CS515 Assignment 5

The purpose of this assignment is to:

- Give you practice using Google Test
- Familiarize you with the behavior of the Map ADT
- Get you started with a set of tests that you can use for later assignments (7P, 9P, & 11P)

You will write a set of black-box tests for the Map ADT that can be expanded on in later assignments. Download all starter files from `~cs515/public/5P`

The file **`mapTest.cpp`** provided in the public directory contains a number of empty tests that you will have to fill. The name of each test describes what it should be testing, and you need to provide code that creates Map objects, performs operations on them, and tests that the operations are returning appropriate results. In order to test your tests, you are given a "golden" `map.o` file that should pass all tests when linked with `mapTest.cpp`. You are also given a number of "broken" object files each with problems that should cause certain sets of tests to fail. The provided makefile creates a test program for each of these broken object files so that you can check to see how your tests fare against each object file.

The keys and values in the Map for this assignment are both C++ strings. C++ strings can be compared with the normal `EXPECT_EQ` and `EXPECT_NE` assertions, but it is also possible to compare *C-style* strings in Google Test using the macros listed below. If you need to get a C-style string from a C++ string, use the `c_str()` method on the C++ string. (You may find that none of this is necessary.)

Fatal assertion	Nonfatal assertion	Verifies
ASSERT_STREQ(*str1*, *str2*);	EXPECT_STREQ(*str1*, str_2);	the two C strings have the same content
ASSERT_STRNE(*str1*, *str2*);	EXPECT_STRNE(*str1*, *str2*);	the two C strings have different content

(from https://github.com/google/googletest/blob/master/googletest/docs/Primer.md)

You should not modify the header file `map.h`. Your tests should be written in a single source file named **`mapTest.cpp`**. You can try your tests against various object (.o) files provided in the starter files.

When grading, we will look for the following:

1. Which tests pass and which tests fail for each object file (broken or not)
2. How a failing test fails--it should be for the same basic reason, even if not with the same exact names or values. It would help if the values could be the same, but we won't require it. Do try to have the failures in the same order if there are two different kinds of errors.
3. That no test...
    a) takes an extraordinarily long time
    b) crashes
    c) exhibits some other unexpected problem

Note: for `assignmentWithManyItemsHasCorrectValues` you should fill the set with hundreds or thousands of items, but just test 10 items deep in that list instead of all of them. Otherwise, your program will produce so much output that we will be unable to properly grade it.

Submission:

- Submit **`mapTest.cpp`**, **`map.h`**, **`makefile`**, and a **`README`** file.
  Make sure the provided header file `map.h` remains unchanged. You may submit the provided makefile, or use your own.
- You should also fill out the README file and submit it as well. To submit your files, use the following command on agate:

  `~cs515/sub515` **`5P mapTest.cpp map.h makefile README`**

- Do not turn in executables or object code.
- Make sure your submission compiles successfully on agate. Programs that produce compile time errors or warnings will receive a zero mark.
- Be sure to provide comments in your program. You must include the information as the section of comments below:

```
/**     CS515 Assignment X
File: XXX.cpp
Name: XXX
Section: X
Date: XXX
Collaboration Declaration: assistance received from TA, PAC etc.
*/
```

Some notes on grading:

- Programs are graded for correctness (output results and code details), following directions and using specified features, documentation and style.
- To successfully pass the provided sample tests is not an indication of a potential good grade; to fail one or more of these tests is an indication of a potential bad grade.
- You must test thoroughly your program with your own test data/cases to ensure all the requirements are fulfilled. We will use additional test data/cases other than the sample tests to grade your program.
- Here is a tentative grading scheme.

constructorCreatesZeroSizeMap	5
insertWhileEmptyReturnsTrueAndMapSizeIsOne	5
insertOfSameKeyReturnsFalseAndMapSizeRemainsSame	10
indexOperatorOfExistingKeyReturnsProperValueAndSizeIsSame	8
indexOperatorOnEmptyMapProperlySetsValueAndSizeIsOne	5
indexOperatorOnMissingKeyProperlySetsValueAndIncrementsSize	8
eraseOfExistingKeyReturnsTrueAndDecrementsSize	8
eraseOnEmptyMapReturnsFalseAndSizeRemainsZero	5
eraseOfMissingKeyReturnsFalseAndSizeRemainsSame	8
copyConstructorMakesCorrectSize	5
copyConstructorMakesSeparateCopy	8
assignmentOperatorMakesCorrectSize	5
assignmentMakesSeparateCopy	8
assignmentWithManyItemsHasCorrectValues	12