OMSCS 6340 - Fall 2016
Assignment 7 (100 points)
Due at: November 28, 8am EDT

Ob jective:
The objective of this assignment is to learn the debugging technique called Cooperative Bug Isolation (CBI) or Statistical Debugging.

Resources:
Cooperative Bug Isolation webpage: http://pages.cs.wisc.edu/~liblit/dissertation/. Download the archive `asgn7.zip` and decompress it. Under directory `asgn7/` you should find:

- `testimg.jpg`, a test image,
- a directory `src` containing the source code of a buggy version of `jpegtran`, and
- a directory `analysis` containing the results of running CBI on `jpegtran`.

Setup:
The setup commands assume a UNIX-like environment. First build `jpegtran` as follows:

```
cd src
CC='gcc -g' ./configure
make
```

You will now have (among other things) a `jpegtran` executable that is compiled with debugging support. Later in the assignment, you will run this executable to discover test cases that cause the program to crash, as well as run this executable with the debugger `gdb` (`clang` and `lldb` are also acceptable for OS X users).

We ran the CBI tool on `jpegtran` using numerous test image files as input. One such image file, `testimg.jpg`, is provided with this assignment. The output produced by the CBI tool is provided in the `analysis` directory. Your starting when inspecting these results should be the file `all_hl_corrected-exact-complete.html`, which may be viewed in a web browser.

Problem:

1. Examining just the CBI results, how many bugs do you believe are represented by this report? Justify your answer. You may find it helpful to look at the affinity lists of the selected predictors (recall that the affinity lists show other predicates that are highly correlated with the selected predicates, with the most highly correlated predicates listed first).
2. For each bug represented in the CBI results, construct a failing test case that exhibits that bug. A "test case" is a selection of switches to pass to jpegtran. The base test case is:
   `{PATH TO EXECUTABLE}/jpegtran -outfile testout.jpg testimg.jpg`
   Use the copy of jpegtran that you compiled with debugging support. You can create other test cases by adding additional switches to jpegtran. A complete list of switches can be obtained

by executing `jpegtran -h`. The manual page (`man jpegtran`) also provides a good summary of the interface.
Your failing test cases should use only the supplied test image `testimg.jpg` as the input data. Report the command line that you use to reproduce each bug and the result of the execution.
Note: `jpegtran` might be pre-installed on your system. You must specify the complete path to the executable that you compiled to prevent running the pre-installed version.

3. Give a succinct description of the cause of the bug associated with the top-ranked (first) predictor for each bug; what does the program do wrong that eventually causes the failure? To answer this question you will probably need to refer to the CBI results, the failing test case for the bug, and the program source. You may also find it useful to step through parts of the execution of your test case in a debugger such as `gdb` or `lldb`.

Items to Submit:
Upload a file named `report.txt` with answers to the above problems.

Assignment Notes and Tips:

- gdb and lldb are fairly complex programs; if you are not already familiar with debugging C programs in a UNIX environment, you may be better off without them.
- Note that we are looking for the cause of the bug or bugs, not the symptoms of them. For example, “the program dereferences a null pointer, causing a segmentation fault” is not an acceptable root cause - what logical error was made which caused that pointer to be null when it should not have been, or that caused the program to dereference a pointer that was correctly set to null?