**Student Name:**  Rahul Kumar Singh  **UID:** 19BCS8016

**Branch:** CSE **Section/Group :** 20 (A)

**Semester:** 5th **Date of Performance:** 20 Aug 2020

**Subject Name :** Design & Analysis of Algorithms Lab **Subject Code:** CSP-309

**WORKSHEET 4**

**4.1. Aim/Overview of the practical:**

**Code to Insert and Delete an element at the beginning and at the end in Doubly Linked List .**

**Task to be done/ Which logistics used:**

To implement insert and delete at beginning and end in Doubly Link List

##### 

##### **Algorithm :**

##### **Insertion in doubly linked list at beginning**

step 1: if ptr = null

write overflow

 go to step 9 [end of if]

step 2: set new\_node = ptr

step 3: set ptr = ptr -> next

step 4: set new\_node -> data = val

step 5: set new\_node -> prev = null

step 6: set new\_node -> next = start

step 7: set head -> prev = new\_node

step 8: set head = new\_node

step 9: exit

##### **Insertion in doubly linked list at the end**

step 1: if ptr = null

 write overflow

  go to step 11

 [end of if]

step 2: set new\_node = ptr

step 3: set ptr = ptr -> next

step 4: set new\_node -> data = val

step 5: set new\_node -> next = null

step 6: set temp = start

step 7: repeat step 8 while temp -> next != null

step 8: set temp = temp -> next

[end of loop]

step 9: set temp -> next = new\_node

step 10c: set new\_node -> prev = temp

step 11: exit

##### **Deletion at beginning**

step 1: if head = null

write underflow

goto step 6

step 2: set ptr = head

step 3: set head = head → next

step 4: set head → prev = null

step 5: free ptr

step 6: exit

##### **Deletion in doubly linked list at the end**

step 1: if head = null

write underflow

go to step 7

[end of if]

step 2: set temp = head

step 3: repeat step 4 while temp->next != null

step 4: set temp = temp->next

[end of loop]

step 5: set temp ->prev-> next = null

step 6: free temp

step 7: exit

**Steps for experiment/practical/Code:**

#include<stdio.h>

#include<stdlib.h>

struct node

{

struct node \*prev;

struct node \*next;

int data;

};

struct node \*head;

void insertion\_beginning();

void insertion\_last();

void deletion\_beginning();

void deletion\_last();

void display();

void main ()

{

int choice =0;

while(choice != 6)

{

printf("\n1.Insert at begining\n2.Insert at last\n3.Delete at Beginning\n4.Delete at last\n5.Display\n6.Exit\n");

printf("\nEnter your Choice:\n");

scanf("\n%d",&choice);

switch(choice)

{

case 1:

insertion\_beginning();

break;

case 2:

insertion\_last();

break;

case 3:

deletion\_beginning();

break;

case 4:

deletion\_last();

break;

case 5:

display();

break;

case 6:

exit(0);

break;

default:

printf("Please enter correct choice..");

}

}

}

void insertion\_beginning()

{

struct node \*ptr;

int item;

ptr = (struct node \*)malloc(sizeof(struct node));

if(ptr == NULL)

{

printf("\nLISTFULL");

}

else

{

printf("\nEnter data to be inserted");

scanf("%d",&item);

if(head==NULL)

{

ptr->next = NULL;

ptr->prev=NULL;

ptr->data=item;

head=ptr;

}

else

{

ptr->data=item;

ptr->prev=NULL;

ptr->next = head;

head->prev=ptr;

head=ptr;

}

printf("\nNode inserted\n");

}

}

void insertion\_last()

{

struct node \*ptr,\*temp;

int item;

ptr = (struct node \*) malloc(sizeof(struct node));

if(ptr == NULL)

{

printf("\nList Full");

}

else

{

printf("\nEnter data at last");

scanf("%d",&item);

ptr->data=item;

if(head == NULL)

{

ptr->next = NULL;

ptr->prev = NULL;

head = ptr;

}

else

{

temp = head;

while(temp->next!=NULL)

{

temp = temp->next;

}

temp->next = ptr;

ptr ->prev=temp;

ptr->next = NULL;

}

}

printf("\nnode inserted\n");

}

void deletion\_beginning()

{

struct node \*ptr;

if(head == NULL)

{

printf("\n List Empty");

}

else if(head->next == NULL)

{

head = NULL;

free(head);

printf("\nnode deleted sucessfully\n");

}

else

{

ptr = head;

head = head -> next;

head -> prev = NULL;

free(ptr);

printf("\nnode deleted\n");

}

}

void deletion\_last()

{

struct node \*ptr;

if(head == NULL)

{

printf("\n List Empty");

}

else if(head->next == NULL)

{

head = NULL;

free(head);

printf("\nnode deleted\n");

}

else

{

ptr = head;

if(ptr->next != NULL)

{

ptr = ptr -> next;

}

ptr -> prev -> next = NULL;

free(ptr);

printf("\nnode deleted\n");

}

}

void display()

{

struct node \*ptr;

printf("\nvalues in list are.\n");

ptr = head;

while(ptr != NULL)

{

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

ptr=ptr->next;

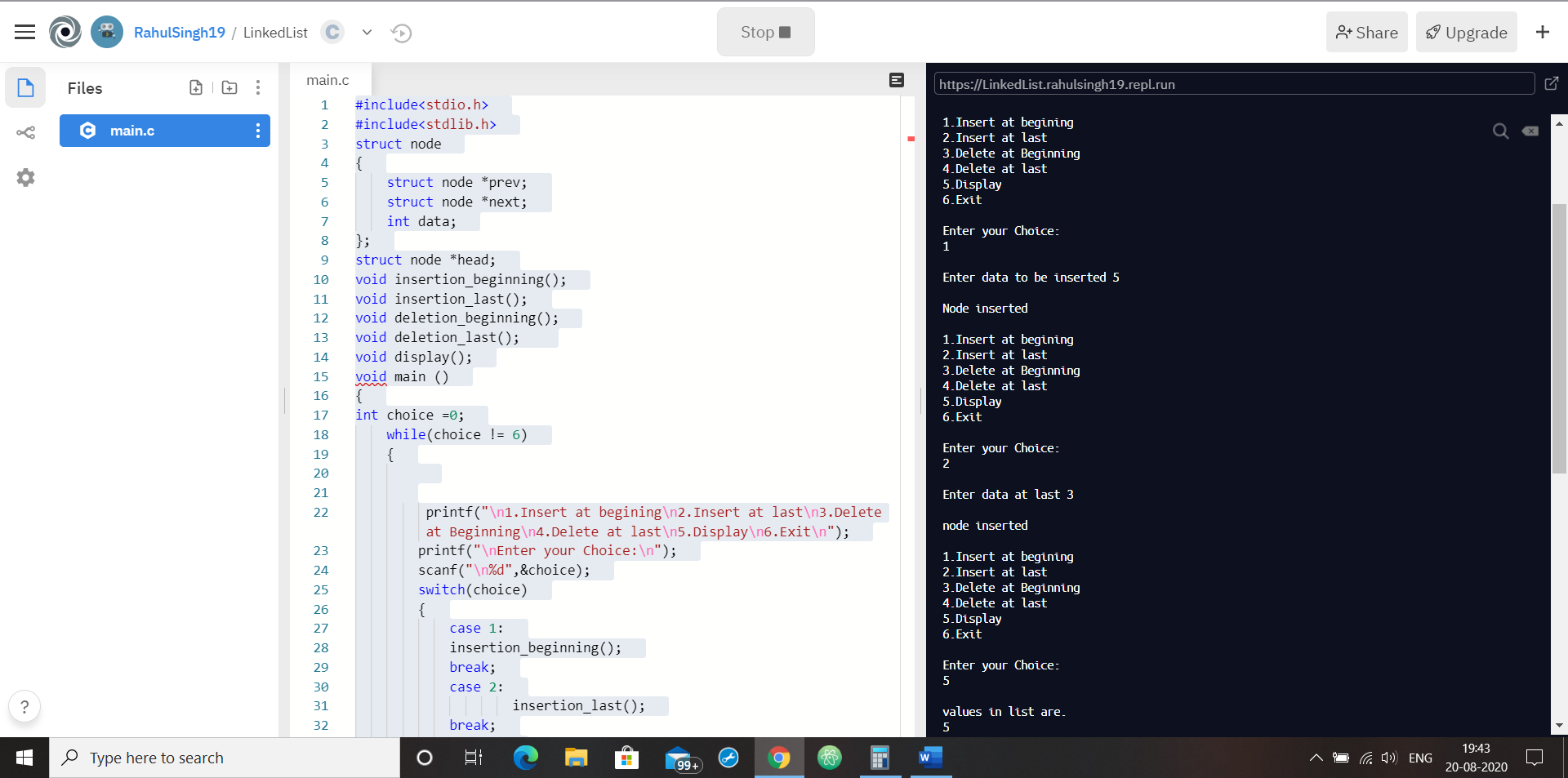
}

}

**Observations/Discussions/ Complexity Analysis:**

Time complexity is O(n)

**Result/Output/Writing Summary:**



**Learning outcomes (What I have learnt):**

1. It will take O(n) time complexity.
2. Learn concept of doubly link list.
3. Learn that how to implement insertion deletion at various positions.

**4.2. Aim/Overview of the practical:**

##### Code to push & pop and check Isempty, Isfull, and Return top element in stacks

**Task to be done/ Which logistics used:**

To implement push , pop , peek operation using stack

**Algorithm :**

**Push :**

if stack is full

     return null

  endif

   top ← top + 1

  stack[top] ← data

end procedure

**Pop :**

 if stack is empty

     return null

  endif

     data ← stack[top]

  top ← top - 1

  return data

end procedure

**Peek :**

begin procedure peek

  return stack[top]

end procedure

**Steps for experiment/practical/Code:**

#include <iostream>

using namespace std;

int stack[100], n=100, top=-1 ;

void push(int val) {

if(top>=n-1)

cout<<"IS FULL"<<endl;

else {

top++;

stack[top]=val;

}

}

void pop() {

if(top<=-1)

cout<<"IS EMPTY"<<endl;

else {

cout<<"POP ELEMENT IS: "<< stack[top] <<endl;

top--;

}

}

void peek() {

if(top<=-1)

cout<<"STACK UNDERFLOW \n"<<endl;

else {

cout<<" TOP ELEMENT IS: "<< stack[top] <<endl;

}

}

int main() {

int ch, val;

cout<<"1) PUSH"<<endl;

cout<<"2) POP"<<endl;

cout<<"3) PEEK"<<endl;

do {

cout<<"ENTER YOUR CHOICE: ";

cin>>ch;

switch(ch) {

case 1: {

cout<<"ENTER VALUE FOR PUSH: ";

cin>>val;

push(val);

cout<<"VALUE IS ADDED \n"<<endl;

break;

}

case 2: {

pop();

cout<<"\n";

break;

}

case 3: {

peek();

cout<<"\n";

break;

}

default: {

cout<<"YOUR CHOICE IS NOT VALID"<<endl;

}

}

}while(ch!=5);

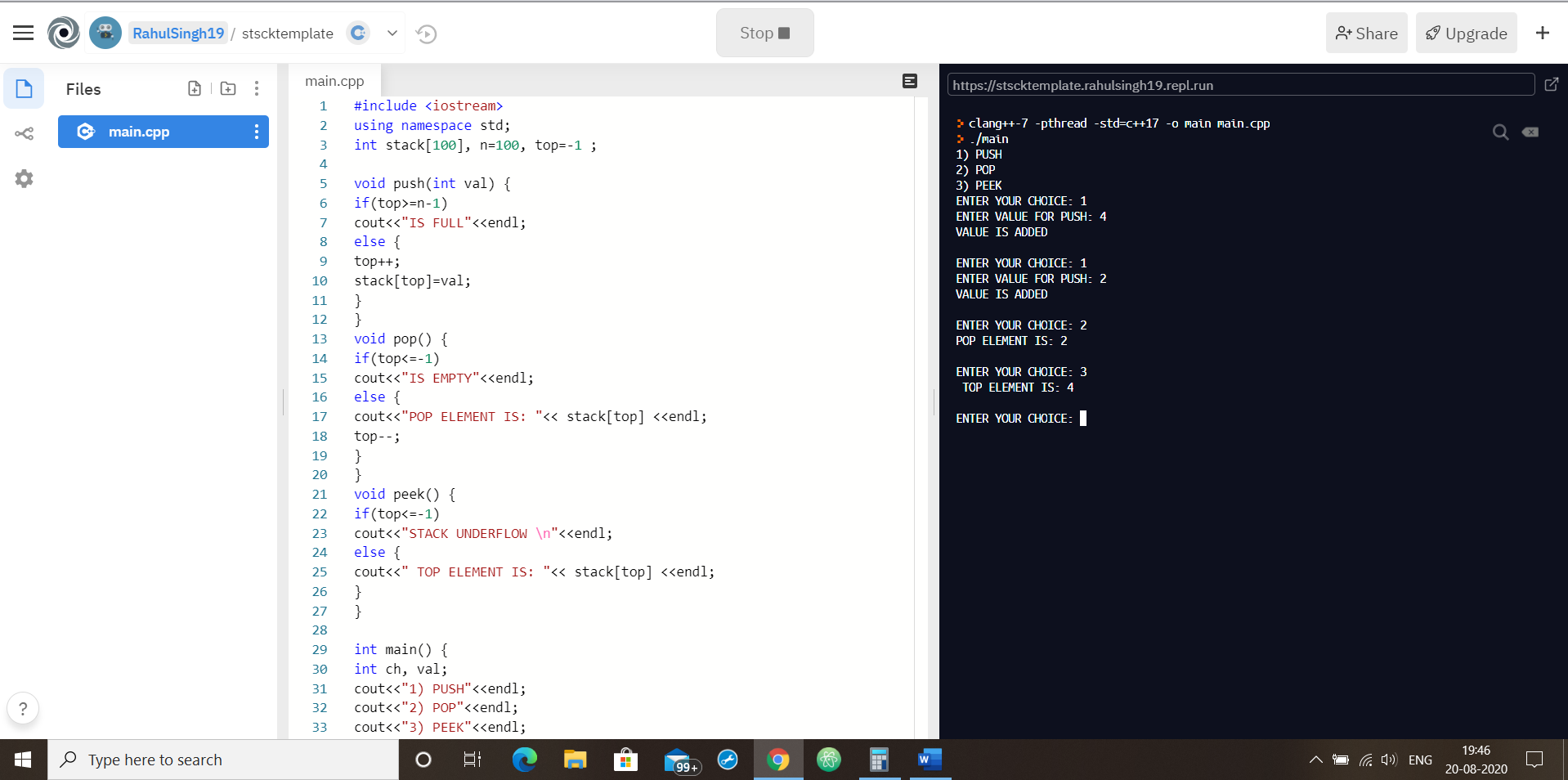
return 0;

}

**Observations/Discussions/ Complexity Analysis:**

Time complexity is O(1)

**Result/Output/Writing Summary:**



**Learning outcomes (What I have learnt):**

1. It will take O(1) time complexity.
2. Learn concept of stack implementation.
3. Learn that how to implement push pop peek operations using stack.