# ENSC 251 Lab Assignment 4

## Lab Assignment Overview:

In the ENSC 251 course, you will work on four lab assignments using the skills learned throughout this course, i.e., Object-Oriented Programming (OOP) with C++. Each lab assignment weighs 10 marks. All lab assignments will be carried out and evaluated in pairs (i.e., two students per group) - I hope by now all of you have already found a partner. Some general grading logistics have been posted on our course website: https://canvas.sfu.ca/courses/70216/pages/lab-logistics.

Please make sure your code can compile and run correctly on the lab computer. If your code cannot compile, then your group can get at most 5 marks. If your code can compile, but cannot run, then your group can get at most 6 marks.

You can divide the work roughly based on the listed tasks; the overall marks for major tasks are also listed for your guidance, but not the detailed marks for every single item. Please include a simple document in each lab (include a txt or pdf file in the .zip file for submission) indicating which parts each member finished. There is some negative marking if you don't complete the listed tasks; details are not listed. Basically, the detailed grading scheme (except the overall marks for major tasks) for each lab will NOT be released before your lab grading is done. Think about you are in a real interview, nobody will tell you what the detailed grading schemes are.

## Lab Assignment 4:

In Lecture 8 and 9, we have introduced the Stack data structure as below. In the lecture (including code tutorial video) and textbook, some code implementations have already been given.

```
struct StackFrame {
    char data;
    StackFrame *link;
};
typedef StackFrame* StackFramePtr;


class Stack {
public:
    Stack( );
    void push(char the_symbol);
    char pop( );
    bool empty( ) const;
private:
    StackFramePtr top;
};
```

In this lab assignment, you will implement a complete version of this Stack class, and generalize it using ***template*** (Lecture 14) to support different types of data items instead of simply char type.

**Specifically, in lab assignment 4, you will implement and test the Stack class by building the following items.**

1) **[6 marks, note: non-template implementation only counts for 3 marks] Implement the *Stack* class.** Implement the following **member functions**:
    a. [0.5 mark] Constructor functions and peek() function. The peek function (essentially the get function) returns the top pointer of the stack.
    b. [1.5 marks] ***push***, ***pop***, and ***empty*** functions, as introduced in the lecture.
    c. [1.5 marks] The dynamic big three for this class since it has a pointer member variable, i.e., destructor, copy constructor, and overloaded assignment operator. I have shown you some of the code in the recorded code tutorial video as well.
    d. [2.5 marks] The ***char reverse()*** function. This reverse function reverses the order of the stack. For example, if the stack originally has items as (from top to bottom) 1, 2, 3, after calling this reverse function, the stack now should have items as (from top to bottom) 3, 2, 1. This function returns the new top value. If it's an empty stack, print an error message and exit the program. Analyze the algorithm time complexity of your reverse function using big-O and document it in the code comments.
    **Important note:** when you implement this reverse function, you are NOT allowed to use any additional data structures. For example, you are NOT allowed to use another stack, queue, array, vector, linked list, etc., to hold the new stack. Moreover, you are NOT allowed to use loop constructs like for, while, etc. Instead, you should use a recursive algorithm to implement this. You are allowed to create more helper functions. In fact, this comes from a real interview question.
    e. Whenever you allocate a new memory, catch the *bad_alloc* exception.
2) **Generalize the Stack class (including all its member functions) using *template*, such that it can support different types of data items, including, char, int, float, double, etc.**
    a. **Note:** To get all 6 marks of item 1) Stack class, it has to be implemented using template; otherwise, get at most 3 marks.
3) **[1 mark] Generalize the struct StackFrame and StackFramePtr types using *template*.**
4) **[1 mark] Implement a helper template function *void printStack (Stack s);* to print out the stack from top to bottom.**
    a. Note that after calling this function, your stack should not change. And this function is an ordinary function, not a member function of the Stack class.
5) **[2 marks] Test your *generalized Stack class* in main.cpp:**
    a. Create ***Stack<int>*** object sint1 and push the following items into sint1: {1, 2, 3, 4, 5, 6}. Create ***Stack<int>*** object sint2, and assign sint1 to sint2, i.e., sint2 = sint1. Reverse sint1. Print out the entire stack of sint1 and sint2.
    b. Create ***Stack<double>*** object sdouble1 and push the following items into sdouble1: {1.5, 2.5, 3.5, 4.5, 5.5, 6.5}. Print out the entire stack of sdouble1. Create ***Stack<double>*** object sdouble2 using copy constructor, i.e., sdouble2(sdouble1). Reverse sdouble2 and print out the top of sdouble2.

c. Ask user to input a string, print out a palindrome based on it using ***Stack<char> and its reverse function.*** For example, if a user inputs "an", you should print "anna". If a user inputs "no", you should print "noon".

6) **Additional requirements**
   a. Make sure the *generalized Stack* class is implemented as abstract data type (ADT) in stack.hpp and stack.cpp, respectively.
   b. For the header filer, use the #ifndef macro to avoid double include.
   c. Adapt your own Makefile based on all prior labs.
   d. In addition to the actual coding implementation, you need to provide good commenting, naming, and other good coding styles; all these count in your lab assignment marking.

**Note: For grading logistics and remote machine access, please refer to the course website. If you have any questions, please post them on the discussion board.**

## Assignment Submission:

Your lab assignment 4 will be submitted electronically through Canvas. You will need to submit a single lab4.zip file. Failure to comply with this format will result in a **ZERO** score for this assignment. To zip your files in Linux,

1. Go to your lab4 directory
2. make clean //**make sure you clean your files**
3. cd .. //go one level up
4. zip -r lab4.zip lab //zip all your lab4 files into a single lab4.zip

## Submission Deadline:

Your lab assignment 4 is due at **11:59:59pm on Friday, Jul 15th, 2022**. You need to meet the deadline: every 10 minutes late for submission, you lose 1 mark; that is, 100 minutes late, you will get zero for this lab.

## Lab Demonstration:

You will have to demo your lab assignment 4 to your TA in the following lab sessions that you enrolled on **Monday (Jul 18th, 2022**). Only code from your Canvas submission is allowed in the lab demo. Each student group has around 10 minutes to explain your code to the TA. If you fail to do the demo (without a medical note), or if it is determined that you do not understand the code being evaluated, you will be awarded zero on this lab assignment. Also please show up in the demo day on time (TA will send out your scheduled time), otherwise you will lose 0.5 mark.