {
          cout<<"***Stack Menu***"<<endl;
          cout<<"1.push()"<<endl;
          cout<<"2.pop()"<<endl;
          cout<<"3.display()"<<endl;
          cout<<"4.Exit"<<endl;
          cout<<"Enter Choice(1-4):";
          cin>>choice;
          cout<<endl;
          switch(choice)
          {
          case 1:
              cout<<"Enter Element to push : ";
              cin>>data;
              s1->push(&top,data);
              cout<<endl;
              break;
          case 2:
              s1->pop(&top);
              break;
          case 3:
              s1->display(&top);
              break;
          case 4:
              break;
          default:
              cout<<"Enter Valid Choice"<<endl;
              break;
          }
          if(choice==4)
              break;
       }
    return 0;
}
```

Q :- WAP to convert an expression from postfix to infix.

*//*****************This program is written by Manan Jain(211B173)******************//*

#include <bits/stdc++.h>

Manan Jain 211B173

```
using namespace std;
bool isOperand(char x) {
    return (x >= 'a' && x <= 'z') || (x >= 'A' && x <= 'Z');
}
string infixConversion(string postfix) {
    stack<string> infix;
    for (int i=0; postfix[i]!='\0'; i++) {
        if (isOperand(postfix[i])) {
            string op(1, postfix[i]);
            infix.push(op);
        } else {
            string op1 = infix.top();
            infix.pop();
            string op2 = infix.top();
            infix.pop();
            infix.push("{"+op2+postfix[i]+op1 +"}");
        }
    }
    return infix.top();
}
int main() {
    string postfix = "xyae+/%";
    cout<<"The infix conversion of the postfix expression '"<<postfix<<"' is : ";
    cout<<infixConversion(postfix);
    return 0;
}
```

Q :- WAP to convert an expression from infix to postfix.

```
//******************This program is written by Manan Jain(211B173)******************//
#include<iostream>
#include<stack>
#include<locale>
using namespace std;
int preced(char ch) {
   if(ch == '+' || ch == '-') {
      return 1;
   }else if(ch == '*' || ch == '/') {
      return 2;
   }else if(ch == '^') {
      return 3;
   }else {
      return 0;
   }
}

string inToPost(string infix ) {
   stack<char> stk;
   stk.push('#');
   string postfix = "";
   string::iterator it;

   for(it = infix.begin(); it!=infix.end(); it++) {
```

Manan Jain 211B173

```
      if(isalnum(char(*it)))
         postfix += *it;
      else if(*it == '(')
         stk.push('(');
      else if(*it == '^')
         stk.push('^');
      else if(*it == ')') {
         while(stk.top() != '#' && stk.top() != '(') {
            postfix += stk.top();
            stk.pop();
         }
         stk.pop();
      }else {
         if(preced(*it) > preced(stk.top()))
            stk.push(*it); //push if precedence is high
         else {
            while(stk.top() != '#' && preced(*it) <= preced(stk.top())) {
               postfix += stk.top();
               stk.pop();
            }
            stk.push(*it);
         }   }}
   while(stk.top() != '#') {
      postfix += stk.top();
      stk.pop();}
   return postfix;
}

int main() {
   string infix = "x^y/(5*z)+2";
   cout << "Postfix Form Is: " << inToPost(infix) << endl;
}
```

Q :- WAP to convert an expression from infix to prefix

```
//******************This program is written by Manan Jain(211B173)*******************//
#include<iostream>
#include<stack>
#include<locale> //for function isalnum()
#include<algorithm>
using namespace std;
int preced(char ch) {
   if(ch == '+' || ch == '-') {
      return 1;      //Precedence of + or - is 1
   }else if(ch == '*' || ch == '/') {
      return 2;      //Precedence of * or / is 2
   }else if(ch == '^') {
      return 3;      //Precedence of ^ is 3
   }else {
      return 0;
   }
}

string inToPost(string infix) {
```

Manan Jain 211B173

```
    stack<char> stk;
    stk.push('#');
    string postfix = "";
    string::iterator it;

    for(it = infix.begin(); it!=infix.end(); it++) {
        if(isalnum(char(*it)))
            postfix += *it;
        else if(*it == '(')
            stk.push('(');
        else if(*it == '^')
            stk.push('^');
        else if(*it == ')') {
            while(stk.top() != '#' && stk.top() != '(') {
                postfix += stk.top();
                stk.pop();
            }

            stk.pop();
        }else {
            if(preced(*it) > preced(stk.top()))
                stk.push(*it);
            else {
                while(stk.top() != '#' && preced(*it) <= preced(stk.top())) {
                    postfix += stk.top();
                    stk.pop();
                }
                stk.push(*it);
            }
        }
    }

    while(stk.top() != '#') {
        postfix += stk.top();

        stk.pop();

    }
    return postfix;
}

string inToPre(string infix) {
    string prefix;
    reverse(infix.begin(), infix.end());
    string::iterator it;

    for(it = infix.begin(); it != infix.end(); it++) {
        if(*it == '(')
            *it = ')';
        else if(*it == ')')
```

Manan Jain 211B173

```
            *it = '(';
    }

    prefix = inToPost(infix);
    reverse(prefix.begin(), prefix.end());
    return prefix;
}

int main() {
    string infix = "x^y/(5*z)+2";
    cout << "Prefix Form Is: " << inToPre(infix) << endl;}
```

Q :- WAP to evaluate postfix expression

```
//*****************This program is written by Manan Jain(211B173)******************//
#include<iostream>
#include<cmath>
#include<stack>
using namespace std;
float scanNum(char ch) {
    int value;
    value = ch;
    return float(value-'0');     }
int isOperator(char ch) {
    if(ch == '+'|| ch == '-'|| ch == '*'|| ch == '/' || ch == '^')
        return 1;
    return -1;     }
int isOperand(char ch) {
    if(ch >= '0' && ch <= '9')
        return 1;
    return -1;   }
float operation(int a, int b, char op) {    if(op == '+')
        return b+a;
    else if(op == '-')
        return b-a;
    else if(op == '*')
        return b*a;
    else if(op == '/')
        return b/a;
    else if(op == '^')
        return pow(b,a);
    elsereturn INT_MIN;   }
float postfixEval(string postfix) {
    int a, b;
    stack<float> stk;
    string::iterator it;
    for(it=postfix.begin(); it!=postfix.end(); it++) {
        if(isOperator(*it) != -1) {
            a = stk.top();
            stk.pop();
            b = stk.top();
            stk.pop();
```

Manan Jain 211B173

```
            stk.push(operation(a, b, *it));
        }else if(isOperand(*it) > 0) {
            stk.push(scanNum(*it)); } }
    return stk.top();}
int main() {
    string post = "53+62/*35*+";
    cout << "The result is: "<<postfixEval(post);}
```

Q :- WAP to implement tower of Hanoi puzzle

```
//*****************This program is written by Manan Jain(211B173)******************//
#include<iostream>
using namespace std;
void TOH(int d, char t1, char t2, char t3)
{
if(d==1)
{cout<<"\nShift top disk from tower "<<t1<<" to tower "<<t2;
return;}
TOH(d-1,t1,t3,t2);
cout<<"\nShift top disk from tower "<<t1<<" to tower "<<t2;
TOH(d-1,t3,t2,t1);}
int main()
{int disk;
cout<<"Enter the number of disks: "; cin>>disk;
if(disk<1)
            cout<<"There are no disks to shift";
else
            cout<<"There are "<<disk<<" disks in tower 1\n";
            TOH(disk, '1','2','3');
cout<<"\n\n"<<disk<<" disks in tower 1 are shifted to tower 2";
return 0;}
```

Manan Jain 211B173

LAB:-10(Queue)

Q :- Write a menu driven program to implement linear queue using array

*//*****************This program is written by Manan Jain(211B173)******************//*

```
#include<iostream>
using namespace std;

class qarr
{
public:
	int arr[5];
	int frnt;
	int rear;

	qarr()
	{
		frnt=-1;
		rear=-1;
	}

	void Enqueue(int value)
	{
		if(frnt==-1 && rear ==-1)
		{
			frnt=0;
			rear=0;
		}
		else if (rear==4)
		{
			if(frnt==0)
			cout<<"QUEUE IS FULL!";

		int temp=frnt;
		int i=0;
		while(temp<=rear)
		{
			arr[i]=arr[temp];
			temp+=1;
			i+=1;
		}
		rear=rear-frnt+1;
		frnt=0;
		}
		else
			rear+=1;
		arr[rear]=value;
	}
	void Dequeue()
	{
		int item=arr[frnt];
		if(rear==-1 && frnt==-1)
		{
			cout<<"EMPTY QUEUE"<<endl;
```

Manan Jain 211B173

```
        }
        else if(frnt==rear)
        {
            frnt=-1;
            rear=-1;
            cout<<"DELETED VALUE:"<<item;
        }
        else
        {
            frnt++;
            cout<<"DELETED VALUE:"<<item;
        }

    }

    void disfrnt()
    {
        if(frnt==-1 && rear==-1)
        {
            cout<<"EMPTY QUEUE"<<endl;
        }
        else
            cout<<"Front : "<<arr[frnt];
    }

    void display()
    {
        if(frnt==-1 && rear==-1)
        {
            cout<<"EMPTY QUEUE"<<endl;
        }
        else{
        cout<<"Queue : ";
        for(int i=frnt;i<=rear;i++)
        {
            cout<<arr[i]<<" ";
        }
        }
    }
};

int main()
{
    qarr q1;
    int choice;
    int data;
    while(1)
        {
            cout<<endl<<"***Queue Menu**"<<endl;
            cout<<"1.enqueue()"<<endl;
            cout<<"2.dequeue()"<<endl;
            cout<<"3.displayfront()"<<endl;
            cout<<"4.displayall()"<<endl;
            cout<<"5.exit()"<<endl;
            cout<<"Enter Choice(1-5):";
            cin>>choice;
            switch(choice)
```

Manan Jain 211B173

```
            {
            case 1:
                cout<<"Enter Element to enqueue : ";
                cin>>data;
                q1.Enqueue(data);
                cout<<endl;
                break;
            case 2:
                q1.Dequeue();
                break;
            case 3:
                q1.disfrnt();
                break;
            case 4:
                q1.display();
                break;
            case 5:
                break;
            default:
                cout<<"Enter Valid Choice"<<endl;
                break;
            }
            if(choice==5)
                break;
        }
    return 0;
}
```

Q :- Write a menu driven program to implement circular queue using array

```
//******************This program is written by Manan Jain(211B173)*******************//
#include<iostream>
using namespace std;

class qarr
{
public:
    int arr[5];
    int frnt;
    int rear;

    qarr()
    {
        frnt=-1;
        rear=-1;
    }

    void Enqueue(int value)
    {
        if(frnt==-1 && rear ==-1)
        {
            frnt=0;
            rear=0;
        }
        else if (rear==4)
        {
            if(frnt==0)
            {
```

Manan Jain 211B173

```
            cout<<"QUEUE IS FULL!"<<endl;
            return;
        }
        else
        rear=0;
    }
    else if((rear+1)==(frnt))
    {
        cout<<"QUEUE IS FULL!"<<endl;
        return;
    }
    else
        rear+=1;
    arr[rear]=value;
}

void Dequeue()
{
    if(rear==-1 || frnt==-1)
    {
        cout<<"EMPTY QUEUE"<<endl;
        return;
    }
    int item=arr[frnt];
    if(frnt==rear)
    {
        frnt=-1;
        rear=-1;
        cout<<"DELETED VALUE:"<<item;
    }
    else if(frnt==4)
    {
        frnt=0;
        cout<<"DELETED VALUE:"<<item;
    }
    else
    {
        frnt=frnt+1;
        cout<<"DELETED VALUE:"<<item;
    }
}

void disfrnt()
{
    if(frnt==-1 && rear==-1)
    {
        cout<<"EMPTY QUEUE"<<endl;
    }
    else
        cout<<"Front : "<<arr[frnt];
}

void display()
{
    if(frnt==-1 && rear==-1)
    {
        cout<<"EMPTY QUEUE"<<endl;
```

Manan Jain 211B173

```
        }
        else if(frnt<=rear)
        {
            for(int i=frnt;i<=rear;i++)
            {
                cout<<arr[i]<<" ";
            }
        }
        else
        {
            for(int i=frnt;i<5;i++)
            {
                cout<<arr[i]<<" ";
            }
            for(int i=0;i<=rear;i++)
            {
                cout<<arr[i]<<" ";
            }
        }

    }
};

int main()
{
    qarr q1;
    int choice;
    int data;
    while(1)
        {
            cout<<endl<<"***Queue Menu**"<<endl;
            cout<<"1.enqueue()"<<endl;
            cout<<"2.dequeue()"<<endl;
            cout<<"3.displayfront()"<<endl;
            cout<<"4.displayall()"<<endl;
            cout<<"5.exit()"<<endl;
            cout<<"Enter Choice(1-5):";
            cin>>choice;
            switch(choice)
            {
            case 1:
                cout<<"Enter Element to enqueue : ";
                cin>>data;
                q1.Enqueue(data);
                cout<<endl;
                break;
            case 2:
                q1.Dequeue();
                break;
            case 3:
                q1.disfrnt();
                break;
            case 4:
                q1.display();
                break;
            case 5:
                break;
```

Manan Jain 211B173

```
            default:
                cout<<"Enter Valid Choice"<<endl;
                break;
            }
            if(choice==5)
                break;
        }
    return 0;
}
```

Q :- Write a menu driven program to implement linear queue using linked list

```
//*****************This program is written by Manan Jain(211B173)******************//
#include<iostream>
using namespace std;

class Node
{
public:
    Node* NEXT;
    int info;

    Node()
    {
        NEXT=NULL;
        info=0;
    }

    Node(int val)
    {
        NEXT=NULL;
        info=val;
    }

};
class QUEUE
{
  public:
      Node* start;
      Node* tail;
      QUEUE()
      {
          start=NULL;
          tail=NULL;
      }
      Node* sCreateQ()
      {
          return start;
      }
      Node* tCreateQ()
      {
          return tail;
      }

      void Enqueue(Node** top,Node** Tail,int data)
      {
          Node *new_node= new Node(data);
```

Manan Jain 211B173

```
    if(new_node==NULL)
    {
        cout<<"------QUEUE OVERFLOW------"<<endl;
    }
    else
  {


    if(*top==NULL && *Tail==NULL)
    {
        *top=new_node;
        *Tail=new_node;
        new_node->NEXT=NULL;
    }
    else
    {
        (*Tail)->NEXT=new_node;
        *Tail=new_node;
    }
  }

}

void Dequeue(Node** top,Node** Tail)
{
    if(*top==NULL)
      cout<<"------QUEUE UNDERFLOW!------"<<endl;
    Node *temp;
    temp=*top;
    if (*top==*Tail)
    {
        *top=NULL;
        *Tail=NULL;
        delete temp;
    }
    else
    {
        *top=temp->NEXT;
        delete temp;
    }

}
void disfrnt(Node** top)
{
    if(*top==NULL)
      cout<<"------EMPTY QUEUE------"<<endl;
    else
      cout<<(*top)->info<<endl;
}

void display(Node** top)
{
    Node* temp=*top;
    if(*top==NULL)
      cout<<"------EMPTY QUEUE!------"<<endl;
    else
```

Manan Jain 211B173

```
          {
          cout<<"QUEUE : "<<endl;
          while(temp!=NULL)
          {
              cout<<temp->info<<" ";
              temp=temp->NEXT;
          }

          cout<<endl;
          }
      }
};

int main()
{
    QUEUE *q1=new QUEUE();
    Node* head;
    Node* tail;
    head=q1->sCreateQ();
    tail=q1->tCreateQ();
    int choice;
    int data;
    while(1)
        {
            cout<<endl<<"********Queue Menu********"<<endl;
            cout<<"1.enqueue()"<<endl;
            cout<<"2.dequeue()"<<endl;
            cout<<"3.displayfront()"<<endl;
            cout<<"4.displayall()"<<endl;
            cout<<"5.exit()"<<endl;
            cout<<"Enter Choice(1-5):";
            cin>>choice;
            switch(choice)
            {
            case 1:
                cout<<"Enter Element to enqueue : ";
                cin>>data;
                q1->Enqueue(&head,&tail,data);
                cout<<endl;
                break;
            case 2:
                q1->Dequeue(&head,&tail);
                break;
            case 3:
                q1->disfrnt(&head);
                break;
            case 4:
                q1->display(&head);
                break;
            case 5:
                break;
            default:
                cout<<"Enter Valid Choice"<<endl;
                break;
            }
            if(choice==5)
                break;
```

Manan Jain 211B173

```
        }
    return 0;
}
```

Q :- WAP to implement priority queue with its basic operations

```
//******************This program is written by Manan Jain(211B173)*******************//
#include <bits/stdc++.h>
using namespace std;
typedef struct node {
    int data;
    int priority;
    struct node* next;
} Node;
Node* newNode(int d, int p){
    Node* temp = (Node*)malloc(sizeof(Node));
    temp->data = d;
    temp->priority = p;
    temp->next = NULL;
    return temp;}
int peek(Node** head) { return (*head)->data; }
void pop(Node** head)
{Node* temp = *head;
    (*head) = (*head)->next;
    free(temp);}
void push(Node** head, int d, int p)
{Node* start = (*head);
    Node* temp = newNode(d, p);
    if ((*head)->priority < p) {
         temp->next = *head;
        (*head) = temp;
    }
    else {
        while (start->next != NULL
                && start->next->priority > p) {
            start = start->next;
        }
         temp->next = start->next;
        start->next = temp;}}

int isEmpty(Node** head) { return (*head) == NULL; }
int main()
{
    Node* pq = newNode(4, 1);
    push(&pq, 5, 2);
    push(&pq, 6, 3);
    push(&pq, 7, 0);
    while (!isEmpty(&pq)) {
        cout << " " << peek(&pq);
        pop(&pq);}
    return 0;}
```

Manan Jain 211B173

LAB:-11(TREES)

Q :- WAP to check whether given tree is a binary search tree or not.

```
//*****************This program is written by Manan Jain(211B173)******************//
#include <bits/stdc++.h>
using namespace std;
  struct Node {
	int data;
	struct Node *left, *right;
	Node(int data)
	{ this->data = data;
	   left = right = NULL; }
};
  bool isBSTUtil(struct Node* root, Node*& prev)
{
	if (root) {
		if (!isBSTUtil(root->left, prev))
			return false;
		if (prev != NULL && root->data <= prev->data)
			return false;
		prev = root;
		return isBSTUtil(root->right, prev);
	}return true;}
bool isBST(Node* root){
	Node* prev = NULL;
	return isBSTUtil(root, prev);
}int main()
{
	struct Node* root = new Node(3);
	root->left = new Node(2);
	root->right = new Node(5);
	root->left->left = new Node(1);
	root->left->right = new Node(4
	if (isBST(root))
		cout << "Is BST";
	else { cout << "Not a BST"; return 0;}
```

Q :- WAP to implement inorder, preorder and postorder traversal in binary tree.WAP to search a node in a given binary search tree. WAP to insert a node in a given binary search tree.

```
//*****************This program is written by Manan Jain(211B173)******************//
#include<iostream>
using namespace std;

class Node
{
public:
	Node* LC;
	int INFO;
	Node* RC;

	Node()
	{
		LC=NULL;
		RC=NULL;
		INFO=0;
	}
	Node(int data)
	{
```

Manan Jain 211B173

```
        LC=NULL;
        RC=NULL;
        INFO=data;
    }

};
Node* LOC=NULL;
Node* PAR=NULL;

class BST
{
public:
    Node* root;
    BST()
    {
        root=NULL;
    }

    Node* createBST()
    {
        return root;
    }

    void search_bst(Node** head,int key)
    {
        LOC=NULL;
        PAR=NULL;
        if(*head==NULL)
            return;
        if((*head)->INFO==key)
        {
            LOC=*head;
            PAR=NULL;
            return;
        }

        Node* temp=*head;
        while(temp!=NULL)
        {
            PAR=temp;
            if(key<temp->INFO)
                temp=temp->LC;
            else
                temp=temp->RC;
            if(temp!=NULL && temp->INFO==key)
            {
                LOC=temp;
                break;
            }
        }
        if(LOC==NULL)
            return ;
    }

    void Insert_BST(Node** head,int key)
    {
        //Node *temp=*head;
```

Manan Jain 211B173

```
    search_bst(head,key);
    if(LOC==NULL)
        return;
    Node* new_node=new Node(key);
    new_node->LC=NULL;
    new_node->RC=NULL;
    if(head==NULL)
        *head=new_node;
    else if(key<PAR->INFO)
        PAR->LC=new_node;
    else
        PAR->RC=new_node;
}

void inorder(Node** start)
{
    cout<<"IN FUNCTION!"<<endl;
    Node* temp=*start;
    if(temp==NULL)
    {
        cout<<"NULL"<<endl;
        return;

    }
    else
    {
        inorder(&(temp->LC));
        cout<<temp->INFO<<" ";
        inorder(&(temp->RC));

    }
}
void preorder(Node** head)
{
    Node* temp=*head;
    if(temp==NULL)
        return;
    else
    {
        preorder(&(temp->LC));
        preorder(&(temp->RC));
        cout<<temp->INFO<<" ";

    }
}
void postorder(Node** start)
{
    Node* temp=*start;
    if(temp==NULL)
        return;
    else
    {
        cout<<temp->INFO<<" ";
        postorder(&(temp->LC));
        postorder(&(temp->RC));

    }
```

Manan Jain 211B173

```
        }
};
int main()
{
        BST *B=new BST();
        Node* head=B->createBST();
        B->Insert_BST(&head,6);
        B->Insert_BST(&head,2);
        B->Insert_BST(&head,1);
        B->Insert_BST(&head,8);
        B->Insert_BST(&head,5);
        B->Insert_BST(&head,3);
        B->Insert_BST(&head,4);
        B->Insert_BST(&head,10);
        B->inorder(&head);
        cout<<"ALL GOOD";
        return 0;
}
```

Q :- WAP to delete a node from a given binary search tree

```
//*****************This program is written by Manan Jain(211B173)******************//
#include <bits/stdc++.h>
using namespace std;

struct Node {
    int key;
    struct Node *left, *right;
};
Node* newNode(int item)
{
    Node* temp = new Node;
    temp->key = item;
    temp->left = temp->right = NULL;
    return temp;
}
void inorder(Node* root)
{
    if (root != NULL) {
        inorder(root->left);
        printf("%d ", root->key);
        inorder(root->right);
    }
}
Node* insert(Node* node, int key)
{
    if (node == NULL)
        return newNode(key);
    if (key < node->key)
        node->left = insert(node->left, key);
    else
        node->right = insert(node->right, key);
    return node;
}
Node* deleteNode(Node* root, int k)
{
    if (root == NULL)
        return root;
```

Manan Jain 211B173

```
    if (root->key > k) {
        root->left = deleteNode(root->left, k);
        return root;
    }
    else if (root->key < k) {
        root->right = deleteNode(root->right, k);
        return root;
    }
    if (root->left == NULL) {
        Node* temp = root->right;
        delete root;
        return temp;
    }
    else if (root->right == NULL) {
        Node* temp = root->left;
        delete root;
        return temp;
    }
    else {
        Node* succParent = root;
        Node* succ = root->right;
        while (succ->left != NULL) {
            succParent = succ;
            succ = succ->left;
        }
        if (succParent != root)
            succParent->left = succ->right;
        else
            succParent->right = succ->right;
        root->key = succ->key;
        delete succ;
        return root;
    }
}
int main()
{
    Node* root = NULL;
    root = insert(root, 50);
    root = insert(root, 30);
    root = insert(root, 20);
    root = insert(root, 40);
    root = insert(root, 70);
    root = insert(root, 60);
    root = insert(root, 80);
    printf("Inorder traversal of the given tree \n");
    inorder(root);
    printf("\n\nDelete 20\n");
    root = deleteNode(root, 20);
    printf("Inorder traversal of the modified tree \n");
    inorder(root);
    printf("\n\nDelete 30\n");
    root = deleteNode(root, 30);
    printf("Inorder traversal of the modified tree \n");
    inorder(root);
    printf("\n\nDelete 50\n");
    root = deleteNode(root, 50);
    printf("Inorder traversal of the modified tree \n");
```

Manan Jain 21B173

```
    inorder(root);
    return 0;}
```

Q :- Write the programs for following:

a. Determining the height of binary tree
b. Determining no. of nodes of binary tree
c. Determining no. of internal nodes of binary tree
d. Determining no. of external nodes (leaf nodes) of binary tree

```
//******HEIGHT************This program is written by Manan Jain(211B173)********************//
#include <iostream>
using namespace std;
class node {
public:
	int data;
	node* left;
	node* right;
};

int maxDepth(node* node)
{
	if (node == NULL)
		return 0;
	else {
		int lDepth = maxDepth(node->left);
		int rDepth = maxDepth(node->right);
		if (lDepth > rDepth)
			return (lDepth + 1);
		else
			return (rDepth + 1);
	}
}
node* newNode(int data)
{
	node* Node = new node();
	Node->data = data;
	Node->left = NULL;
	Node->right = NULL;
	return (Node);}
int main()
{
	node* root = newNode(1);
	root->left = newNode(2);
	root->right = newNode(3);
	root->left->left = newNode(4);
	root->left->right = newNode(5);
	cout << "Height of tree is " << maxDepth(root);
	return 0;
}
```

Manan Jain 211B173

```
//******NODES ************This program is written by Manan Jain(211B173)*******************//
#include <iostream>
using namespace std;
int count(node *tree)
{
        int c =     1;
        if (tree ==NULL)
                return 0;
        else
        {
                c += count(tree->left);
                c += count(tree->right);
                return c;
        }
}
int main()
{

        c = count(root);
        printf("Number of node %d \n",c);
}
```

```
//*******Internal Nodes *******This program is written by Manan Jain(211B173)*******************//
include <iostream>
using namespace std;
struct Node {
    int data;
    struct Node* left;
    struct Node* right;};
int countNonleaf(struct Node* root)
{
    if (root == NULL || (root->left == NULL &&
                         root->right == NULL))
        return 0;
    return 1 + countNonleaf(root->left) +
               countNonleaf(root->right);}
int main()
{
    struct Node* root = newNode(1);
    root->left = newNode(2);
    root->right = newNode(3);
    root->left->left = newNode(4);
    root->left->right = newNode(5);
    cout << countNonleaf(root);
    return 0;}
```

```
//*******Leaf Nodes *******This program is written by Manan Jain(211B173)*******************//
#include <iostream>
using namespace std;

struct node{
        int data;
        struct node* left;
        struct node* right;};
long double getLeafCount(struct node* node){
        if(node == NULL)
```

Manan Jain 211B173

```
			return 0;
		if(node->left == NULL && node->right == NULL)
			return 1;
		else
			return getLeafCount(node->left)+getLeafCount(node->right);}
int main()
{		struct node *root = newNode(1);
		root->left = newNode(2);
		root->right = newNode(3);
		root->left->left = newNode(4);
		root->left->right = newNode(5);
cout << "Leaf count of the tree is : "<<getLeafCount(root) << endl;return 0;}
```

Manan Jain 211B173