**Worksheet - 4**

**Student Name:** Pratham Mittal **UID:** 20BCS1040

**Branch:** BE CSE **Section/Group:** 20BCS\_WM\_707-B

**Semester:** 5 **Date of Performance:** 03/09/2022

**Subject Name:** Design & Analysis of Algorithm **Subject Code:** 20CSP-312

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

(I) Code for inserting and removing elements at the start and end of a doubly and circular linked list.

(II) Using templates, write code to push and pop elements, check Isempty and Isfull, and return the top element in stacks

**2. Code:**

**(I) Doubly Linked List**

#include <bits/stdc++.h>

using namespace std;

struct node {

int data;

node \*pre;

node \*next;

node() { node(0); }

node(int data) {

this->data = data;

next = pre = NULL;

}

};

void insertAtBegin(node\*\* head, int d) {

node\* n = new node(d);

n->next = \*head;

if(\*head != NULL) {

(\*head)->pre = n;

}

\*head = n;

}

void insertAtEnd(node\*\* head, int d) {

node\* n = new node(d);

node\* t = \*head;

if(\*head == NULL) {

\*head = n;

return;

}

while(t->next != NULL)

t = t->next;

t->next = n;

n->pre = t;

}

void deleteAtBegin(node\*\* head) {

if(\*head == NULL) {

return;

}

\*head = (\*head)->next;

free((\*head)->pre);

(\*head)->pre = NULL;

}

void deleteAtEnd(node\*\* head) {

if(\*head == NULL) {

return;

}

node\* t = \*head;

while(t->next != NULL)

t = t->next;

t = t->pre;

free(t->next);

t->next = NULL;

}

void display(node\*\* head) {

node\* t = \*head;

cout<<"Current List: ";

while(t != NULL) {

cout<<t->data<<" -> ";

t = t->next;

}

cout<<"NULL\n";

}

int main(){

int c;

cout<<"Staring element number: ";

cin>>c;

node\* head = new node(c);

c=0;

while(c != 6) {

cout<<"\n1. Insert at begin\t2. Insert at end\n3. Delete at begin\t4. Delete at end\n5. Display\t\t6. Exit\nEnter: ";

cin>>c;

switch (c) {

case 1:

cin>>c;

insertAtBegin(&head, c);

break;

case 2:

cin>>c;

insertAtEnd(&head, c);

break;

case 3:

deleteAtBegin(&head);

break;

case 4:

deleteAtEnd(&head);

break;

case 5:

display(&head);

break;

case 6:

cout<<"Exiting...";

return 0;

default:

cout<<"\*\*\*\*\*\*\*INVALID INPUT\*\*\*\*\*\*\*";

}

}

return 0;

}

**(II) Stack using template**

#include <iostream>

using namespace std;

#define SIZE 5

template <class T>

class Stack {

private:

int t;

T st[SIZE];

public:

Stack() { t = -1; }

void push(T k) {

if (isfull()) {

cout<<"Overflow\n";

return;

}

cout<<k<<" is Inserted\n";

t++;

st[t] = k;

}

T pop() {

if(isempty()) {

cout<<"Underflow\n";

} else {

T pe = st[t];

t--;

return pe;

}

}

T top() {

return st[t];

}

void print() {

if(isempty()) {

cout<<"Empty Stack\n";

return;

}

cout<<"Stack: ";

for(int i=0; i<=t; i++) {

cout<<st[i]<<" ";

}

}

bool isfull() {

if(t == SIZE-1) {

return 1;

}

return 0;

}

bool isempty() {

if(t == -1) {

return 1;

}

return 0;

}

};

int main(){

int c = 0;

Stack<int> s;

while(c != 7) {

cout<<"\n\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\n1. push\t4. isempty\n2. pop\t5. isfull\n3. top\t6.print\n7. Exit\tEnter: ";

cin>>c;

switch (c) {

case 1:

cin>>c;

s.push(c);

break;

case 2:

s.pop();

break;

case 3:

cout<<"Top element: "<<s.top();

break;

case 4:

cout<<s.isempty();

break;

case 5:

cout<<s.isfull();

break;

case 6:

s.print();

break;

case 7:

cout<<"Exiting...";

return 0;

default:

cout<<"\*\*\*\*\*\*\*INVALID INPUT\*\*\*\*\*\*\*";

}

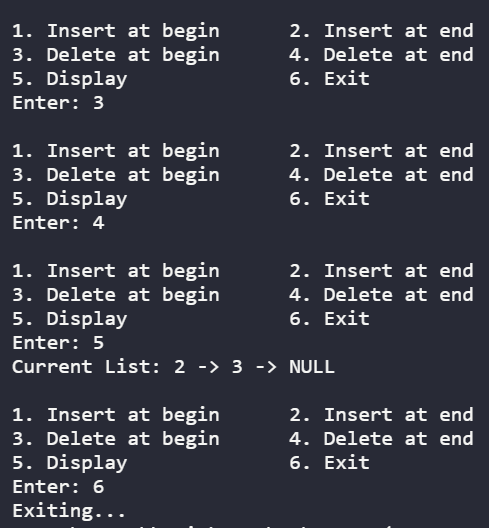
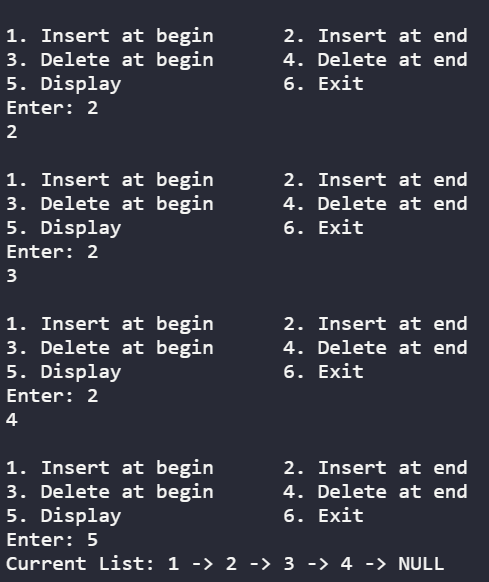
}

return 0;

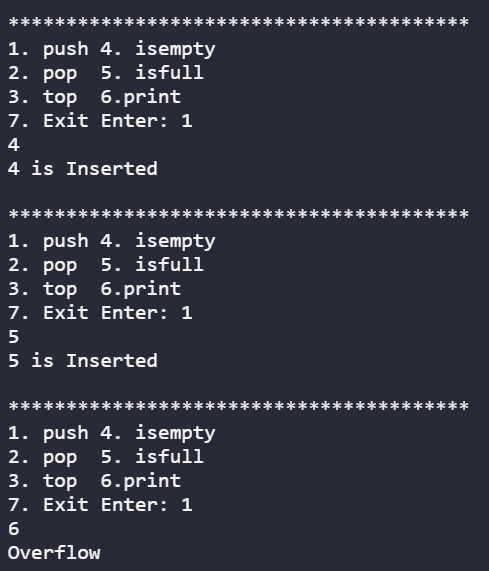
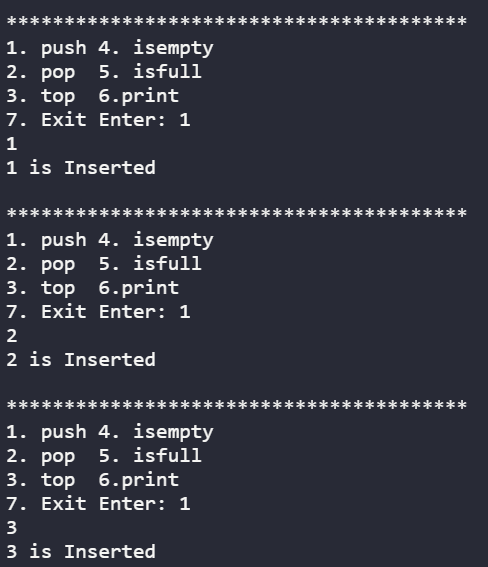
}

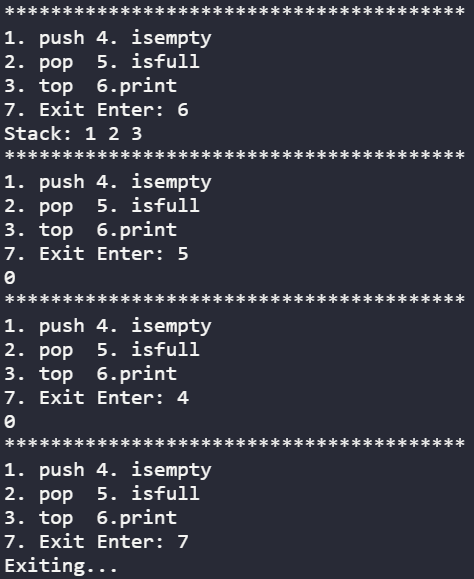
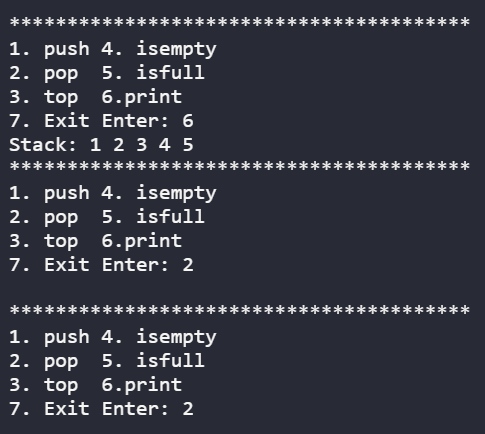
**5. Result/Output:**

**(I) Doubly Linked List**



**(II) Stack using template**





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

1. Understood the concept of insertion and deletion at the beginning and end of the doubly linked list.
2. Understood the concept of templates and implementing functions like isempty , isfull and top on stack.

**Evaluation Grid (To be created as per the SOP and Assessment guidelines by the faculty):**

|  |  |  |  |
| --- | --- | --- | --- |
| Sr. No. | Parameters | Marks Obtained | Maximum Marks |
| 1. |  |  |  |
| 2. |  |  |  |
| 3. |  |  |  |