OMSCS 6340 - Fall 2016
Assignment 8 (100 points)
Due on: December 5, 8:00am EST

Objective:
The objective of this assignment is to learn the debugging technique called Delta Debugging.

Resources:
Delta Debugging webpage: http://www.st.cs.uni-sb.de/dd/.
After decompressing asgn6.zip, you should find the following files under `asgn8`:

- `delta-2003.7.14.tar.gz` , the delta debugging tool.
- `mysort.c`, a buggy integer sorting program.
- `big_failing_test.txt` , an input list of integers, one per line, on which the buggy integer sorting program fails.
- `Makefile`, a build file using Make for `mysort`.

Setup:
We assume a UNIX-like environment. Install the delta debugging tool by untar-ing `delta-2003.7.14.tar.gz` using the command `tar -xvf delta-2003.7.14.tar.gz` .
We will be using the program `delta/bin/delta` in the unpacked directory. It is a perl script; no building is necessary but your system needs to have perl installed. See `delta/README` after unpacking for more information.
Execute make and verify that mysort compiles. Its usage is: `mysort <input list> <timeout>` where `<input list>` is the name of a text file containing a list of integers, one per line, to be sorted, and `<timeout>` is a positive integer denoting the number of seconds for which the sorting routine should run.
In the case of a timeout, mysort will print the message `Error: timeout.` Ensure that `<timeout>` is reasonably large so that the program times out only due to infinite looping, not large inputs. Generally, a couple of seconds should suffice. Setting the timeout too high may mean that delta takes a long time to reduce the test input.

Problems:

1. The bug in the sorting function `sort` in `mysort.c` manifests when `mysort` is run with command line arguments consisting of a reasonably large timeout and the file `big_failing_test.txt` . It fails to produce a sorted list of integers that is a permutation of the list of integers in `big_failing_test.txt` . However, as its name suggests, `big_failing_test.txt` is not the minimal test case on which mysort fails. Use the delta debugging tool to obtain a minimal test case on which mysort fails for the same reason as on `big_failing_test.txt` . You will need to design a script, called `script`, and run the command `delta/bin/delta -test=script -cp_minimal=min_failing_test.txt < big_failing_test.txt` If `script` is written correctly, this will create a minimal test case named `min_failing_test.txt` .

You may run into problems if any directory along your file path has a space in its name. You should also read the file `delta/doc/using delta.txt` (except the sections "Topformflat" and "Multidelta" since we are using line granularity and running the tool on a single file). **Example scripts in the documentation are shell scripts, but you are free to use any scripting language you like provided your script is self-executable and runs with the exact command provided above.** If you do choose to write a shell script, be sure to specify the exact shell to use (e.g., `/bin/bash` or `/bin/zsh`) since `/bin/sh` refers to different shells on different machines. Note that delta runs the test script in a subdirectory like `tmp0/arena/`. You may need to take this into account when referring to the `mysort` program in your script. We should be able to run the above command by placing your submitted files and our own copies of `mysort` and `big_failing_test.txt` in the same directory (so do not use absolute paths, for example).

2. Examine the sorting function `sort` in `mysort.c`, simulating its execution on the list of integers in `min_failing_test.txt`, and answer the following questions:
    - 2.1. Explain the cause of the bug in the source code of the sort function (that is, the reason why mysort fails on `min_failing_test.txt`).
    - 2.2. Characterize (describe the common feature(s) of) all input lists on which the sort function will fail. For example, if the problem were sorting data with overflow, you might say "mysort will fail on any input list containing data larger than a 32 bit integer".
    - 2.3. Describe a fix for the bug. For example, if the bug were a compilation error caused by an extra close quote, a fix could be "delete extra close quote on line ##".

Items to Submit:

Upload the following files to T-Square:

- `report.txt`, a report containing your answers to the questions in Problem 2.
- `script`, the script you wrote for Problem 1.
- `min_failing_test.txt`, the minimal test input produced from Problem 1.