PIC 10A 1C. Week 6b Exercises. TA: Eric Kim. [Solutions]

1. Be the Compiler

Determine the output of the following code snippets. Assume that all libraries are included, and we are using the standard namespace. If there is an error, explain the error.

int N = 5;
for (int i = 0; i < N; ++i) {
  cout << i * 2 << " ";
}
	Output:
	0 2 4 6 8

string s = "meow";
for (size_t i = 0; i < s.length(); ++i) {
  if ((i % 2) == 0)
  s[i] = 'a';
  else
  s[i] = 'b';
  i += 1;
}
cout << s;
	aeaw

int i = 1;
while ((i % 2) == 0) {
  i *= 3;
}
cout << i;
	1

int i = 1;
do {
  i *= 3;
} while ((i % 2) == 0);
cout << i;
	3

string s = "raptor";
for (char& c : s) {
  if (c == 'a')
  c = 'e';
  if (c == 'e')
  c = 'i';
}
cout << s;
	riptor

string s = "captain";
for (char c : s) {
  if (c == 'a')
  c = 'o';
}
cout << s;
	captain

2. Apples and Oranges

Consider the following class interfaces:

```
class Apple {
private:
  int mya;
public:
  int myb;
  Apple();
  Apple(int a);
  void foo(Apple a);
};
```

```
class Orange {
public:
  int myc;
  Orange(int c);
  void garply(Apple a, Orange b);
};
```

Consider the following code. Are there any issues?

```
Apple apple1(2);
Orange orange1(3);
Apple apple2(orange1.myc);
Orange orange2(apple2.mya);
garply(apple1,orange1);
orange1.foo(apple1);
orange1.garply(Apple(1), Orange(2));
```

[Solution]

```
Apple apple1(2);
Orange orange1(3);
Apple apple2(orange1.myc);
Orange orange2(apple2.mya); // mya is private! can't access
garply(apple1,orange1); // need to call garply on Orange object, ie
    // orange1.garply(apple1,orange1);
orange1.foo(apple1); // foo is not a method of the Orange class,
    // is a method of the Apple class.
orange1.garply(Apple(1), Orange(2));
```

3. const Practice

Identify any possible issues with the code:

[Solution]

```
int a = 1;
const int b = 1;
int& c = a;
int& d = b; // ERROR: Can't point non-const reference to const.
c = 0;
const int& aa = a;
const int& bb = b;
aa = 4; // ERROR: Can't modify a const reference
bb = 4; // ERROR: Can't modify a const reference
```

4. Robbers Robbing Robbers

Define a Robber class interface that satisfies the following code:

```
Robber rusty("Rusty");
Robber dan("Dan")
rusty.greet(dan);
dan.greet(rusty);
int item_to_steal = dan.lookat(rusty);
dan.steal(rusty, item_to_steal);
cout << rusty.yell();
```

[Solution]

Note: Many of the method return types are ambiguous (ie Robber::greet could return, say, an int, and still adhere to the above code), but here's one valid interface:

```
class Robber {
public:
    string myname;
    Robber(string name);
    void greet(Robber r);
    int lookat(Robber r);
    void steal(Robber r, int i);
    string yell();
};
```

5. 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;
 unsigned int cnt = 0;
 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;
}
```

6. 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.

7. 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) {
       size_t bi = wd.size() - i - 1; // Backward index
       cout << setw(bi + 1) << setfill(' ') << wd[bi] << endl;
   }
   cin.ignore();
   cin.get();
   return 0;
}
```

8. 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];
   }
```

9. Search and ~~Destroy~~Replace

Write code that, given three strings sent, s1, s2, performs the following:

Wherever s1 is found within sent, replace s1 with s2.

Assume that s1, s2 are the same length. Examples:

`string sent = "i am happy";` `string s1 = "i";  string s2 = "u";`	`Output:` `"u am happy"`
`string sent = "apples to apples";` `string s1 = "app";  string s2 = "boo";`	`"booles to booles"`

[Solution]

```
// v1: using string::find
  string sent = "apples to apples", s1 = "app", s2 = "boo";
  string cur = sent;
  // offset keeps track of where in s1 cur is located
  size_t offset = 0;
  size_t loc = cur.find(s1);
  /* Loop while s1 is within cur */
```

```
  while (loc != string::npos) {
    /* s1 is found, replace it with s2 */
    // istart, iend index into sent, not cur!
    size_t istart = offset + loc;
    size_t iend = istart + s2.size();
    for (size_t i = istart, j = 0; i < iend; ++i, ++j) {
      sent[i] = s2[j];
    }
    cur = cur.substr(loc+s2.size()); // advance tmp
    offset += loc + s2.size(); // update offset
    loc = cur.find(s1); // Repeat search for s2
  }



// v2: Without string::find(string s)
  string sent = "apples to apples", s1 = "app", s2 = "boo";
  size_t i = 0;
  while (i < sent.size()) {
    bool is_found = sent.substr(i,s1.size()) == s1;
    if (!is_found) {
      /* s1 not found, advance i */
      i += 1;
    } else {
      /* s1 found, replace with s2 */
      for (size_t j = 0; j < s2.size(); ++j) {
        sent[i+j] = s2[j];
      }
      i += s1.size(); // can skip by s1.size(), rather than just by 1
    }
  }
```