Procedural Programming- COMP1711
Guidelines for Testing Project 3

You should thoroughly test the following critical features of the program:

- No two stars should have the same coordinates.
- No two stars should have the same serial number (id number).
- The x coordinate should be in the range [0-60].
- The y coordinate should be in the range [0-30].
- The name command should find the currently available closest pair of stars, i.e. if a pair of stars has been named it cannot be used gain.
- When the universe is saved to a binary file then loaded again, all stars and named pairs of the original saved universe should be completely and accurately restored.
- All stars should appear within the drawing window.
- The *Plot* and *WriteAt* functions should not cause run-time errors if an attempt is made to plot or write text outside the boundaries of the frame buffer, such as when trying to display the name of a star located at the very bottom of the frame buffer, or if text overflows the right-hand side of the frame buffer.

Note: If a star is located at, or very close to, the boundary of the universe, then a simple implementation of the show command may not display the name of the star. We will accept this slight imperfection for the time being, and you are not required to find a more elaborate way of doing it.

The Test Data

Since this program generates random values for the coordinates of the stars, deterministic test data cannot be used. However, as you recall if the *rand* function is **not** seeded at program start-up, it will always generate the same sequence of pseudorandom numbers, which will permit deterministic test data to be used. Yet, notice that this sequence may differ from one programming environment to the other. The programming environment is a combination of the operating system, the compiler, and the standard libraries used. Therefore, the program should be tested within an identical environment to the one used to generate the test data in the first place.

For this project, the test data was generated using a machine identical to the ones in the DEC-10 lab. To test your program with this data, you will need to do the following:

1- Comment out the call to the srand function in your code.
2- Compile and run your program on one of the machines in the DEC-10 lab using the provided input data file (*proj3in.txt)*.
3- Compare the output of your program with the provided output data file (*proj3out.txt*)

The *proj3in.txt* file contains sample input data. You can feed this data to your program using the Linux *input redirection operator* (<). For example, if your program's executable code is in a file called *a.out*, you can make your program read all the input data from the proj3in.txt file using this command:

./a.out < proj3in.txt

You can also redirect the output of your program to a file using the *output redirection operator* (>) like this:

./a.out > myout.txt

where myout.txt is the name of the file that you want to redirect all the program output to. Finally, you can redirect both the input and output of your program like this:

./a.out < proj3in.txt > myout.txt

You may like to know that the provided input file mimics the following test scenario:

- Create a universe of 20 stars
- List these stars
- Draw the universe
- Name 10 pairs
- Try to name another pair (this will fail because no more pairs are left)
- List all named pairs
- Show the pair of stars for Irma Gobb and Mr Bean (notice we used Irma Gobb's name to find the pair)
- Save the universe to a file
- Create another universe of 100 stars
- Draw this universe
- Name one pair (Mr New and Mrs New)
- Show this pair
- Load the saved universe
- Draw the loaded universe (to make sure it is the one we saved)
- List all stars (again to check that all stars have been restored)
- List all pairs
- Quit

The output of your program may not have the exact format as the sample output file, but it should definitely have the same values for the data.