Lab 1

Counting Uppercase and Lowercase Characters

You are going to write a program that takes a string and prints out two messages. One message tells you how many uppercase characters are in the string. The other message tells how many lowercase characters are in the string. The program will ignore all numbers and special characters (punctuation, symbols, etc.).

String Functions

You will need two string functions that were not covered earlier to help with this project:
* `isupper()` - Returns an integer **greater than 0** if the character is uppercase, **0** if the character is not.
* `islower()` - Returns an integer **greater than 0** if the character is lowercase, **0** if the character is not.

Variables

You will need three variables for this project. One variable will count all of the lowercase characters, another to count the uppercase characters, and one for the string itself.

```
int lower_count = 0;
int upper_count = 0;
string my_string = "Roses are Red, Violets are Blue";
```

Iterating Over the String

The next thing to do is iterate over the string and to check each character of the string. A enhanced `for` loop works best.

```
for (char ch : my_string)
```

Checking for Uppercase and Lowercase

It does not matter if you check for an uppercase character first or check for a lowercase character. Let's start with lowercase characters. Ask if the character is lowercase and increment the appropriate counting variable.

```
if (islower(ch)) {
  lower_count += 1;
}
```

What you **do not** want to do is use an `else` statement. This will not give you an accurate count. For example, asking if a special character is lowercase will return `0`. However, that does not mean that it is an uppercase character either. Special characters are neither uppercase nor lowercase. So use an `else if` statement and ask if the character is uppercase. If so, increment the uppercase counting variable.

```
else if (isupper(ch)) {
  upper_count += 1;
}
```

Print the Results

The final step is to print the messages with the count values.

```
cout << "There are " << lower_count << " lowercase characters."
        << endl;
cout << "There are " << upper_count << " uppercase characters."
        << endl;
```

There should be 4 uppercase characters and 21 lowercase characters.

▼ **Code**

```
int lower_count = 0;
int upper_count = 0;
string my_string = "Roses are Red, Violets are Blue";

for (char ch : my_string) {
  if (islower(ch)) {
    lower_count += 1;
  }
  else if (isupper(ch)) {
    upper_count += 1;
  }
}

cout << "There are " << lower_count << " lowercase
      characters." << endl;
cout << "There are " << upper_count << " uppercase
      characters." << endl;
```

Lab 2

Reverse a String

You are going to write a program that takes a string and prints it in reverse order.

Variables

All you need for this program is the string variable and a loop to iterate through the string.

```
string my_string = "The brown dog jumps over the lazy fox";
```

String Iteration

Since we are going to reverse the order of the string, we will need to start at the end of the string and iterate back to the front. Unfortunately, an enhanced `for` loop will not help us in this case because it only iterates from left to right. However, we can still use a regular `for` loop.

The `for` loop should start at the back `my_string.length()-1` and run as long as `index` is greater than or equal to 0. After each iteration, the iterating variable should also decrement by `1` to allow the loop to reverse.

```
for (int i = my_string.length()-1; i >= 0; i--)
```

Reversing a string comes down to taking the character from the end printing that first, then go backwards. This will be done by accessing the indices with `at()`.

```
my_string.at(i);
```

Printing the result

All that's left to do is print. Remember **not** to include `endl` or the system will print a newline after each character.

```
cout << my_string.at(i);
```

You should see `xof yzal eht revo spmuj god nworb ehT`.

▼ **Code**

```
string my_string = "The brown dog jumps over the lazy fox";

for (int i = my_string.length()-1; i >= 0; i--) {
  cout << my_string.at(i);
}
```

Lab 3

Swapping the Case of Characters

You are going to write a program that takes a string and prints a new string where all of the uppercase letters become lowercase, and the lowercase letters become uppercase.

Variables

You are going to need two string variables. The first string variable represents the original string and the second represents the modified string. For now, the modified string can be empty.

```
string original_string = "THE BROWN DOG JUMPS over the lazy
        fox!";
string modified_string;
```

String Iteration

It does not matter if you start at the beginning of the string or the end for iteration. An enhanced `for` loop is the easiest way to iterate through the `original_string`. Set the iterating variable as `ch`.

```
for (char ch : original_string)
```

String Functions

You are going to use the `isupper()` and `islower()` functions to test if a character is uppercase or lowercase. In addition, you will be using the `toupper()` and `tolower()` functions to convert characters to their new cases.

Conditional

For consistency, we will test if a character is lowercase first. However, you may choose to test for uppercase first. It does not matter as long as the conversion is correct.

```
if (islower(ch))
```

If this is true, then append the uppercase version of the character to the variable `modified_string`.

```
modified_string += toupper(ch);
```

If the conditional is false, then append the lowercase version of the character to `modified_string`.

```
else {
  modified_string += tolower(ch);
}
```

You do not need to worry about special characters. Converting them to uppercase or lowercase has no effect.

Printing the Results

Once the loop has finished, print both the original string and the modified string.

```
cout << "The original string is: " + original_string << endl;
cout << "The modified string is: " + modified_string << endl;
```

You should see the following output:

```
The original string is: THE BROWN DOG JUMPS over the lazy fox!
The modified string is: the brown dog jumps OVER THE LAZY FOX!
```

▼ **Code**

```
string original_string = "THE BROWN DOG JUMPS over the lazy
      fox!";
string modified_string;

for (char ch : original_string) {
  if (islower(ch)) {
    modified_string += toupper(ch);
  }
  else {
    modified_string += tolower(ch);
  }
}

cout << "The original string is: " + original_string << endl;
cout << "The modified string is: " + modified_string << endl;
```

Lab 4

Count the Vowels

You are going to write a program that counts the number of vowels that appear in a string. For the purpose of this exercise, vowels are upper and lowercase `a, e, i, o, u`.

Variables

For this project, you will need three variables. One will be the string. Another will be a char to represent each character of the string. The final variable will be a count of all the vowels.

```
string my_string = "The Brown Dog Jumps Over The Lazy Fox";
char ch;
int count = 0;
```

String Iteration

Use a `for` loop to iterate through the string. Then set `ch` to check every character in the string.

```
for (int i = 0; i < my_string.length(); i++) {
  ch = my_string.at(i);
}
```

Checking for a Vowel

Use a conditional to see if the characters in the string are equal to any vowels. Make sure to account for both uppercase and lowercase vowels.

```
if (ch == 'a' || ch == 'e' || ch == 'i' ||
    ch == 'o' || ch == 'u' || ch == 'A' ||
    ch == 'E' || ch == 'I' || ch == 'O' ||
    ch == 'U') {
```

Note that characters are wrapped in single quotes `' '`, not double quotes in C++.

Incrementing the Counter

If `ch` equals any of the vowels, increment the `count` variable.

```
count += 1;
```

Printing the Result

The string may contain no vowels, one vowel, or more than one vowels. Thus, you'll need conditionals to output the appropriate responses.

```
if (count == 0) {
  cout << "There are no vowels in the string." << endl;
}
else if (count == 1) {
  cout << "There is 1 vowel in the string." << endl;
}
else {
  cout << "There are " << count << " vowels in the string." <<
        endl;
}
```

You should see that there are 9 vowels in the string.

▼ **Code**

```
string my_string = "The Brown Dog Jumps Over The Lazy Fox";
char ch;
int count = 0;

for (int i = 0; i < my_string.length(); i++) {
  ch = my_string.at(i);
  if (ch == 'a' || ch == 'e' || ch == 'i' ||
      ch == 'o' || ch == 'u' || ch == 'A' ||
      ch == 'E' || ch == 'I' || ch == 'O' ||
      ch == 'U') {
    count += 1;
  }
}

if (count == 0) {
  cout << "There are no vowels in the string." << endl;
}
else if (count == 1) {
  cout << "There is 1 vowel in the string." << endl;
}
else {
  cout << "There are " << count << " vowels in the string." <<
        endl;
}
```

Lab Challenge: Vowel Replacement

Replacing Vowels with a *

You are going to write a program that takes a string called `my_string` and returns the string but with a `*` in the place of vowels. Assume that vowels are upper and lowercase `a, e, i, o, u`. For example, if `my_string = "Hello"`, then your program will print `"H*ll*"`.

Some of the code has already been filled out for you. Your task is to complete the program so that it produces some of the sample output below. If you accidentally change anything from the original code, you can copy and paste the code back into the text editor.

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

int main(int argc, char** argv) {

  string my_string = (argv[1]);
  char ch;

  //add code below this line



  //add code above this line

  return 0;

}
```

Compile and test your code with a few different values

▼ **Expected Output**

H*ll*

▼ **Expected Output**

ppl

▼ **Expected Output**

```
W trm ln!
```

Requirements

- You **should not** make any changes to the code that already exists. If you accidentally delete any existing code, you can copy and paste the entire program from above.
- You can use **any** number of string functions and conditionals to produce the desired output.