PIC10A 1C. Week 6b Problems. TA: Eric Kim. [Solutions]

1. Vowel Counter

Write a program that, given a user-inputted string, counts the number of vowels. Assume that the user only inputs text in **lowercase**. For instance, here is an example expected output:

```
    Please enter a word: apple
    The word "apple" has 2 vowels.
```

[Solution]

```
#include <iostream>
#include <string>
using namespace std;
int main () {
  cout << "Please enter a word: ";
  string wd;    cin >> wd;
  for (char c : wd) {
      if ((c == 'a') || (c == 'e') || (c == 'i')
          || (c == 'o') || (c == 'u')) {
          cnt = cnt + 1;
      }
  }
  cout << "The word \"" << wd << "\" has " << cnt << " vowels.";
  cin.ignore();    cin.get();
  return 0;
}
```

2. Loopy

Rewrite the following while loop as an equivalent do-while loop and for loop:

```
int i = 0;  int n = 20;  int acc = 0;
while (i < n) {
    acc = acc + (2*i);
    if (acc > 10) {
        break;
    }
    i += 1;
}
// As a do-while loop: INSERT CODE BELOW
```

[Solution]

```
int i = 0;  int n = 20;  int acc = 0;
do {
    if (i >= n)
        break;
    acc = acc + (2*i);
```

```
    if (acc > 10)
        break;
    i += 1;
} while (i < n);

// As a for loop: INSERT CODE BELOW
```
[Solution]
```
int i = 0;  int n = 20;  int acc = 0;
for (i = 0; i < n; i += 1) {
    acc = acc + (2*i);
    if (acc > 10)
        break;
}
```

Part 2: Louis Reasoner suggests the following answers:

```
do {
    acc = acc + (2*i);
    i += 1;
} while ((i < n) && (acc <= 10));

for (int i = 0; i < n; i += 1) {
    acc = acc + (2*i);
    if (acc > 10)
        break;
}
for (int i = 0; (i < n)&&(acc <= 10); i += 1) {
    acc = acc + (2*i);
}
```

Are these equivalent to the original while loop? If so, explain why. If not, describe why not.
Hint: consider the values of acc and i at the end of the original while loop.

[Solution]

All are not equivalent. Note that, after the original while loop terminates, acc will be 12, and i will be 3.

After the do-while loop terminates, i will be at 4. This is because the do-while loop increments i before checking if (acc > 10), whereas the original while loop checked (acc > 10) first.
Also, the original while loop will never run the body if (i >= n), ie if i starts off at, say, 40. However, do-while loops always run the body at least once.

The first for loop is incorrect because, after the for loop terminates, i is 0. This is because we create a new "int i = 0" within the for loop's scope, which disappears once the for loop terminates.

The second for loop is incorrect because i is incremented to 4 (instead of 3). This is because, like in the do-while loop, the condition (acc <= 10) is checked *after* incrementing i += 1, whereas the original while loop checks (acc < 10) before incrementing i += 1.

3. Forward Back

Write a program that, given a user-inputted string, repeats the string in the following pattern:

```
Please enter a word: Apple
A
 p
  p
   l
    e
   l
  p
 p
A
```

Hint: Recall the <iomanip> library, ie: setw(), setfill().

[Solution]

```
#include <iostream>
#include <iomanip>
using namespace std;
int main() {
    cout << "Please enter a word: ";
    string wd;    cin >> wd;
    /* Print it forward once */
    for (size_t i = 0; i < wd.size(); ++i) {
        cout << setw(i+1) << setfill(' ') << wd[i] << endl;
    }
    /* Print it backwards once */
    // Start i at 1 to not repeat last letter
    for (size_t i = 1; i < wd.size(); ++i) {
        int bi = wd.size() - i - 1; // Backward index
        cout << setw(bi + 1) << setfill(' ') << wd[bi] << endl;
    }
    cin.ignore();
    cin.get();
    return 0;
}
```

4. Debugging

Louis Reasoner wants to write a program that, given a user-inputted string, outputs it in reverse order. He thinks to himself, "A-ha! I'll just write a for-loop that traverses the string in reverse order!". He writes the following program:

```
int main() {
    cout << "Enter a word: ";
    string wd;    cin >> wd;
    for (size_t i = (wd.size()-1); i >= 0; --i) {
        cout << wd[i];
    }
    return 0;
}
```

When he runs the program, the program correctly outputs the string in reverse order, but then quickly crashes. The error message complains about a string index out of bounds error. What could be the problem? Can you fix the code?

[Solution]

Recall that size_t is an unsigned type! When i becomes 0, and we output the first character of wd, the for loop then decrements i by 1. Since i is unsigned, it can't take on negative values - thus, it instead wraps around to the largest positive value (ie ~4 billion), and since a large positive value is greater than or equal to 0, we run the body again, causing an error when we try to access the ~4 billion-th entry of wd.

To fix this bug, we can do the following:

```
    for (size_t i = (wd.size()-1); i > 0; --i) {
        cout << wd[i];
    }
    cout << wd[0]; // Handle separately
```

Or, we can do the following:

```
    for (size_t i = 0; i < wd.size(); ++i) {
        size_t bi = wd.size() - i - 1; // Backwards index
        cout << wd[i];
    }
```