Introduction to C Strings

Supplemental Document for Assignment #1

Account Ticketing System

Introduction

The section of the notes regarding C strings is not completely covered until week #8, however, you will need to know some very simple facts about string handling in C for this assignment. If you wish, you can read more about string handling here: https://ict.senecacollege.ca/~btp100/pages/content/strin.html

A C string is an array of type char with a special terminator character called the null byte (represented by the following symbol '**\0**'). When declaring a C string array, it is necessary to always make the array one character larger than the maximum number of characters it needs to be able to store.

Declaring a C string array to store up to 30 characters

```
char collegeName[31];  // 30 meaningful characters + 1 for the null byte
```

To read a C string from standard input (keyboard)

```
scanf("%30s", collegeName);
```

Note

- There is no ampersand (& - address of) before collegeName, the name of an array is its address.
- The **30** specifies the maximum number of characters to be read which protects the array from an attempt at storing more characters than it is sized for.
- The **s** specifier represents a C string.
- Caution: Whitespace characters (space, tab, etc.) act as a delimiter and scanf will only assign the characters up to the first whitespace character. Entering: "Seneca College" will assign "Seneca" to the collegeName array variable leaving " College" in the standard input buffer

Read a C String including whitespace characters

```
scanf("%30[^\n]", collegeName);
```

Note

- The **[^\n]** interprets to: "Read everything up to but not including the newline character".
- This will permit whitespace characters to be assigned to the collegeName array variable.

Displaying a C string

```
printf("%s\n", collegeName);
```

Just as in the ***scanf*** function where the **%s** specifier represents a C string, the same applies to the ***printf*** function.

Initializing a C string

```
char collegeName[31] = "Seneca";
```

The ***collegeName*** variable is sized to accommodate up to 30 printable characters plus a null byte. The above example initializes a C string simply by using **double quotes** which will automatically add a null byte character at the end.

The result of this initialization will set the first 6 characters (indexes 0-5) with the letters “Seneca”, and append the null byte character to index 6:

Index:	**[0]**	**[1]**	**[2]**	**[3]**	**[4]**	**[5]**	**[6]**
Character:	**S**	**e**	**n**	**e**	**c**	**a**	**\0**

Hint: You can determine the end of a C string (or its length) by iterating the character array until you find the null byte character ‘**\0**’. In the case above, the null byte character is at index 6 which so happens to be the length of the word “Seneca”.

Warning

You will not be able to assign to a C string character array a value using the double quotes AFTER the array has been declared – double quote assignment will only work during initialization of the character array.

For example, the below is not possible and will cause a compile error:

```
char collegeName[31];
collegeName = "Seneca"; // <- - ERROR! Can’t do this!
```