## **Stack Implementation using Linked List**

**StackLinkedList.h (header file)**

//

// Created by f12r on ১৫/৮/২১.

//

#ifndef ASSIGNMENT\_2\_STACKLINKEDLIST\_H

#define ASSIGNMENT\_2\_STACKLINKEDLIST\_H

class FullStack {

};

class EmptyStack {

};

template<class T>

class StackLinkedList {

struct NodeType {

T data;

NodeType \*link;

};

private:

NodeType \*topPtr;

public:

StackLinkedList();

~StackLinkedList();

void push(T);

void pop(T &);

bool isEmpty();

T top();

void PrintStack();

};

#endif //ASSIGNMENT\_2\_STACKLINKEDLIST\_H

**StackLinkedList.cpp (definition file)**

//

// Created by f12r on ১৫/৮/২১.

//

#include "StackLinkedList.h"

#include <iostream>

using namespace std;

template <class T>

StackLinkedList<T>::StackLinkedList()

{

topPtr = NULL;

}

template <class T>

bool StackLinkedList<T>::isEmpty()

{

return topPtr == NULL;

}

template <class T>

void StackLinkedList<T>::push(T item)

{

NodeType \*location = new NodeType;

location->data = item;

location->link = topPtr;

topPtr = location;

}

template <class T>

void StackLinkedList<T>::pop(T &popItem)

{

NodeType \*temp;

temp = topPtr;

if (isEmpty())

{

throw EmptyStack();

}

else

{

popItem = topPtr->data;

topPtr = topPtr->link;

delete temp;

}

}

template <class T>

T StackLinkedList<T>::top()

{

if (isEmpty())

{

throw EmptyStack();

}

return topPtr->data;

}

template <class T>

void StackLinkedList<T>::PrintStack()

{

NodeType \*temp;

temp = topPtr;

if (topPtr == NULL)

{

throw EmptyStack();

}

else

while (temp != NULL)

{

cout << temp->data << " ";

temp = temp->link;

}

}

template <class T>

StackLinkedList<T>::~StackLinkedList()

{

NodeType \*tempPtr;

while (topPtr != NULL)

{

tempPtr = topPtr;

topPtr = topPtr->link;

delete tempPtr;

}

}

**main.cpp (driver file)**

#include "StackLinkedList.h"

#include "StackLinkedList.cpp"

#include <iostream>

using namespace std;

int main() {

StackLinkedList<int> list;

// check the stack is empty or not

if (list.isEmpty())

{

cout<<"Stack is empty"<<endl;

}else{

cout<<"Stack is not empty"<<endl;

}

// push items

cout<<"Insert 5 items: ";

for (int i = 0; i < 5; i++)

{

int x;

cin>>x;

list.push(x);

}

// display items

cout<<"Display all the items: ";

list.PrintStack();

cout<<endl;

cout<<"Remove an item and the removed item is : ";

// pop items

int x;

list.pop(x);

cout<<x<<endl;

if (list.isEmpty())

{

cout<<"Stack is empty"<<endl;

}else{

cout<<"Stack is not empty"<<endl;

}

// display items

cout<<"Display all the item: ";

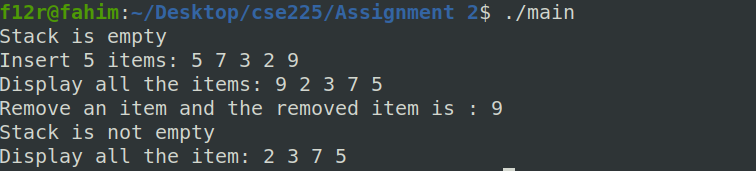
list.PrintStack();

cout<<endl;

return 0;

}

**OUTPUT**

****