5.3 Implementing a Stack (optional, gives bonus)

Due 6 Oct 2020 by 23:59 **Points** 0 **Submitting** an external tool

Implement a class *Stack* (base type *int*) that has **exactly** the following interface. You are given the main program in which the class definition is already present. The main program will exercise your stack with a series of commands.
All you need to do is implement the function members of the class.

```
class Stack {
    public:
        Stack();

        bool isEmpty() const;
        // returns true if stack has no elements stored

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

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

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

        std::string toString() const;
        // returns the contents of the stack in the format [top,next,...,bottom]
        // e.g.: push(1), push(2), toString() -> [2,1]

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

The main program reads commands from *cin* until either end-of-file is reached or the command **end** is entered.

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
```

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

Here is the main program into which you should include your implementation of class Stack:
stack-skeleton.cpp (https://canvas.vu.nl/courses/50259/files/2755962/download?download_frd=1)

This tool was successfully loaded in a new browser window. Reload the page to access the tool again.