#ifndef STACK\_H

#define STACK\_H

#include <iostream>

using namespace std;

template <typename T>

class Node{

public:

Node\* next;

T value;

Node(T data){

value = data;

next = nullptr;

}

};

template <typename T>

class Stack{

public:

Node<T>\*tos; //pointer which points to top of the stack

Stack(); //Constructor

bool isEmpty(); //Function to check if stack is empty

Node<T>\* makeNode(char val); //Function to make a new node

void push(char val); //Function to push value to the stack

char pop(); //Popping the value from stack

char peek(); //Function to peek at the top of stack

void display(); //Function to display stack elements

};

//\*\*\*\*\*\*\*\*\*\*\*\*\*Constructor definition\*\*\*\*\*\*\*\*\*\*\*\*\*\*//

template <typename T>

Stack<T>::Stack() {

tos = nullptr;

}

//\*\*\*\*\*\*\*\*\*\*\*\*\*Function to check if the stack is empty\*\*\*\*\*\*\*\*\*\*\*\*\*//

template <typename T>

bool Stack<T>::isEmpty() {

return(tos == nullptr);

}

//\*\*\*\*\*\*\*\*\*\*\*\*\*Function to create a new node\*\*\*\*\*\*\*\*\*\*\*\*\*//

template <typename T>

Node<T>\* Stack<T>::makeNode(char val) {

Node<T>\* temp = new Node<T>(val);

if(temp == nullptr) {

cout<<"Full!"<<endl;

return nullptr;

}

return temp;

}

//\*\*\*\*\*\*\*\*\*\*\*\*\*Function to push a value onto the stack\*\*\*\*\*\*\*\*\*\*\*\*\*//

template <typename T>

void Stack<T>::push(char val){

Node<T>\* new\_node = makeNode(val);

new\_node->next = tos;

tos = new\_node;

}

//\*\*\*\*\*\*\*\*\*\*\*\*\*Function to pop a value from the stack\*\*\*\*\*\*\*\*\*\*\*\*\*//

template <typename T>

char Stack<T>::pop(){

if (tos == nullptr) {

return '\0';

}

else{

Node<T>\* temp = tos;

char popped = temp->value;

tos = tos->next;

delete (temp);

return popped;

}

}

//\*\*\*\*\*\*\*\*\*\*\*\*\*Function to peek at the top of the stack\*\*\*\*\*\*\*\*\*\*\*\*\*//

template <typename T>

char Stack<T>::peek(){

if(tos == nullptr){

return '\0';

}

return tos->value;

}

//\*\*\*\*\*\*\*\*\*\*\*\*\*Function to display the stack elements\*\*\*\*\*\*\*\*\*\*\*\*\*//

template <typename T>

void Stack<T>::display() {

Node<T>\* temp = tos;

while (temp != nullptr) {

cout<<temp->value<<" ";

temp = temp->next;

}

}

#endif