CSC3002 2023 spring Assignment 3

Due 23:59, March 26, 2023

Problem 1 (Exercise 11.6, Points: 25)

Problem Description

Using the following definitions of `MAX_JUDGES` and `scores` as a starting point:

```
const int MAX_JUDGES = 100;
double scores[MAX_JUDGES];
```

Write a program that reads in gymnastics scores between 0 and 10 from a set of judges and then computes the average of the scores after eliminating both the highest and lowest scores from consideration.

In & Out Statement

Your program should accept input values until the maximum number of judges is reached or the user enters a blank line.
If the number of input values is less than 3, prompt a error message. If a score that is out of range is received, such score will not be counted and a error message will also be prompted. The processing steps are in the function `readScores()`, which you should not modify but are highly recommended to read.
This program supports both text file input and manual input. To input a text file:

```
p1.exe < in/p1_in1.txt // for win user
./p1 < in/p1_in1.txt // for Mac user
```

Note that The text file you prepared should end with a blank line if the number of your input values is less than `MAX_JUDGES`.
To input manually:

```
p1.exe // for win user
./p1 // for Mac user
```

Then input your values following the prompts:

```
Enter a score for each judge in the range 0 to 10.
Enter a blank line to signal the end of the list.
Judge #1:  6
Judge #2:  5
Judge #3:  3.1415
Judge #4:
The average after eliminating 3.14 and 6.00 is 5.00.
```

Please see *out/p1_1.txt, out/p1_2.txt* for output format.

About P1

- Please finish `sumArray`, `findLargest` and `findSmallest` functions in the file *p1GymJudge.cpp*.
- DO NOT modify the `main()` part, which is for testing unit.

Problem 2

Part1:(Exercise 12.4, Points: 15)

Problem Description

Design and implement a class called `IntArray` in *p2intArray.cpp* that implements the following methods:

- A constructor `IntArray(n)` that creates an `IntArray` object with `n` elements, each of which is initialized to 0.
- A destructor that frees any heap storage allocated by the `IntArray`.
- A method `size()` that returns the number of elements in the `IntArray`.
- A method `get(k)` that returns the element at position `k`. If `k` is outside the vector bounds[1], `get` should call `error` (see `void error(std::string msg)` in *lib.cpp* with message:`"Index range out of bounds!"`.
- A method `put(k, value)` that assigns value to the element at position `k` (directly assign, not insertion). As with `get`, the `put()` method should call `error` with the same message if `k` is out of bounds.

Use `int[]` instead of wrapping an exiting `vector<int>`.

In & Out Statement

Two integers should be input to test the functionality of this program:

Position	meaning
0	index of the task
1	size of array

where index of the task = 1 in this part.
This program supports both text file input and manual input.
To input a text file:

```
p2.exe < in/p2_in1.txt // for win user
./p2 < in/p2_in1.txt // for Mac user
```

To input manually:

```
p2.exe // for win user
./p2 // for Mac user
```

For instance, when input:
`1 0`
You are expected to get:

```
array.size():  0
ERROR: Index range out of bounds!
ERROR: Index range out of bounds!
```

Please see *out/p2_ 1.txt* for output format.

Part2:(Exercise 12.5, Points: 5)

Problem Description

You can make the `IntArray` class from the preceding exercise look a little more like traditional arrays by overriding the bracket-selection operator, which has the following prototype:

[1]Here we consider the bound to be >=0 and < `n`

```
int & operator[](int k);
```

Like the `get` and `put` methods, your implementation of `operator[]` should check to make sure that the index `k` is valid. If the index is valid, the `operator[]` method should return the element by reference so that clients can assign a new value to a selection expression.

In & Out Statement

Two integers should be input to test the functionality of this program, with the same meanings of those in Part 1. The index of the task = 2 in this part.
Other test input modes are the same as those in Part1.
Please see *out/p2_ 2.txt* for output format.

Part3:(Exercise 12.6, Points: 5)

Problem Description

Implement deep copying for the `IntArray` class from P2Part1 and P2Part2.
The following functions should be completed in this part:

- A function that copies the data from the src parameter into the current object. All dynamic memory is reallocated to create a "deep copy" in which the current object and the source object.

  ```
  void IntArray::deepCopy(const IntArray & src)
  ```

- A function that implements deep copy via "=" operator.

  ```
  IntArray & IntArray::operator=(const IntArray & src)
  ```

- A function that implements deep copy though the constructor.

  ```
  IntArray::IntArray(const IntArray & src)
  ```

In & Out Statement

Two integers should be input to test the functionality of this program, with the same meanings of those in Part 1. The index of the task = 3 in this part.
Other test input modes are the same as those in Part1.
Please see *out/p2_ 3.txt* for output format.

About P2

- Please complete **all** the functions included in the problem descriptions, other wise your program will be deducted even if the OJ running results go well.
- In all test cases for this problem, size of array `n` is a non-negative integer.
- Pre-test cases on the OJ platform are test case of part1, part2, and part3, respectively.
- DO NOT modify the main() part, which is for testing unit.

About this assignment

- You don't need to modify or submit any other files other than `p1GymJudge.cpp` and `p2intArray.cpp`.
- Submission to OJ platform is required, while submission to Blackboard again is not necessary.

- You can only view your pre-test scores and such scores do NOT equal to your final score. Pre-test cases take up 20-40% cases of the final test (also known as formal test) cases in this assignment. The formal test will be conducted after the assignment. Your final score will ONLY depend on the codes in your latest submission.
- If you are late, 5 points will be deducted from each missing problem immediately after 00:00, and 5 points more every hour afterward until no more points can be deducted.