Chapter 9: Strings

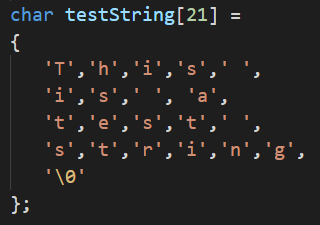
Last week, while creating your inventory array you were given code containing the key word “string”. This week we will be going into more depth about what a string is and how to use them, along with the use of char arrays. Before string was introduced as part of standard C++, the C-style char arrays were the common method used to create sentences.

If you have not yet completed your array work from last week, I suggest you do so before continuing, so you can get to grips with what we are making a little easier.

**Char Array**

The char array approach to strings comes from the C programming language. The way it works is that we create an array of char data type, which is terminated with a null character ‘\0’

To declare a string using this approach we have two options. The first is to categorically state what each character in the array is and add the terminating null character ourselves:



Note: The size of the array is 1 bigger than the size of the string. This is to cater for the null character.

You can of course miss out the size as in array declarations, but can then use the double quotes as shown here:



**Example Char Array**

If you missed the lecture this week, go ahead and replicate the code below to familiarise yourself with how to create char arrays.



Within the string header (also known as cstring header), there are a variety of functions, which can be used on null terminated strings. We will cover a few as detailed in Table 9.1: String Functions and in the examples below, but please look at: [http://www.cplusplus.com/reference/cstring/](http://www.cplusplus.com/reference/cstring/%20) for further details.

|  |  |
| --- | --- |
| **Function** | **Description** |
| Strcpy\_s(s1, s2) | Copies string s2 into string s1. |
| Strcat\_s(s1, s2) | Concatenates s2 on to the end of s1. |
| strlen(s1) | Returns the length of s1. |
| strcmp(s1, s2) | Returns 0 is s1 and s2 match.  Returns less than 0 if s1 < s2.  Returns greater than 0 if s1 > s2. |

Table 9.1: String Functions

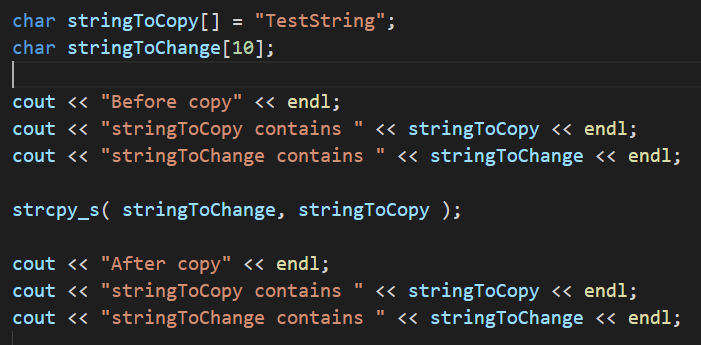
Note: In all the below examples it is assumed that the program has included the <string> header like so:



Note: As Visual Studio is updated often by Microsoft, functions can become depreciated. If you get a warning of a depreciated function, you will need to investigate how to use the replacement.

**Example of Copying Character Strings**

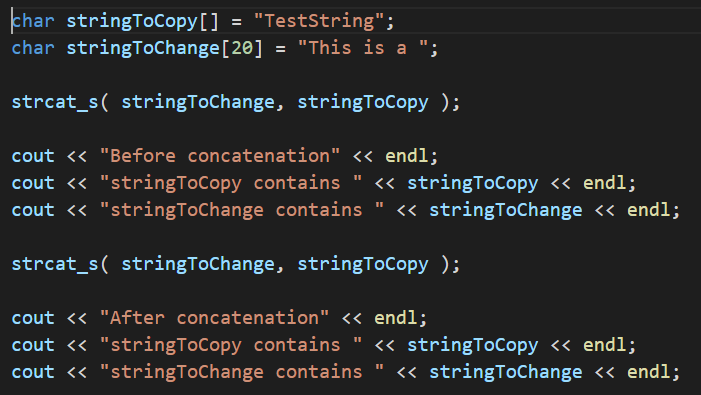
To copy the contents from one char array to another you can use the strcpy() function. The first char array parameter passed in will be the one copied to; the second char array passed in is the one to be duplicated.



In this example we copied the entire string. It is possible to copy a portion of the string using the strncpy\_s() function.

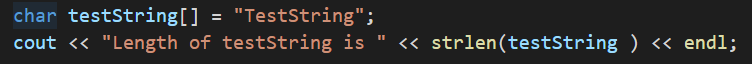
**Concatenating Character Strings**

To copy the contents from one char array and attach it to the end of another you can use the strcat() function. The first char array parameter passed in will be the one copied to, the second char array passed in is the one to be duplicated.



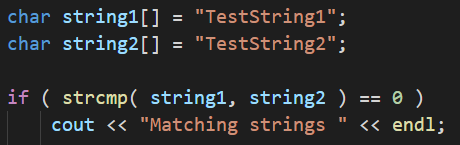
**Character String Length**

To determine the length of a string there is a function for that too. Use the strlen() function as shown below:



**Comparing Strings**

To determine whether two char arrays match simply pass the two strings through as parameters to the strcmp() function. This will only return a result of 0 if they are the same. Any other result means they do not match.



**Program 25: Initials**

Write a program to prompt the user to enter in a single line their first name initial, followed by a space, their middle name initial followed by a space and the entirety of their surname. Store this in a char array.

The program should then output the first initial on one line, then the middle initial on a separate line and the surname on a line of its own.

Note: This program is to demonstrate the use of char arrays. Ensure that you complete this program using a char array and that the data input by the user is stored in a single char array. You can separate out the various parts of the name into separate arrays afterwards, but the initial read from the console should put the entirety of the input into a single char array.

**Tips**

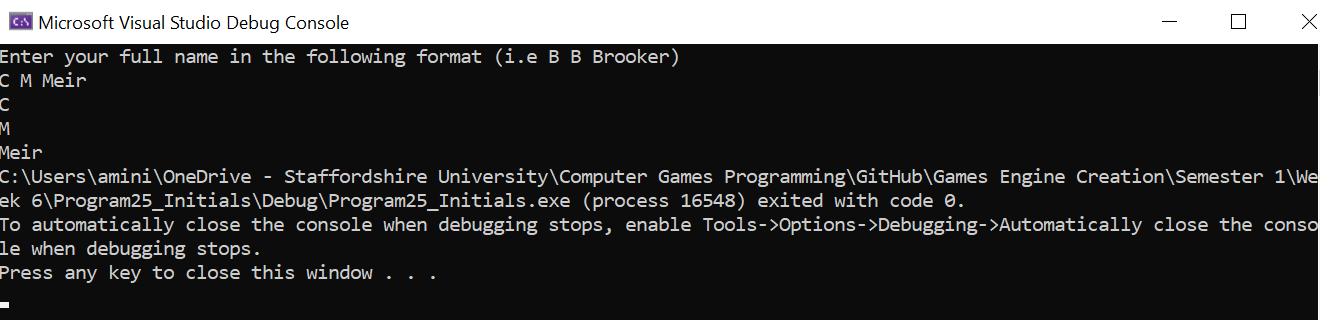
* You will need to loop through your characters to find the first occurrence of a space. This should give you the information you require to be able to access the middle name initial.
* You will need to use **cin.getline** instead of **cin**, if cin is used then only characters leading up to the first space will be stored. To use it with a char array you call the function, and as parameters pass the array name, then size. Like so**: get.line(name, 50);** You can read up on this function here:

<http://cplusplus.com/reference/istream/istream/getline/>

**Program 25 Source Code:**

****

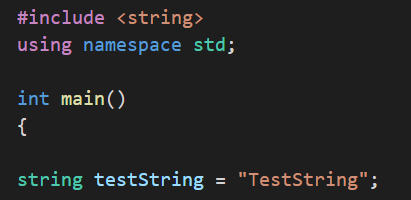
**Program 25 Screenshot:**

****

**The String Approach**

The string approach is an alternative way of dealing with a string of characters. Using string you are in fact using an underlying class, which is found in the string header. You must include this header at the top of any source file that uses string. String introduces a host of new functions, some of which are described below, others that you will need to research on your own.

So, to create a variable of string type it is as simple as:

****

Note: You MUST now use double quotes. Single quotes will create a syntax error.

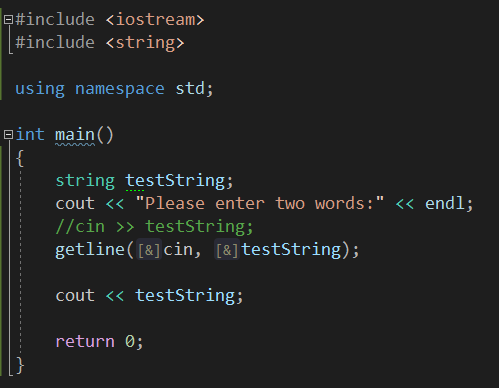
When using a string type with the console, unlike the char array where you pass the array name and size to the getline function, you instead pass (cin, stringName) like so:



As before by just using cin alone, only characters that come before the first whitespace will be stored.

**Example of Input/Output with Strings**

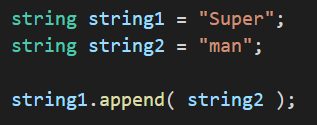
Create a program and add the following code, run it twice to see what happens with cin and getline.

****

Note: If you were to create a program where you use cin to take in one word followed by getline to take in numerous words; you would be faced with getline skipping input. This is due to the new line or **endl** not being flushed and counting as an input. There is a number of ways to fix this, with one being to use cin.ignore before getline. You can express what it is to be ignored by the console as a function parameter. The most common being: 

**Example of String Concatenation**

There are a couple of approaches to combining string objects. The first to be shown is the append() function, the second is using operators that have been overloaded such as + and +=.

****

The above example will result in string1 containing the text “Superman”. The use of operators is shown next, which has the same result.



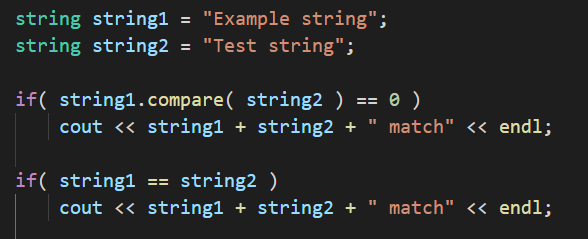
Alternatively, you could create a new string to hold the result:



**Note:** Overloading Operators refers to changing the functionality of an operator depending on the type being used. For example, int += int results in an int, string += string results in a string.

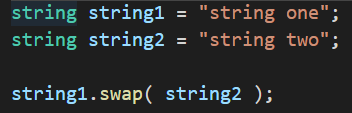
**Example of String Comparison**

There are also a couple of possible approaches to determining if two string types match. The first is the compare() method, the second is an overloaded == operator. Both examples below will give the same result.



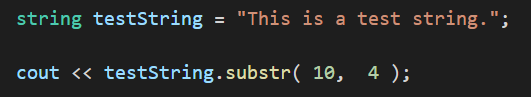
**Example of Swapping Strings**

**<string>** has a function called **swap()** and this allows you to swap the contents of one string for the contents of another.



**Example of Substrings**

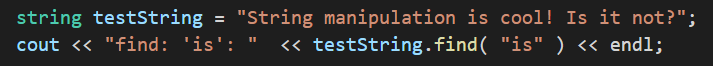
Another useful function is the substr(), which allows you to retrieve a particular portion from a string. The first integer parameter is the index in the string to start at, and the second integer parameter is how many characters from the first index.



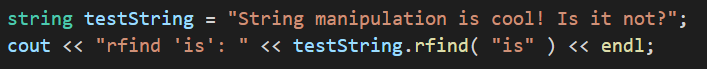
**Example of Substring – Finding**

At times you will need to locate a particular phrase or character. **<string>** provides functionality for this. There are a few variants of the **find()** function and an example of each is shown below.

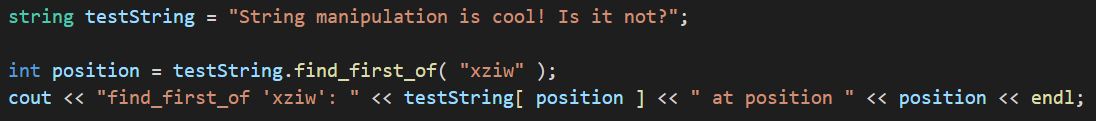
To find the location of the first occurrence of a phrase:



To find the location of the first occurrence of a phrase starting at the end of the string, use the rfind() method.



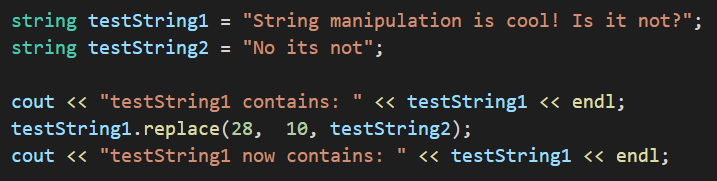
To locate the first / last occurrence of a character from a string you supply use the following examples.



The above example will find a ‘i’ at position 3.

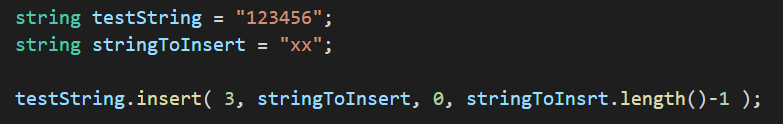
**Example of Substring – Replacing**

If you want to replace a character in a string with a different character then it’s as simple as using the replace() function. You will no doubt need to use the find() function first to locate the position of the character you want to replace. The first integer parameter is the position to start overwriting, the second parameter is how many characters to replace and the third is the string to replace with.



**Example of Substring – Inserting**

Replacing characters in a string is all well and good but what if you need to add some characters? That’s where the insert() function comes in useful.



The above example will result in testString containing “123xx456”.

Note: Before we move onto the string programs for you to implement it is important to note that there are many overloaded versions of the functions covered in this chapter, and a lot of others that have not been covered. It is up to you to do your own research to discover what they are and how they work.

**Program 26: Replacing Characters**

Write a program to prompt the user to enter in a sentence, which is then stored in a variable of type string. Output the string they have entered to the console screen, then replace every occurrence of ‘e’ with an ‘X’.

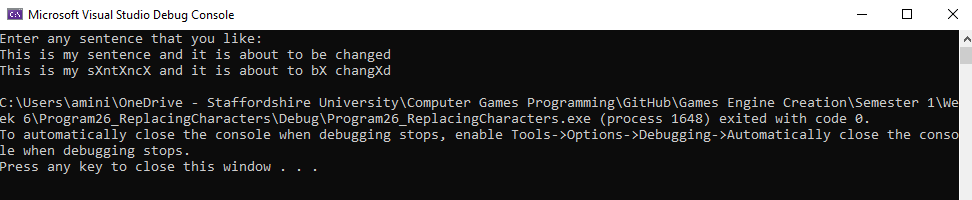
The program should then output the altered sentence.

**Tips**

* You will need to keep an index of position
* You will need to use a loop to access the characters in the string using the find and replace
* Alternatively, do research on the functions available in the <algorithm> file.

**Program 26 Source Code:** ****

**Program 26 Screenshot:**

****

**Program 27: String Manipulation**

Write a program to prompt the user to enter their name, which is then stored in a variable of string data type. Insert their name in between the ‘XX’ in the following string:



Output this string to the console. The program should then remove the two X’s and then output the string to the screen again. (You should have two outputs at this point:

“Do you know who loves C++ XnameX does!” and “Do you know who loves C++ name does!”)

The program should then ask the user to input another name (this one **MUST** longer than the first). The program should then locate the first name in the test string and replace it with the new name.

Output the altered string with the second name in place of the first to the console and screenshot the entire process.

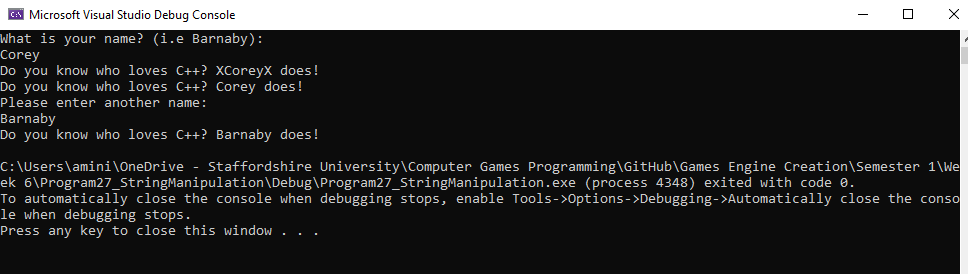
**Tips**

* Store the first and second name in a strings of their own.
* Don’t forget to test for a longer name on the second entry

**Program 27 Source Code:**

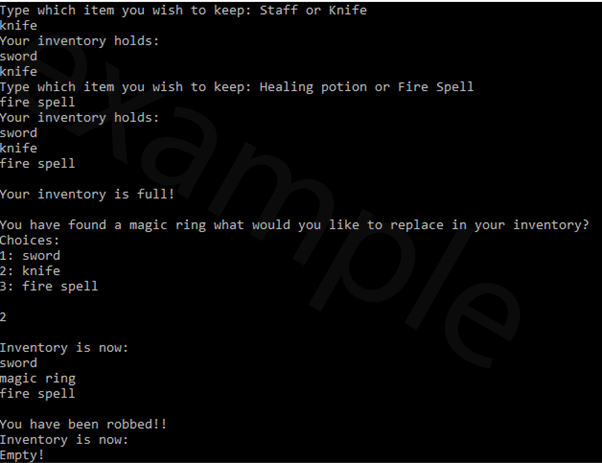
****

**Program 27 Screenshot:**



**Program 28: Strings and Vectors**

Below is the output of a program you have to try and replicate.



The program must:

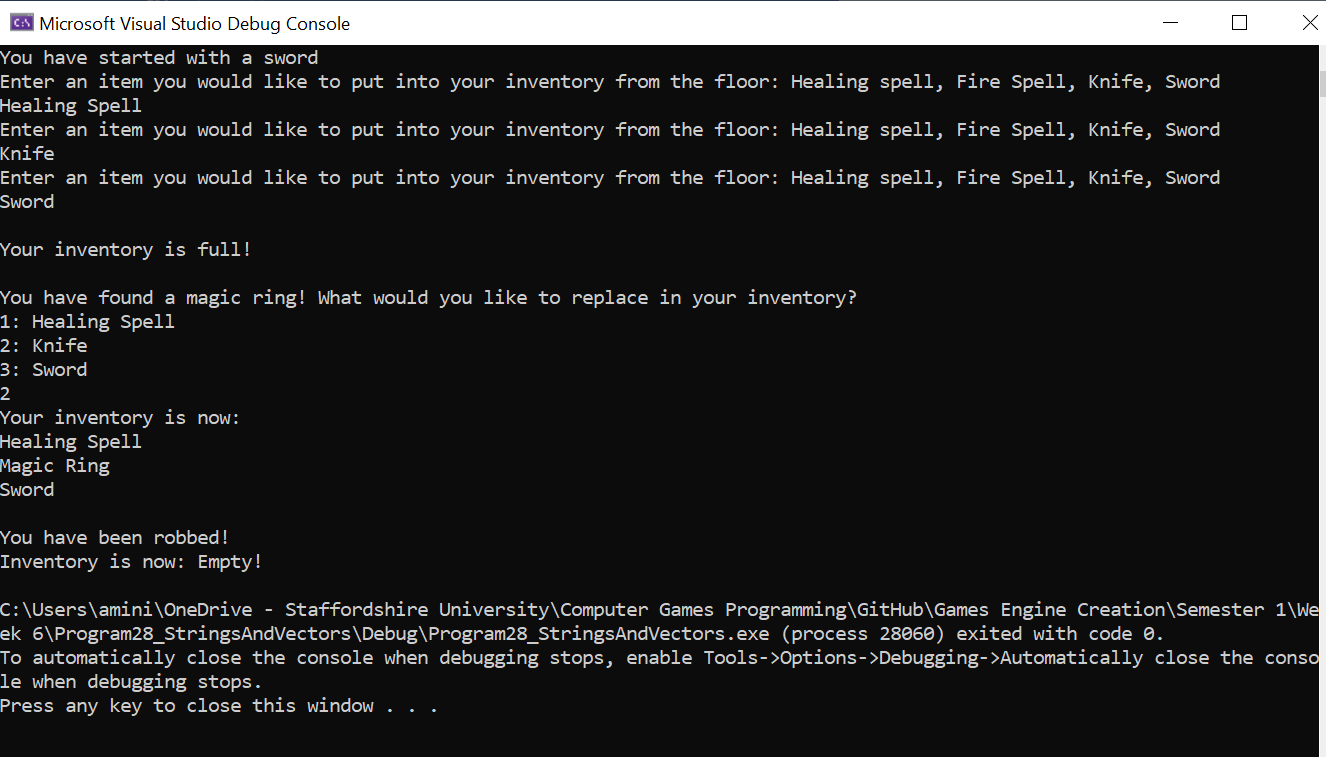
* Start with an empty vector of string
* Store the user’s choice in the aforementioned vector
* **Make use of cin and getline**
* Use a loop to ask for items until inventory is full. Set inventory max to 3.
* Give the user the contents of their inventory and replace their choice with the magic ring
* After being robbed you must delete the contents of the inventory and output the contents of the inventory one last time.
* If you are up to the challenge, proof check every entry to ensure no wrong entries. This is not a necessity.

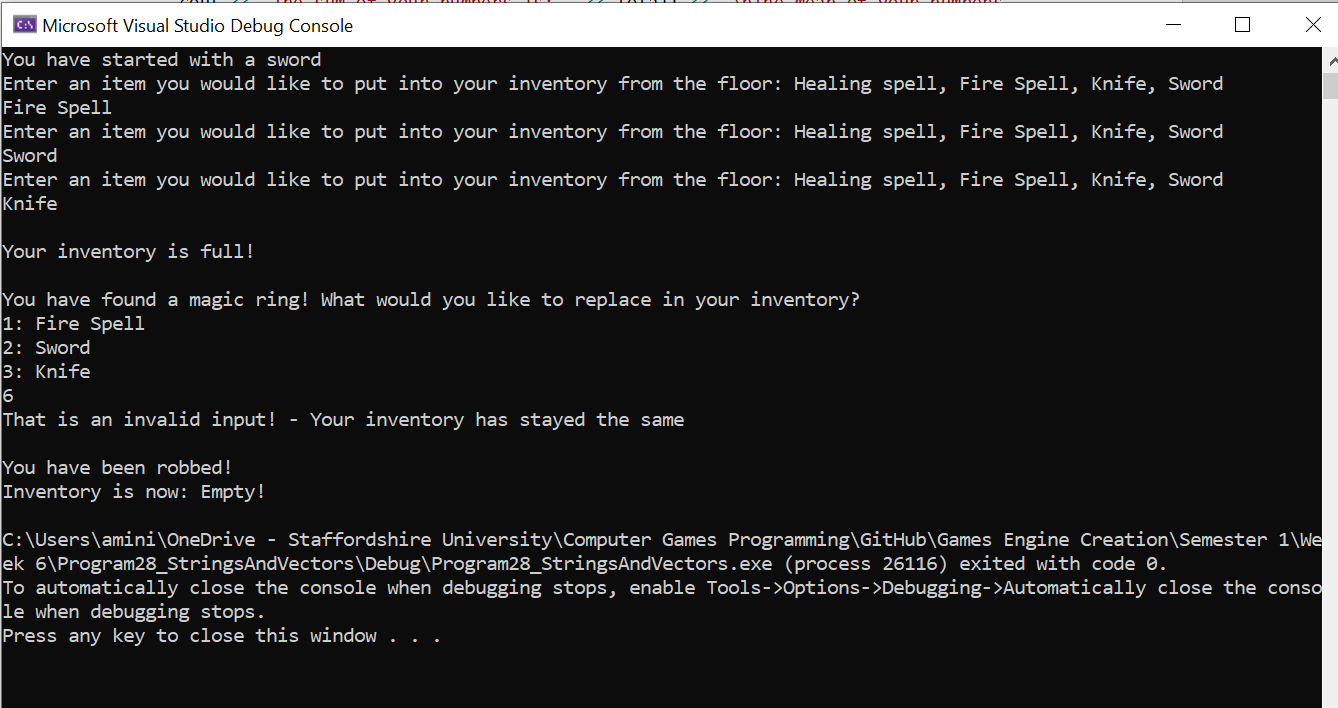
**Program 28 Source Code:**

****

****

**Program 28 Screenshot:**

****

****

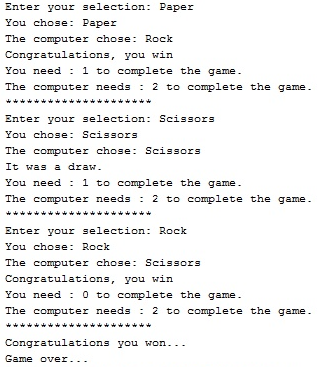
**Program 29: Rock, Paper, Scissors**

Write a game of Rock, Paper, Scissors; where the user is prompted to enter their choice as a string (e.g. Paper or paper) and the computer's choice is generated randomly. The first to score 3 wins, is the overall winner.

At the conclusion of the game your program should output the result (who won) and the scores of both players.

Those not familiar with the game can learn a little about it here (<https://en.wikipedia.org/wiki/Rock-paper-scissors)>

When you have completed your program, take a screen shot of the last three plays and the concluding output (see below for an example).



**Tip**

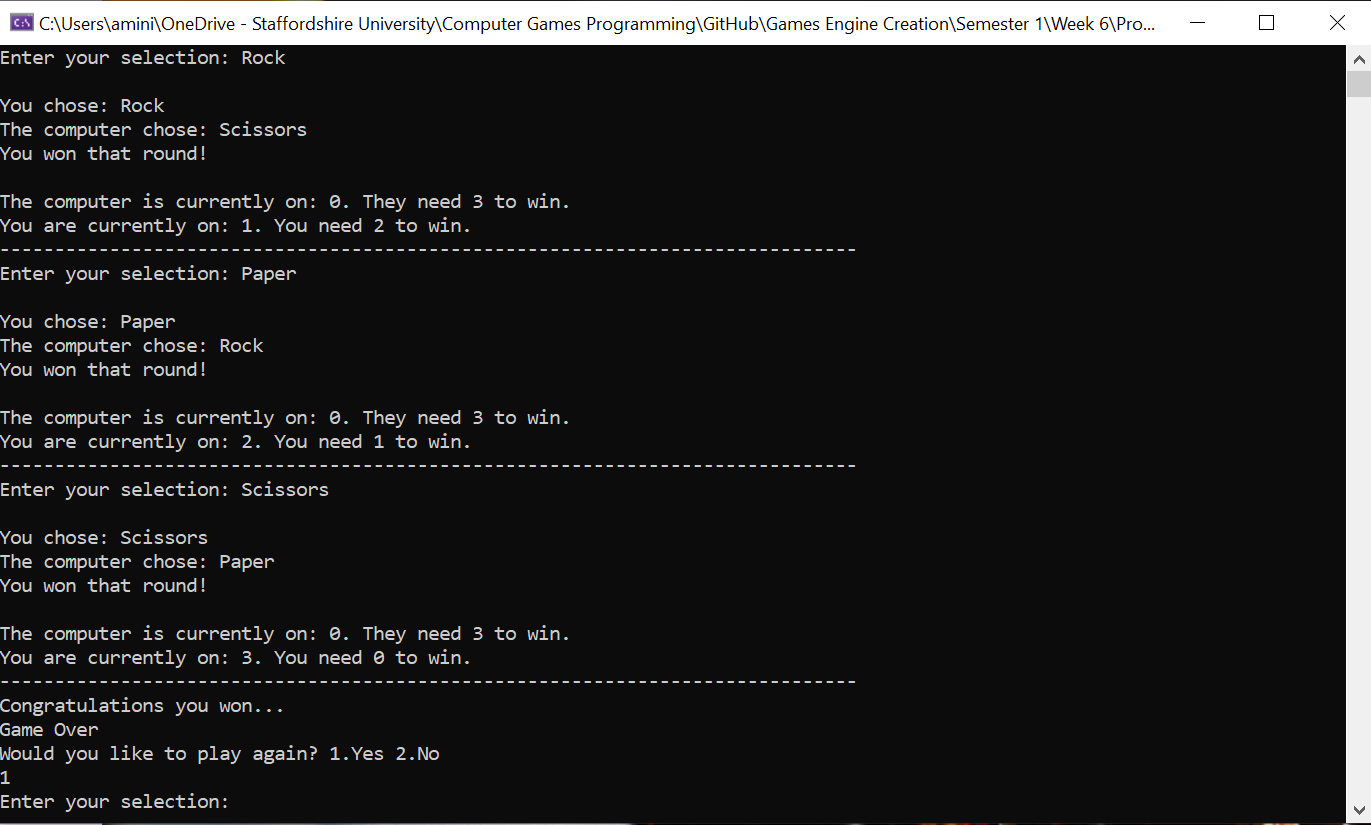
* Research the use of Rand or Random Device within the <random> file

**Program 29 Source Code:**

****

****

****

**Program 29 Screenshot:** ****

