Seth Denney
COMP 2710 - Software Construction: Homework 1

1.)
```
#include <iostream>
#include <string>
using namespace std;

int main () {

	string firstname = "John", lastname = "Smith";

	cout << firstname + " " + lastname;

	return 0;
}
```

2.)
```
#include <iostream>
#include <string>
using namespace std;

int main () {

     const string PRE = "<", POST = ">";
     string username = "John", message = "Welcome to the group!",
     messagebuffer;

     messagebuffer = PRE + username + POST + message;
     cout << messagebuffer;

     return 0;
}
```

3.)
```
#include <iostream>
#include <string>
using namespace std;

int main () {

     const string PRE = "<", POST = ">";
     string username = "John", message = "Welcome to the group!",
     messagebuffer;

     messagebuffer = PRE + username + POST + message;
     cout << messagebuffer << endl;

     string username2 = "Jane", message2 = "Glad I'm in the group!";

     messagebuffer += PRE + username2 + POST + message2;
     cout << messagebuffer;

     return 0;
}
```

4.)

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

int main () {

     const string PRE = "<", POST = ">";
     string messagebuffer;

     string message3 = "Let's go Auburn!", message4 = "Let's go!";

     messagebuffer = message3 + "\n" + message4;
     cout << messagebuffer;
     //When printed, the console will display each message on a separate line.

    return 0;
}
```

5.) The function **string.size()** returns the number of characters present in the string object (ie: the length of the array used to store the contents of the string). In question 2, **message.size()** would return **21.**

6.) Executing **message.clear()** in question 2 would remove the contents of **message** and leave its source array empty, but not **null.** Subsequently, **message.size()** would then return **0.**

7.)
```
int index = message_buffer.find("<");
```

8.)
```
int index = message_buffer.find("<", message_buffer.find("<") + 1);
```

9.)
```
string temp = message_buffer.substr(message_buffer.find("<") + 1,
string::npos);
temp = temp.substr(0, temp.find(">"));
cout << "First User Name: " << temp;
```

10.)
```
string temp = message_buffer.substr(message_buffer.find("<",
message_buffer.find("<") + 1) + 1, string::npos);
     temp = temp.substr(0, temp.find(">"));
     cout << "Second User Name: " << temp;
```

11.)
```
string temp = message_buffer.substr(message_buffer.find(">",
message_buffer.find(">") + 1) + 1, string::npos);
     temp = temp.substr(0, temp.find("<"));
     cout << "Second Message: " << temp;
```

12.) For a given **string messagebuffer**, different input methods will result in different values being stored.

Given that the user enters “**Welcome to the Message System**”:

a) **cin >> messagebuffer** will only store “**Welcome**”, because **cin** reads only the first set of characters until a “ “ character is found.

b) **getline(cin, messagebuffer)** will read and store the entire line of characters, and advance the read pointer to the next line, denoted by a “**\n**” character.

Source Code in HW1.cpp.

Compile as follows:
>>g++ HW1.cpp
>>./a.out

Sample Script from Console:
```
============================================================
               Welcome to the Message System!
============================================================

Enter user name: Bob
Enter the message: Hey! My name is Bob!
Who else is here?
$

Any more users? yes

Enter user name: Sally
Enter the message: Yes, this is Sally!
How are you Bob?
$

Any more users? yes

Enter user name: Bob
Enter the message: Great!
$

Any more users? no

The current messages are:

Bob ~ Hey! My name is Bob!
Who else is here?

Sally ~ Yes, this is Sally!
How are you Bob?

Bob ~ Great!
```