User Id : weizhe.goh@digipen.edu | Started : 2020.07.23 16:48:32 | Score : 100%

DigiPen

Exceptions

Assignment

Rules

Read carefully and check all rules you agree with:

- ■ Your code must represent your own individual work. Cheating of any kind (copying someone else's work, allowing others to copy your work, collaborating, etc.) will not be tolerated and will be dealt with SEVERELY.
- ■ Each exercise has description which must be strictly followed.
- ■ All programs must pass all tests in the main function (when given) to get the final grade. You are not allowed to make any change in the main function in this case.
- ■ Keep the code proper formatted (correct indentation, max line width is 40 characters).
- ■ Every week the instructor is available **during the lab time** to discuss following matters:
  - your disagreement with rule in this card,
  - misunderstanding of the current assignment specs,
  - solution for given problems.

Specs

- Implement a new class named Stack based on the List class developed in the previous assignment. Your implementation must comply with following requirements:

- ■ The Stack class must be implemented as a **wrapper class** over the List, so your new class and its methods provide a new way of data management, but all the actual work is done by the old List class underneath.
- ■ Both classes Stack and List must be templates, so can be used to create containers of different data types.
- ■ An object of the List class is the only member data of the Stack class. (Remove all members from the class List that are not used in the main function. Implement members that are missing when necessary.)
- ■ As a stack, your new class must implement push, pop, isEmpty member functions.
- ■ The pop member must throw an exception when the stack is empty.
- ■ Make tests for the Stack members in the main function for integer and float data types.
- ■ In the main function catch and handle the exception from the pop member function call.
- ■ Make your own output from the main function for testing purpose. Any non empty output will be evaluated as correct by this service. After submission it will be re-evaluated by instructor.

Code

★ Code 1 ☑

Run

```
#include <iostream>

template <typename T>
struct Node
{
    T data;
    Node* next;
};

template <typename T>
class List
{
private:
    Node<T>* head;

public:
    List() :head(nullptr) {}
```

Survey

- What is approximate number of hours you spent implementing this assignment?

```
10 hours
```

- Indicate the specific portions of the assignment that gave you the most trouble

```
Popping sequence and definition of
popping
```

```
    Node<T> *getHead()
    {
        return this->head;
    }

    Node<T>* setHead(Node <T>* node)
    {
        return this->head = node;
    }

    void push_front(T value)
    {
        Node<T>* newNode = new Node<T>;

        newNode->data = value;
        newNode->next = head;

        head = newNode;
    }
};

template <typename T>
class Stack
{
    List<T> list;

public:
    void push(T value)
    {
        list.push_front(value);
    }

    T pop() noexcept (false)
    {
        if (isEmpty() == true)
        {
            throw "Empty Stack";
        }

        T deleteValue = 0;

        deleteValue =
        list.getHead()->data;

        Node<T> *deleteNode = nullptr;
        Node<T> *current =
        list.getHead();

        deleteNode = current;

        list.setHead(current->next);

        delete deleteNode;
        return deleteValue;
    }

    bool isEmpty() noexcept
    {
        if (list.getHead() != nullptr)
            return false;
        else
            return true;
    }
};

int main(void)
```

```
{
        Stack<int> s1;

    s1.push(3);
    s1.push(2);
    s1.push(1);

    try
    {
        std::cout << s1.pop() << "->";
        std::cout << s1.pop() << "->";
        std::cout << s1.pop() << "->";
        std::cout << s1.pop() << "->";
    }
    catch(const char * str)
    {
        std::cout << str << std::endl;
    }

    Stack<float> s2;

    s2.push(10.3f);
    s2.push(10.2f);
    s2.push(10.1f);

    try
    {
        std::cout << s2.pop() << "->";
        std::cout << s2.pop() << "->";
        std::cout << s2.pop() << "->";
        std::cout << s2.pop() << "->";
    }
    catch (const char* str)
    {
        std::cout << str << std::endl;
    }

        return 0;
}
```

```
1->2->3->Empty Stack
10.1->10.2->10.3->Empty Stack
```

By signing this document you fully agree that all information provided therein is complete and true in all respects.	Responder sign: