6.1 Implementing a Stack

Due 14 Oct 2019 by 23:59 **Points** 0 **Submitting** an external tool

Write a program that implements a stack of integers, and exercises the stack based on commands read from *cin*. To do this, write a class called *Stack* with **exactly** the following members:

```
class Stack {
    public:
        bool isEmpty();
        // returns true if stack has no elements stored

        int top();
        // returns element from top of the stack
        // throws runtime_error("stack is empty")

        int pop();
        // returns element from top of the stack and removes it
        // throws runtime_error("stack is empty")

        void push(int);
        // puts a new element on top of the stack

    private:
        vector<int> elements;
};
```

The program should read commands from *cin* until either end-of-file is reached or the command **end** is entered. (You can manually test for end-of-file by entering CTRL-D.) Each time the program expects a new command it should first print a prompt: "**stack>** "
Your program should catch all errors that happen and continue execution until the **end** command or end-of-file.

In case the command **push** is read, the program should read an integer value from *cin* and push it onto the stack. In case **top** is read, the program should print the top integer of the stack to *cout*. In case **pop** is read, the program should print the top integer of the stack to *cout*, and remove it from the stack. In case the command **list** is read, the program should print all values currently on the stack, ***without modifying the stack***. (Exact format see below.)

Your program should check whether a "number" to be pushed is actually a number. If not, print an error message (see below), and reset *cin* such that it will again accept commands. (See Section

7.6 of the zyBook.) Also, your program should ignore all characters behind a number to be pushed that are on the same input line (example see below).

An example of a correct execution of this program is shown below:

```
stack> push 5
stack> pop
5
stack> pop
error: stack is empty
stack> push 6
stack> push 4bb
stack> push foo
error: not a number
stack> list
[4,6]
stack> list
[4,6]
stack> top
4
stack> hello
error: invalid command
stack> end
```

You may want to use the **compare() function (http://www.cplusplus.com/reference/string/string/compare/)** on a string to check which command has been entered.

Use of arrays, a built-in stack class, or container classes from std:: other than vector, is not allowed.

This tool needs to be loaded in a new browser window

Load 6.1 Implementing a Stack in a new window