CSC 3002 (Fall 2021) **Assignment 3**

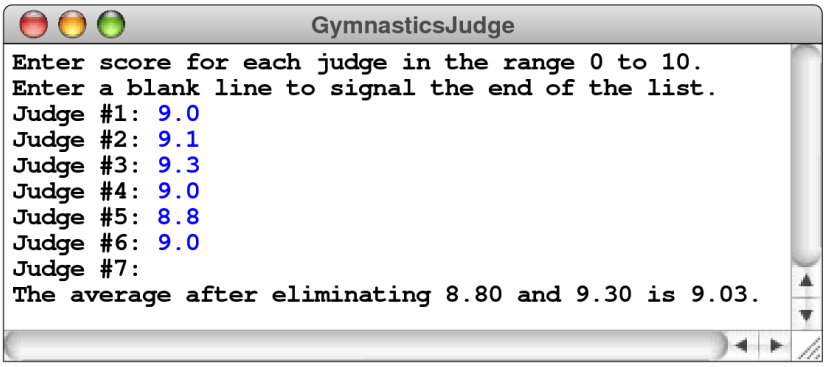
Problem 1

Exercise 11.6:

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. Your program should accept input values until the maximum number of judges is reached or the user enters a blank line. A sample run of this program might look like this:



Requirments & Hints:

Please fill in the **TODO** part of **sumArray**, **findLargest** and **findSmallest** functions in *GymnasticsJudge.cpp*. You can define your own functions in the codes if necessary, but you need to follow the provided code framework.

Problem 2

(a) Exercise 12.4:

Design and implement a class called **IntArray** 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, **get** should call **error** with an appropriate message.
- A method **put(k, value)** that assigns **value** to the element at position **k**. As with **get**, the **put** method should call **error** if **k** is out of bounds.

Requirments & Hints:

Please fill in the **TODO** part of **constructor**, **destructor**, **size**, **get** and **put** functions in *intarray.cpp*. You can define your own functions in the codes if necessary, but you need to follow the provided code framework.

(b) Exercise 12.5:

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:

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 it is, the **operator[]** method should return the element by reference so that clients can assign a new value to a selection expression.

Requirments & Hints:

Please fill in the **TODO** part of **operator[]** function in *intarray.cpp*. You can define your own functions in the codes if necessary, but you need to follow the provided code framework.

(c) Exercise 12.6:

Implement deep copying for the **IntArray** class from Problem 2(a) and (b).

Requirments & Hints:

Please fill in the **TODO** part of copy constructor and assignment operator in *intarray.cpp*. You can define your own functions in the codes if necessary, but you need to follow the provided code framework.

Requirements for Assignment

I've provided a project named as *Assignment3.pro*. You should write **TODO** part in each cpp file according to the problem requirements. The test file is provided under *src* folder named *main.cpp*. Please pack your whole project files into a single .zip file, name it using your student ID (e.g. if your student ID is 123456, hereby the file should be named as 123456. zip), and then submit the .zip file via BB system.

Please note that, the teaching assistant may ask you to explain the meaning of your program, to ensure that the codes are indeed written by yourself. Please also note that we may check whether your program is too similar to your fellow students' code using BB.

Reminder: For windows users, please switch you input language to English before interacting in Stanford console. Or, you will get no response.