CS 150 Lecture 14 Exercises

Complete each of the exercises below and upload to Canvas before the deadline.

A. Exceptions

This exercise is designed to get you to use exceptions. There is an empty **try-catch** block provided which will do the catching but it's up to you to redesign the code to make use of it.

* There is an imaginary roster of students which we'll pretend is dynamic, but for the purposes of the exercise, is hard coded. You'll find the names in the source code. The roster is implemented as an **unordered\_map**, but you don't need to know how that works to solve this problem.
* When running the program, you are asked to enter a name. After doing so, it will print out the age of that student. Unfortunately, if the user enters a name that is not in the roster, **an exception is thrown** which we don't handle and the program just terminates.

Make two modifications to the program:

1. Catch the exception so that it can at least exit gracefully and write out a nice error telling the user that the name they entered doesn't exist. (Print the actual name they entered back at them) [Use this CodeCheck link](https://codecheck.io/files/2107192344dspijvd5ebkn9208vhlhpsqdg).
2. Modify it so that an invalid name results in the program going back around so that they can try entering a valid name. It should do this continuously until they actually enter a valid one. [Use this CodeCheck link](https://codecheck.io/files/21071923396hg2k0op24tgprihuuoe744v)

| *Screenshot of the results for modification 1* |
| --- |

| *Screenshot of the results for modification 2* |
| --- |

B. Assertions, Documentation & Exceptions

Function **preconditions** and **postconditions** state what is **assumed** to be true before a function is called and what is assumed to be true when the function ends. Complete the following for the **parseInt()** and **readInt()** functions declared in the **inthelper** library.

* Document the functions in the header, including **preconditions**, **postconditions** and **exceptions**.
* Throw an **invalid\_argument** with an error message when **parseInt()** fails.
* Use **parseInt()**, and endless loop and **try-catch** to implement **readInt()**;
* Use **assert()** to check that you never reach the end of **readInt()**.
* Break out of your loop to see that the assertion works.

| *Screenshot of inthelper.h after documenting the functions* |
| --- |

| *Screenshot of your source code for both functions.* |
| --- |

| *Screenshot of unit tests running.* |
| --- |

| *Use make run and show a parseInt test passing and failing.* |
| --- |

| *Use make run and show a readInt() with some examples* |
| --- |

C. Template Functions

Create the following function templates from Purdue University:

* **xchg(a, b)** has two parameters of the same type. A function which is **instantiated** from **xchg** will exchange or swap the values of these two parameters.
* **multiples(sum, x, n)**. The first two parameters will have the type represented by the function template type parameter **Kind**. **n** will always be **int**. The return type is **void**. All parameters are passed by value except for **sum** which is passed by reference. A function instantiated from **multiples** will compute:

**sum = 1 + x + 2x + 3x + ... + nx**

Use [this CodeCheck link](https://codecheck.it/files/2009102301c5aadpk8mgdh6alinvmc8xm6a) to solve this problem. Paste the requested screenshots below.

| *Screenshot of the source code for* |
| --- |

| *Screenshot of your source code for both functions.* |
| --- |