Workshop #9

Please use the two template source files to complete the two exercises.

1. **Program: wk9_1.cpp**

   Description: `std::string` and `std::vector`

   This program helps you get familiar with the `std::string` class and `std::vector` class in C++. The program asks the user to enter a list of first names and last names. Then you need to combine the first name and last name into a new string, save the names to a string vector, and then print out all the names at the end. The program asks the user if they want to continue to enter names. The user can enter `Y` or `N` only. You have to do input validation for this part. Your program should be able to output as follows:

   Sample run:

```
Enter first name: Hello
Enter last name: Kitty
Continue? Y
Enter first name: Mickey
Enter last name: Mouse
Continue? Yes
Wrong input! Enter again: No
Wrong input! Enter again: Y
Enter first name: Snow
Enter last name: White
Continue? N

Names:
1.  Hello Kitty
2.  Mickey Mouse
3.  Snow White
```

   Hints:

   1. You need to declare an empty vector and use its member function `std::vector.push_back()` to add name by name.
   2. Use the `+` overloaded operator to attach a string at the end of another string.
   3. Use the `std::getline` function to save an entire line of string from the keyboard to a `std::string` object.
   4. You can use a comparison operator (>, >=, <=, ==, !=) to compare a `std::string` object with another `std::string` object.

(*Continued on next page*)

2. **Program: wk9_2.cpp**
Description: Overloaded functions and default arguments
Complete the following program with appropriate overloaded functions.

```
#include <iostream>
#include <cstring>
using namespace std;

// Function prototypes here

int main() {
    cout << "Testing function overloading with add()!" << endl;
    cout << "add(1, 1) is: " << add(1, 1) << endl;
    cout << "add(1.1, 1.1) is: " << add(1.1, 1.1) << endl;
    cout << "add(\"Super\", \"Mario\") is: "
         << add("Super", "Mario") << endl;
    cout << "Now testing default parameters with add()!" << endl;
    cout << "add(1) is: " << add(1) << endl;
    cout << "add() is: " << add() << endl;

    return 0;
}
// Function definitions here
```

Sample run:

```
Testing function overloading with add()!
add(1, 1) is: 2
add(1.1, 1.1) is: 2.2
add("Super", "Mario") is: Super Mario
Now testing default parameters with add()!
add(1) is: 1
add() is: 0
```