```cpp
/*
 * Programming Challenge 1
 */
#include <cassert>
#include <iostream>
using namespace std;

string greet (string name);
string checkExperience (char hasExperience);

/* for unit testing -- do not alter */
template <typename X, typename A>
void btassert(A assertion);
void unittest (string s, char c);

int main (int argc, char* argv[])
{
    // CODE HERE
    //
    // 1 declare a string variable named userName
    // 2 declare a char variable named programmedBefore
    //   and initialize programmedBefore to the value 'z'
    // 3 display a welcome message to standard output
    // 4 prompt the user for a name
    // 5 read in the name from standard input and store
    //   in the variable userName
    // 6 ask the user the yes/no question "Have you programmed
    //   in C++ before?"
    // 7 read in the answer from standard input and store
    //   in the variable programmedBefore

    unittest(userName, programmedBefore);

    return 0;
}

/*
 * Create a string greeting that is the concatenation of a message
 * and a name.
 * @param name a string containing a user name
 * @return the string "Nice to meet you, NAME" where NAME contains
 *         the paremeter value
 */
string greet (string name)
{
    // CODE HERE
}

/*
 * Create a string message based upon whether or not a user has C++
```

```
 * programming experience.
 * @param hasExperience a char ('Y'es or 'N'o) representing whether or
 *        not a user has C++ programming experience
 * @return "Welcome back" when hasExperience is 'Y', else 'Get ready to
 *         have some fun"
 */
string checkExperience (char hasExperience)
{
    // CODE HERE
}

/*
 * Unit test. Do not alter this function.
 */
void unittest (string s, char c)
{
    if (s == "teacher")
    {
        cout << "\nSTARTING UNIT TEST\n\n";

        try {
            btassert<bool>(greet(s) == "Nice to meet you, teacher");
            cout << "Passed TEST 1: greet function\n";
        } catch (bool b) {
            cout << "# FAILED TEST 1 #\n";
        }

        if (toupper(c) == 'Y')
        {
            try {
                btassert<bool>(checkExperience(c) == "Welcome back");
                cout << "Passed TEST 2: checkExperience function\n";
            } catch (bool b) {
                cout << "# FAILED TEST 2 #\n";
            }
        }
        else if (toupper(c) == 'N')
        {
            try {
                btassert<bool>(checkExperience(c) == "Get ready to have some fun");
                cout << "Passed TEST 2: checkExperience function\n";
            } catch (bool b) {
                cout << "# FAILED TEST 2 #\n";
            }
        }

        cout << "\nUNIT TEST COMPLETE\n\n";
    }
    else
    {
```

```
        cout << greet(s) << endl;
        cout << checkExperience(c) << endl;
        cout << "\nRun program with username \"teacher\" to see unit test output.\n" ;
    }
}

template <typename X, typename A>
void btassert (A assertion)
{
    if (!assertion)
        throw X();
}
```