Prepare your answers to the following questions in a single plain ASCII text file. If you work with a partner, make sure the submitted file contains a properly-completed copy of the partners form posted on the assignments page. Failure to do that will result in at least one of you not receiving credit for the assignment.

Submit your file to the Curator system, under the heading `C03`, by the posted deadline for this assignment. No late submissions will be accepted.

Download the associated tar file from the website, and unpack it into a subdirectory on your CentOS 7 installation or rlogin.

1. In the `q1` subdirectory, you will find the 64-bit Linux executable file `q1`. Run `q1` with the parameter 1234:

```
Linux>   ./q1 1234
```

a) **[6 points]** What happened? Be precise.

Start gdb on the `q1` program. The function where the computer is guessing a new value is called `takeGuess()`. Set a breakpoint on that function, and then begin running the program with the same 1234 argument. For each of the following questions, copy and paste the relevant part of your gdb session into your text file and explain your conclusions.

b) **[8 points]** When you have entered the `takeGuess()` function (line 100 should be displayed), what are the values of `hi` and `lo`?

c) **[6 points]** Is this the cause of the infinite loop error, or just a symptom related to an underlying problem? Why?

d) **[6 points]** Step through two more instructions (line 102 should be displayed, and the guess should have printed). What is printed? What is the value of `guess`?

e) **[8 points]** Is this the source of the infinite loop error, or just a symptom related to an underlying problem? Why?

f) **[8 points]** Continue through the while loop several times. You will soon notice some unusual behavior in the formal parameters of `takeGuess()`. What is it?

g) **[8 points]** Is this the source of the infinite loop error, or just a symptom related to an underlying problem? Why?

h) **[8 points]** Set a breakpoint at line 39 (the top of the while loop body) and delete your first breakpoint. Let's switch back to stepping through code lines one-by-one. After the computer takes a guess, the `handleGuess()` function determines whether the guess it correct, too high, or too low. A response value is set into the `guessReturnCode` variable. What is the value of `guessReturnCode` after line 43 has executed?

i) **[6 points]** Examine the comments above the `handleGuess()` function, which begins on line 113. Does this value of `guessReturnCode` make sense?

j) **[8 points]** With all of the information that you have seen you can now determine the source of the infinite loop error in my code. What is it? How can you fix it?

k) **[10 points]** A common mistake that novice programmers make is not thoroughly testing their code. There are some valid (e.g., with integer input between 1 and 10000 inclusive) test cases where the code will still run with no problem, and other cases where it will fail as you saw with input 1234. Experiment with the executable, finding two more cases where the input causes an infinite loop, as well as two cases where the input runs successfully and the computer makes a correct guess. What condition is needed to find a test case that does not induce the infinite loop bug?

2. [16 points] The `q2` directory contains a `q2` executable with the same overall goal as `q1`, but now with new bugs introduced. Use gdb to analyze the behavior of the `q2` executable, and explain what goes wrong.

You will have to find suitable test cases, and copy/paste parts of your gdb analysis to support your conclusions. Your score will depend on the correctness of your diagnosis, the completeness of your diagnosis, and how convincing the supporting gdb evidence you include in your submission is to the person who grades it.

Be aware that there may be more than one issue that affects the program's correctness. For full credit, you should add comments about your gdb analysis to explain why you've selected the particular gdb commands you are using, and what the output tells you.