Cassidy Brown – cmb195
OS HW4 10/3/16

Each design implementation uses two processes to complete the Shubert computations.

- - - - -

```
cassidy@ubuntu:~/os/hw4$ ./fork
37.99 -73.98 49.96 15.89 -27.48 10.73 -10.99 7.29 -5.07
-73.98 144.06 -97.27 -30.95 53.51 -20.89 21.40 -14.20 9.88
49.96 -97.27 65.68 20.90 -36.13 14.11 -14.45 9.59 -6.67
15.89 -30.95 20.90 6.65 -11.49 4.49 -4.60 3.05 -2.12
-27.48 53.51 -36.13 -11.49 19.88 -7.76 7.95 -5.27 3.67
10.73 -20.89 14.11 4.49 -7.76 3.03 -3.10 2.06 -1.43
-10.99 21.40 -14.45 -4.60 7.95 -3.10 3.18 -2.11 1.47
7.29 -14.20 9.59 3.05 -5.27 2.06 -2.11 1.40 -0.97
-5.07 9.88 -6.67 -2.12 3.67 -1.43 1.47 -0.97 0.68
min = -97.27
```

- - - - -

```
cassidy@ubuntu:~/os/hw4$ ./server
37.99 -73.98 49.96 15.89 -27.48 10.73 -10.99 7.29 -5.07
-73.98 144.06 -97.27 -30.95 53.51 -20.89 21.40 -14.20 9.88
49.96 -97.27 65.68 20.90 -36.13 14.11 -14.45 9.59 -6.67
15.89 -30.95 20.90 6.65 -11.49 4.49 -4.60 3.05 -2.12
-27.48 53.51 -36.13 -11.49 19.88 -7.76 7.95 -5.27 3.67
min = -97.27
```

(Concurrently, in a separate terminal window)

```
cassidy@ubuntu:~/os/hw4$ ./client
10.73 -20.89 14.11 4.49 -7.76 3.03 -3.10 2.06 -1.43
-10.99 21.40 -14.45 -4.60 7.95 -3.10 3.18 -2.11 1.47
7.29 -14.20 9.59 3.05 -5.27 2.06 -2.11 1.40 -0.97
-5.07 9.88 -6.67 -2.12 3.67 -1.43 1.47 -0.97 0.68
```

- - - - -

```
cassidy@ubuntu:~/os/hw4$ ./threads
10.73 -20.89 14.11 4.49 -7.76 3.03 -3.10 2.06 -1.43
-10.99 21.40 -14.45 -4.60 7.95 -3.10 3.18 -2.11 1.47
7.29 -14.20 9.59 3.05 -5.27 2.06 -2.11 1.40 -0.97
-5.07 9.88 -6.67 -2.12 3.67 -1.43 1.47 -0.97 0.68
37.99 -73.98 49.96 15.89 -27.48 10.73 -10.99 7.29 -5.07
-73.98 144.06 -97.27 -30.95 53.51 -20.89 21.40 -14.20 9.88
49.96 -97.27 65.68 20.90 -36.13 14.11 -14.45 9.59 -6.67
15.89 -30.95 20.90 6.65 -11.49 4.49 -4.60 3.05 -2.12
-27.48 53.51 -36.13 -11.49 19.88 -7.76 7.95 -5.27 3.67
min = -97.27
```

- - - - -

If I were to design a parallel programming application, I would use threads. Forking required extra conditionals to assure each process only did its own part, and the client-server design required a lot of boilerplate (this method required two files, both of which had a lot of setup code). The threads shared data through global variables, so there was no need to set up pipes or other read/write communication, and the child operated in a separate function, so it was easier to distinguish the two threads.