STRINGS (THE STRING TABLE)

1 String Table Tests

Answers:

1. Before running ch10_exer_010(), for each of the tests, please circle the answer you think the test will produce:
   - **Test 1** - YES
   - **Test 2** - NO
   - **Test 3** - YES
   - **Test 4** - NO
   - **Test 5** - NO
   - Test **6** - YES
   - **Test 7** - NO
   - **Test 8** - NO
   - **Test 9** - YES

2. What are the reasons for your answers?
   - **Test 1** - This entry was added in the code at the beginning of the function.
   - **Test 2** - This entry was NOT added in the code at the beginning of the function. Only the first 12 letters were added ("Hello World!"). Additionally, since this was the same value as for string1, nothing was added in that step (see below).
   - StringTable->insertn(string2, 12); // No add actually occurs
   - **Test 3** - This entry was added, but since we're doing a case-insensitive lookup, the string table will actually retrieve the entry that was added for "Hello World!"
   - **Test 4** - The string table creates its own storage for strings and doesn't reuse the addresses supplied by the compiler when it makes space for the const char * "Hello World!".
   - **Test 5** - This fails because "Hello World IGNORE THIS" is not in the string table. So, myString1 contains NULL which definitely doesn't match string1.

STRINGS (THE STRING TABLE)

• **Test 6 -** Because both lookups use a case-insensitive search, the string table will retrieve the first match it finds for string1 and string3, even though they each have different cases. It each instance the string table will provide the address that "Hello World!" is stored at, so both addresses (the values in myString1 and myString2) will match.

• **Test 7 -** This is almost the same as test 6, except when we stored string3, we used a case-sensitive insert. Now, by using a case-sensitive lookup for string3, we find the value that was stored earlier. This value will, of course, not match the one found for string1. i.e. The "Hello World!" storage location is not the same as the "HELLO WORLD!" storage location.

• **Test 8 -** Since "Hello World!" is not the same as "Hello World! IGNORE THIS", we don't get the same hash value.

• **Test 9 -** In this case, we are truncating the hash calculation for "Hello World! IGNORE THIS" to 12 characters. That is, the second hash calculation is done on the string "Hello World!", which is the same as the first hash.