LAB #6
Debugging/Getting Caught up

I’m aware that many students feel behind right now, and so I will allow some time in lab #6 to be dedicated to you finishing lab#5!!! Make sure you get lab #5 checked off by the end of you lab #6 time.

Remember, if you need to get Lab #5 graded, you need to show your lab to the TAs within 10 minutes of getting to lab, and you and your partner will not receive lab credit if you do not get checked off before leaving each lab. Once you have a zero on a lab, then it cannot be changed because we have no way of know if you were there or not!!!

Reminder: All of our labs involve paired programming. You do not have to keep the same partner for each lab, but you MUST work with someone in each lab!!! First, find a partner for this lab. It can be the same partner from the previous lab or a different partner.

(2 pts) Conditional Compilation: One of the useful features of it is the ability to conditionally include code, based on macro definitions. For instance, this is often referred to as a DEBUG macro:

```
#ifdef DEBUG
/* your debug code here */
#else
/* your non-debug code here, if differences exist */
#endif
```

OR

```
#ifndef DEBUG
/* your non-debug code here, if differences exist */
#else
/* your debug code here */
#endif
```

In some cases, you have different function calls for debugged versions, in other cases, debugging code is simple output statements. For this task, ensure that all of your print statements are wrapped in DEBUG macros. Now, you can #define DEBUG to turn it on, and comment this statement to turn it off OR you can compile with a –D DEBUG to define it or leave it out.

```
g++ prog.cpp –D DEBUG
```

(6 pts) Debugging and Errors. Here are some example bugs, and you need to find the bug, correct the bug, match the bug to the error below, and explain how you found the bug and corrected the bug.

Bug 1: http://classes.engr.oregonstate.edu/eecs/spring2015/cs162-001/labs/bug1.tar

Bug 2: http://classes.engr.oregonstate.edu/eecs/spring2015/cs162-001/labs/bug2.tar

Bug 3: http://classes.engr.oregonstate.edu/eecs/spring2015/cs162-001/labs/bug3.tar

Bug 4: http://classes.engr.oregonstate.edu/eecs/spring2015/cs162-001/labs/bug4.tar

To untar the files, `tar –xvf bug1.tar`

Types of Errors to Find/Correct/Match

- Off by one error in an array
- Initialize a different object than the one you are using
- Forget “Big Three” with dynamic member variable
- Access memory that hasn’t been allocated

1. **Trace code to say what it does (plain English)**

2. **How do you know what the code is doing? What is it supposed to do?**

3. **Fix the errors and show the fixes to your TA.**

(2 Pts) Let’s learn about exceptions.
In an OO Language, we get two ways to catch some errors. Since a string object is an empty string, then we most certainly want to check that size is greater than 0 before accessing an element. For instance, say we want to print back the first initial of a user’s name, but we forgot to prompt them☺ We do not want to print the initial if the user’s name is size 0. We can do a check manually:

```
if(name.size() != 0)
   cout << “Hello “ << name[0] << endl;
else
   cout << “Uh Oh, You didn’t initialize me!” << endl;
```

Or we can use the at() function that does a check and returns an exception when we go outside the bounds, i.e. 0 technically doesn’t exist.

```
#include <stdexcept>
…


try {
   cout << “Hello “ << name.at(0) << endl;
}
catch(out_of_range &e) {
   cout << “Uh Oh, You didn’t initialize me!” << endl;
}
```