Testing and Debugging with Pin

For the first assignment in this class, I like to ask students to brainstorm for a while based on their own experience. In previous semesters, the focus has been on debugging use of dynamic memory, but I thought that it might be more fun to open the problem space a bit more this semester and to try to leverage an opportunity.

The introductory programming class in ECE (190) recently replaced paper exams with coding exams. As a result, the instructors would like to have tools that automatically evaluate student responses so as to provide feedback that is both timely and fair. Evaluation of correct functional (input-output) behavior is not particularly hard, nor is it particularly difficult to write code that wraps a student’s implementation of some program with a number of test executions and compares exact answers. The problem is that roughly a third of students get no credit from purely functional tests, even though their program is often fairly close to being correct (as judged later by a human). Some of these problems are syntactic, and we probably need to make use of a compiler infrastructure such as LLVM or GCC to check whether the mistakes are minor. We’ll leave those for now. Students also make a number of subtle logical and semantic errors, however, that result in their programs’ failing completely.

So here’s the challenge: given a working program (a solution) and a tool that allows you to watch and modify a test program’s execution, can you make it easy to express common problems and to test for them? Can you also provide information that makes it easy for a designer to identify symptoms or causes of errors and to create rules to check for them?

I was thinking about the problem over the last few weeks and thought that it might be fun to have you do the same. If the answer to the questions just posed is yes, we might not only be able to solve 190’s immediate problem, but we might also be able to create a fairly effective tool for helping people to learn to program on their own. The tool, after all, can be used to create tests for many small code examples, which can be provided online along with directed feedback for the user.

For the purpose of observing dynamic program execution, we can use Pin, a dynamic binary instrumentation tool developed by Intel. The Pin tool allows you to statically and/or dynamically modify a program to add and remove instrumentation in the form of general callbacks to your code. In an extreme case, you could have your code executed before each instruction in the program, but obviously this kind of interaction produces a dramatic slowdown. Pin is written in C++, so you’ll also get a fairly gentle introduction to C++ interfaces in case you haven’t used the language before.

The tool and documentation is available at www.pintool.org

The assignment as a whole will consist of a design, implementation, and possibly an application. You won’t need to download Pin until we start implementing, but feel free to grab a copy to play around with it. If you do write start writing code, be aware that I will limit groups to four people, so don’t pass code around unless you know with whom you’re going to work.

Your task in this homework is to brainstorm about the kinds of errors that people make and how you can detect and identify them using a tool like Pin.
**Examples:** Here’s an example of a problem from a midterm along with several ways in which students made errors. Students were asked to write a function that created a reversed copy of a C string. A number of students did not produce code that worked, but closer inspection revealed that in many cases fairly small oversights or lack of knowledge led to incorrect results. For example, some students failed to design their loops properly (although they were close), or simply could not remember how to express ASCII NUL in order to identify the end of the string being reversed. Perhaps the most interesting one is the off-by-one error in which students also moved the terminal NUL to the front of the reversed string, which always produces an empty string as the result (followed by a nearly correct answer!).

Another set of questions focused on linked lists. Students were given the following node definition:

```c
typedef struct node_t node_t;
struct node_t {
    int     data;
    node_t* next;
};
```

The students were then asked to write one of the following functions:

- // returns 1 if list element data fields are in increasing order, or 0 otherwise
  int is_sorted (node_t* head);

- // returns the number of list elements with data == item
  int count_occurrences (node_t* head, int item);

- // returns 1 if any two (or more) list element data fields have the same
  // value, or 0 otherwise
  int contains_duplicates (node_t* head);

In these sorts of problems, ideally one could give a student some partial credit for knowing how to walk the linked list, even if the student’s answer does not actually produce any reasonable result. To test that property, a Pin-based tool could simply check that the code accessed the head and each of the next pointers, possibly constrained to the proper order.

Generally, students in 190 might be asked to solve simple problems involving control flow (loops, conditionals, and so forth), arrays, pointer-based data structures, I/O using C’s standard library, use of problem-specific auxiliary routines, dynamic allocation, and recursion. They are typically expected to handle some or all error responses from function calls.

**The Assignment:** Brainstorm about the kinds of tests that you might be able to create using Pin’s capabilities, then write up a page (or a bit more if you’re going to work in groups, which I encourage you to do) discussing how you can express these tests and how you can use Pin to evaluate them. You should also try to organize your ideas according to the type of problem being solved.

In your writeup, try to give specific examples of how one can execute the testing process with Pin and how accurate one can expect the results to be. For example, for each type of test you may want to perform, what actions will you need to monitor (certain types of instructions, specific function calls, etc.)? Can you guarantee that the test result is correct? That is, to what degree can a user confidently assert that the student did or did not meet the test criterion based on the automated result using Pin? Discuss reasons that the answer might sometimes be no, for either false positive or false negative results or both. Also think about design tradeoffs, particularly between incurring substantial overhead (by tracking every instruction, for example) as opposed to minor overhead. This last issue is not major when testing small amounts of student code, but becomes important if one wants to generalize the ideas for other uses. Finally, can you generate any feedback to the student that might help them understand what they’re doing wrong in their program?