Lab 3: Exceptions

1. Introduction

As we saw a demo in the lecture today, C++ language provides programmers with a way to deal with exception events which could happen while the program is running. In this lab, you will gain experiences throwing exceptions and handling exceptions thrown.

The code skeleton is under: **/comp/15/files/l3**

1. Log in to the CS homework server.
2. Move to your **comp15** directory that you created in Lab 1.
3. Create a directory named **lab3** under the comp15 directory.
4. Move to the lab directory.
5. Copy the code skeleton to the current working directory.
6. Leverage the features of Git to manage your progress toward writing the program.

The code skeleton provides you with three files: **test.cpp**, **Array.hpp**, **Array.cpp**. First, take a look at the files to find the internal structure of the **Array** class. Note that the copy constructor and assignment operator are commented out so that this lab can be completed within our lab time, but you are encouraged to work on them later on. **test.cpp** provides you with the **add()** function and one test case that your implementation is supposed to pass.

Note: To use exceptions, you need to include: **`stdexcept`**
Note: To use `INT_MAX` and `INT_MIN`, you need to include: **`climits`**

2. Requirements

1. Implement the **Array** class.
    a. The constructor: The capacity of this array must be initialized to be **5**.
    b. The add() method: When there is no available spot for the given data, expand this array by **doubling** its capacity. i.e. After the first expansion, the capacity is 10. After the second expansion, the capacity is 20. and so on.
    c. The at() method: If the given index is not appropriate for this array, then throw a **`std::range_error`** with a message whose format can be found at Line:35 and Line:47 in **test.cpp**
2. Implement the **add()** function in **test.cpp**
    a. The add() function takes two integers.
    b. If adding the two integers results in integer overflow, then throw a **`std::overflow_error`** with the message: **OVERFLOW**

   c. If adding the two integers results in integer underflow, then throw a **`std::underflow_error`** with the message: **UNDERFLOW**
   (Note: Be aware that there exists the word "floating point underflow", too.)
3. Complete the test case in **test.cpp**
   a. There are 6 places, in the main function, that need to be completed. Each of them is marked with **//===(X)**, where X is a number.
4. Pass the test case written in **test.cpp** with no memory leaks and no memory errors.

3. README

Create the README file that includes the following categories with appropriate section headers.

1. **Name**: Your name.
2. **Date**: The last updated date.
3. **Summary**: A brief summary of the lab.
4. **Files**: A list of files that are necessary to build and test the program.
5. **Instructions**: A sequence of commands to compile and test the program. Note that you are expected to report procedures without using the make command.
6. **References**: A list of citations to information used to complete the lab.

4. Submission

Submit your files listed below using Gradescope.
Files: **test.cpp Array.hpp Array.cpp README**