CS 569 Assignment 1

Yueting Zhu
zhuyue@onid.oregonstate.edu

1. Was the program correct or buggy?
   The program is buggy at first. I wrote the code and did not debug in Visual Studio. Then I wrote the harness and check if the code is correct or not. After fixing, the code is correct. And I changed it to be buggy.

2. What was hard to specify?
   One function calls another function part is hard to specify. Because if there is a buggy there, I do not know which function is buggy. Actually the loops in those functions are difficult to be checked out. I should look into each function to make sure if everything is correct.

3. Was there functionality you could not specify?
   No, because the parameter I use in each function are different with others. So if there is a bug, from the counter examples, I can find the buggy parameter and go through into its function to check out the bug.

4. How long did it take to verify with different loop bounds?
   As the loop bounds are increasing, the time it costs is increasing as well. From my program, it is exactly clear. Because there are many loops in my code with calling relationship. So if I add just several loop bounds, it costs extremely long time more than initial loop bound.

5. How did turning on/off bounds and pointer checker affect cbmc runtime?
   As my test, there is nearly no different on runtime if turning on bounds and pointer checker or not.

6. Discuss the ability of the harness to find the bugs you introduced, and how to address the problem if it did not, including a revised harness to find them.
   As my code is about max prime number under a given input. So in my harness, I check all the numbers that bigger than the result of calling function and check if it is a prime number. Just as the following code:

```
b = maxprime(e);
assert(b <= e);
assert(prime(b));
for (int k = b + 1; k <= e; k++)
    {
            assert(prime(k) == 0);
    }
```

If is shows that assert(b <= e) cannot pass, there is a bug in function maxprime(int). Other bugs, I will check the value of b. If b is not a prime number but it passes assert(prime(b)), then there is a bug in function prime(int). otherwise, there is a bug in the maxprime(e) function. In addition, according to the parameter values, I can lock the location of bugs in the functions.