CS 010 - Introduction to Computer Science I

Lab 5 - Introduction to for Loops

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

Zyante Chapter 4.1-4.5 and corresponding Codelab exercises

Video tutorials: [Module 5 playlist](https://www.youtube.com/playlist?list=PLTTJbxrH72A1KtVyIg7uQ8IrK1Y-zjDq5)

**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:

|  |  |  |  |  |
| --- | --- | --- | --- | --- |
| * at() | * find() | * for loops | * bool | * relational expressions |

**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. **All the code for an exercise should go in the corresponding branch.**

|  |  |
| --- | --- |
| **Framework** | **Setup** |
| #include <iostream>  using namespace std;  int main()  {  int ex;  cout << "Which exercise? ";  cin >> ex;  cout << endl;  if (1 == ex)  {  // All Exercise 1 code  }  else if (2 == ex)  {  // All Exercise 2 code  }  ...etc...  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 - Characters of a string!**

Write a program that takes a single word string from the user. After acquiring this input from the user, your program will **use a loop** to output whether the string contains *either* an 'x' *or* an 'e' (or *both*). You will need to utilize ***only*** the size() and at() member functions of the string class as you iterate through the entire string and acquire each character individually to check its value.   
*Since we know the beginning index of strings and its size, what type of loop should we utilize?*

**Example Run** (inputs are **bold & underlined** to emphasize the difference between output and typed input)  
 *(as always, you must output a blank line following every input statement to permit testing)*

**Example 1**

Which exercise? 1

Enter a word: **heel**

Your word, heel, contains the character ‘e’

**Example 2**

Which exercise? 1

Enter a word: **extra**

Your word, extra, contains the character ‘x’

Your word, extra, contains the character ‘e’

Note: Even though the word "heel" has two e's, there is only **one** output statement.

How did we do that? Well, we obviously did not output the information each time we detected it in the loop; in fact, the *only* way to do it is to keep track (inside the loop) of whether an 'e' is found, and then *after* the loop has finished, report it (or not).

***We expect you to use one or multiple boolean variables for this exercise.***

### 

### 

### Exercise 2 - modifying strings

In the previous exercise. you used the at() function to *test* the value of individual characters in a string. This time, you will use the same function to *modify* characters.

Follow exactly the same process as in Exercise 1, but now instead of reporting whether the characters are present, *change* all occurrences of ‘e’ to ‘3’, of ‘i’ to ‘1’, and of ‘x’ to ‘\*’ *(this is an unfortunately common tactic for creating “strong” passwords)*.

As you can see, the at() function is versatile - it works for both reading ("looking at"), and writing ("changing") individual characters inside a string.

**Example Run** (inputs are **bold & underlined** to emphasize the difference between output and typed input)  
 *(as always, you must output a blank line following every input statement to permit testing)*

Which exercise? **2**

Enter a word: **heel**

Your word transformed is h33l

**Exercise 3 - looking for stuff**

Frequently, we will need to know if a certain character, or word, or phrase, is present in a string; and if it is present, we need to know where. We could use the looping technique we practiced above - but this is such a common task that the string class provides us with a function for it (find).

**Prompt the user for a string, read in a single word.**

Now, using the find function, report the index of the first occurrence of:

1) the character '.' and

2) the location of the first occurrence of the string "stop".

If what you are searching for is NOT present in the string, report that fact.

Remember, the find function can take an argument of type char, or of type string, and if the requested value is not present, the function returns the special value string::npos

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

|  |  |
| --- | --- |
| **Example 1**  Which exercise? **3**  Enter a word: **home??**  The entry does not contain the character '.'  The entry does not contain the word "stop" | **Example 2**  Which exercise? **3**  Enter a word: **pit-stop.**  The character '.' is located at index 8  The word "stop" starts at index 4 |