Quiz 2 - Due Tuesday January 31st 11:45 PM

College of San Mateo

Computer Science Department

CIS 279 – (CS2) Data Structures: C++

Quiz Exercise # 2

Debugging

Debugging is easy to overlook – until you cannot figure out what is wrong with your code.

In this exercise you will use some rudimentary testing skills to get you back in the swing of things by finding bugs in a piece of code, not through code inspection, but by choosing test cases well. This is an invaluable skill.

There are two high-level approaches to testing code: the code coverage approach and a black-box approach.

The code coverage approach is based on explicitly evaluating the code itself, making sure you have proper testing coverage. This means that there must be a test that covers every path through the code (assuming your code has several if statements / loops / etc.). This requires you to look at the code.

The other approach is to treat the code as a black box, meaning that you cannot look *into* it. This in turn puts all of your focus on the input and how you think it will exercise different aspects of functionality.

Before getting started on this assignment become familiar with the topics discussed in Section 1.3 *Testing and Debugging* of our textbook.

Objectives:

The following are the primary objectives of this exercise session:

- Understand debugging tasks using black box testing (i.e. select inputs for your test cases to exercise program functionality).
- Get up and running with your *'automated'* debugging skills (i.e. be able to use purposefully selected input to generate a test driver program).

Testing should always be methodical, never just randomly trying out test cases. For each input, make sure you think of all of the different, meaningful ways that input could change. For example, if you have an integer, think of testing with a normal size integer (i.e. 5), then a large integer (i.e. 10000), then zero, then a negative integer (i.e. -5), then a small integer (i.e. -10000).

People
Participants
Activities
Assignments
Forums
Resources
Administration
Course administration
Quickmail
Compose New Email
Signatures
View Drafts
View History

If there are multiple inputs, try all combinations of interesting inputs (both positive, 1st positive and 2nd negative, etc.).

If there is a range of acceptable inputs (1 to 10, for example), then you need to test at the borders and just across the border. For example, you would test values 0, 1, 10, and 11.

For any code based on user input, there is another category of inputs –*ill-formed* inputs. So make sure that you also think about inputs that are not what the program expected.

For this exercise, the program you are debugging is a square function. That is if you give the input 5, it will output the number 25.

Identify the ranges.

For this problem there are four ranges of numbers:

- Range A: negative inputs whose results is too large to fit (error)
- Range B: negative inputs whose result is acceptable (normal)
- Range C: positive inputs whose result is acceptable (normal)
- Range D: positive input whose result is too large to fit (error)

In order to construct a thorough test suite you may consider the following test cases:

For each range of acceptable answers, create 6 test cases: Three test cases that are relatively small and easy to calculate, then 3 test cases that are very large numbers.

For each border between ranges, create a test case for each side of the border. That is have a test case that is the input in range A next to range B, in range B next to range A in range B next to range C, in range C next to range B, in range C next to range D, and in range D next to range C.

As you can see *planning* test cases is needed to construct a comprehensive automated test suite.

Now there is only one thing left to do – try to find the bugs in this code!

```
int sqr ( int n ) {

    return n * n;

}
```

Deliverable: For each error range, determine a single test input. Your submission will include 4 test cases providing 1 input that satisfies each of the ranges shown below:

- Range A
- Range B
- Range C
- Range D

Submit your test inputs in a **.txt** file.

To illustrate, you are determining values for an int variable **n** as would be exercised in a code snippet such as shown below:

```
#include <iostream>

using namespace std;

int sqr(int); // Function declaration

int  main()

{

  //given a call sqr ( n ) provide test     //cases for integer input n

   return 0;

}

/* --------- OUTPUT -----------

List test cases used – need one test case from each range listed above (A-D).

----------------------------  */
```

Homework Guidelines:

- **Be sure your name is included at the top of your .txt file.**
- **Label the test range that your test input satisfies.**

Grading

This exercise is worth 1 point:

Test inputs provided (4 total test inputs; 1 input required that satisfies each of the following ranges):

- Range A (.20)
- Range B (.20)
- Range C (.20)
- Range D (.20)

Meets Homework Guidelines (.20)

Questions? Post your questions in the forum discussion.

Submission status

Submission status	No attempt
Grading status	Not graded
Due date	Tuesday, January 31, 2017, 11:45 PM
Time remaining	5 days 5 hours
Last modified	-
Submission comments	Comments (0)

Add submission

Make changes to your submission