CS 010 - Introduction to Computer Science I

Lab 4 - String Member Function, Characters and Branches

**Suggested Pre-Lab Work (you should have completed at least some of these items)**

Zyante Chapter 3 and corresponding Codelab exercises

Video tutorials: [Module 4 playlist](http://www.youtube.com/playlist?list=PLTTJbxrH72A32ELhC-htWiKrHTFjDu6tw)

**Collaboration policy**

Collaboration on Piazza, Cloud9 or in person on these lab exercises is strongly ENCOURAGED. They are intended for practice, not assessment -- feel free to ask for help from, and provide help to, others. **You may not, of course, blindly copy solutions from one another (or from anywhere else) or simply write code for someone else,** but you can certainly help each other debug, give plenty of suggestions and hints, *explain* why things work or don't work, etc.

Read the full policy at: [Full Collaboration Policy](https://docs.google.com/document/d/1WyzL3qvKLrC1UCRf178b_wYWQmEZlhDObFNFb79U63I/edit?usp=sharing)

**Lab Objectives**

To gain experience with:

|  |  |  |  |
| --- | --- | --- | --- |
| * string at() | * characters | * branches | * compound logic |

**Exercise 0: Setup & Lab Submission Framework**

You will continue to **submit a single file to R'Sub**, but you will follow the framework below to set up your program.

Your source code file will have multiple blocks, one for each exercise. The framework allows the tester (human or R'Sub) to specify the exercise to test during program execution. **All the code for an exercise - everything you would previously have put inside main by itself - will now go in the corresponding code block (see comments below).**

|  |  |
| --- | --- |
| **Framework** | **Setup** |
| #include <iostream>  using namespace std;  int main()  {  int ex;  cout << "Which exercise? ";  cin >> ex;  cout << endl;  if ( ex == 1 )  {  // All Exercise 1 code  }  else if ( ex == 2 )  {  // All Exercise 2 code  }  return 0;  } | 1. Create a file named lab4.cpp within the proper directory. 2. Copy and paste the denoted framework into your file. 3. Copy-paste the proper assessment header, then fill in. |

**Exercise 1: Character arithmetic**

Although it might seem odd, the char data type (which we use for storing character data - i.e. the symbols on a keyboard) is actually a variant of the integer data type: it is just a number representing the ASCII code for characters. Since it only occupies 1 byte (8-bits) of memory, it can only store very small numbers, from -128 to +127 *(the positive numbers represent the standard keyboard symbols: letters, numbers, punctuation, etc)*.

Since it is just a number, we can do some interesting character manipulations with arithmetic - which will come in handy later on when we try our hand at cryptography - but for now, we will just do some really simple operations.

The exercise: Ask the user to input a **string**, then extract the first character of the provided string using the string member function at (at(0)). After extracting the character at position 0 of the string, use character math to calculate the *position* of the letter within the alphabet and output that location. Letter 'a' will be considered position 1 of the alphabet.

For this exercise, consider only strings containing all lower case letters.

***Note****: you do* ***NOT*** *need to know the ASCII values for the letters to do this exercise!!  
You only need to know that their values are all in sequence, as you would expect. In fact it is frowned upon if you use the ASCII values within your source code.*

**Example Run** (inputs are **bold & underlined** to emphasize the difference between output and typed input)

|  |  |
| --- | --- |
| **Example 1**  Which exercise? **1**  Enter a string: **ball**  b is letter 2 of the alphabet | **Example 2**  Which exercise? **1**  Enter a string: **zoo**  z is letter 26 of the alphabet |

**Exercise 2: A test of character**

Your goal is to get a single keystroke from the user and report whether the keystroke is:

* a lowercase alphabetic character ('a' through 'z')
* an uppercase alphabetic character ('A' through 'Z')
* a numeric digit ('0' through '9')
* some other character

Write a program that prompts the user to enter a single character: you will capture that input in a variable of type char.Then, using a sequence of if - else if - else statements, report back to the user what type of character they entered, as above.

**How will you know?**

Recall that the char data type is really just a "small number" type, a char can store numbers from -128 to +127 (or from 0 to 256, depending on how you use it). Each number from 0 to 127 is mapped to a character on your keyboard, grouped more or less as you would expect:

* Characters from '0' to '9' in sequence come first (i.e. lower number mappings).
* Characters from 'A' to 'Z' in sequence are in the middle.
* Characters from 'a' to 'z' in sequence are the higher number mappings.
* The punctuation marks and other symbols are all over the place.

So each if statement just has to test whether the value input to your char variable lies in one of those ranges (or not). You compare chars just as you compare numbers, so the expression:

('A' > 'a') -> returns false

and the expression

('9' < 'A') -> returns true

**Output Strings**

* You entered a lowercase letter.
* You entered an uppercase letter.
* You entered a numeric digit.
* You entered an unknown character type.

**Example Run** (inputs are **bold & underlined** to emphasize the difference between output and typed input)

|  |  |
| --- | --- |
| **Example 1**  Which exercise? **2**  Enter a character: **b**  You entered a lowercase letter. | **Example 2**  Which exercise? **2**  Enter a character: **@**  You entered an unknown character type. |