Lab Exercise 4
11/11/2020

Debugging is an essential skill for any software developer. The simple and naive approach of using debug print statements is inefficient, error-prone, and unreliable in general. Real debugging requires a more sophisticated tool called a *debugger*. In this lab, we'll introduce you to the basic usage of GDB, GNU's Debugger tool.

1. Download the starter code from the Blackboard or clone the exercise repository by executing the following command:

```
$ git clone https://github.com/fnegahbani19/-COMP201-Lab-B-04

```

2. To get started, we'll need to compile the program specifically for GDB by using the `-g` flag, it's specified in the makefile so just run:

   `make install`

3. Check out the code to get an idea of what the program is trying to compute, but do not change anything yet. Try running the program; it takes 1 command line argument: an integer representing number of the first prime numbers. For debugging that's better to keep it small. The default value is 10 if you don't pass any argument variable. For e.g.:

   `./main.out 3`

4. To start the GNU debugger running your program with command line arguments you can use:

   `gdb --args main.out 5`

   Alternatively, you can start GDB without args using `gdb main.out` and once it has started, you can set (and reset) the argument(s) using `set args 5`.

5. After you have GDB prompt and you were able to run the code using debugger, start looking for bugs and fix them. Remember that you can check the reference card that is sent to you. You have many options like setting breakpoints, executing line by line, checking the variable values, and etc.

6. In the end, please submit your fixed code to the Blackboard.

Acknowledgement. This exercise is adapted from materials prepared by Chris Bourke from Department of Computer Science & Engineering, University of Nebraska-Lincoln.