Reading and Testing

Preventing, detecting, and removing bugs is difficult. We spent two labs discussing and reading code. The two main tools were reading and testing.

As you start work on the next project, boggle, you have a chance to put the ideas from lab into practice.

You also can learn from common mistakes people made in the first assignments.

Common Memory/Pointer Errors

The two most common memory/pointer errors were

• Not checking for NULL
• Memory leaks

In detail:

NULL Pointers

Code that looks like:

```
if ( p->val )
```

will crash the program if pointer p is NULL. The solution is to check the pointer before using the value of the pointer.

Write Readable Code: Keep your functions short, write comments to explain what you are doing. When your program does not work, you will want to know what each function was supposed to do. Assume you will need to read the code, so make the code readable and commented.

Read Your Code: One of the debugging techniques is to read your code. You can prevent NULL pointer bugs by reading each of your functions looking for code that uses pointers. Look at each use of a pointer. Ask yourself what would happen if that pointer had a NULL value. Add code to test for NULL and take the correct action.

Test Your Code: Write functions in main.cpp that call your functions with NULL pointers. See if your functions crash. If so, correct the functions. An alternative is to comment the function to say the function should never be passed a NULL pointer and then make sure that all the clients of that function obey that rule.

Memory Leaks

A common error has been code like:

```
Node* p = new Node;
p = head->next;
```

This code will not crash the program. This code *will* allocate a block of memory and then lose the address of that block. Over time, the program will reserve more and more memory but will be unable to use or recycle that memory.

Read Your Code: These bugs are not hard to find if you know what to look for. Look for places where you call new. Then ask yourself what you do with the address new returns. If you over-write the variable holding the address, you have a memory leak. If you never call delete for that address when you are done using the block of memory, you have a leak.

You can also use valgrind, but learn to read your code.

Test Your Code: There is no easy test to find memory leaks. Read your code and comment it as you go. Explaining in comments what you are doing can help you find these bugs

Word Lookup Using a Trie

In the second lab on program review, we discussed programs that were incorrect but produced correct output. A stopped clock is correct twice a day, but you will only learn it is not working if you test it in enough cases.

The requirement of this first part of the assignment is to build a data structure and write code to store words and look up words. Quickly.

Read Your Code: Write clear contracts for each function and then read the code to see if your function does what your comments promise.

Test Your Code: Make a list of tests for your functions. Testing every possible value is impossible. As we discussed in lab, though, it is possible to test types of values. For example, make sure your function works for:

• The empty string
• Duplicate strings
• UppeR aNd lOWer case
• with.or-without-punctuation?
• ... add more here