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. You are advised to consult the course notes, the Matloff book online, and a good `gdb` cheat sheet. Note also that it is easy to copy text from a Linux terminal window and paste it into a text editor.

For each of the following questions, copy and paste the relevant part of your `gdb` session into your text file and explain your conclusions. Since the point of this assignment is to learn to use some feature of `gdb`, answers without `gdb` verification will receive no credit.

1. If you are working with a partner, each of you will have to answer this question separately. Just include your answers, clearly labelled in the file you submit.

In the `q1` subdirectory, you will find a C source file, `q1.c`. Compile the file with the command:

```
gcc -o q1 -std=c11 -Wall -ggdb3 q1.c
```

Start `gdb` on `q1` and set a breakpoint at line 15, then run `q1` (from within `gdb`) with your PID as a parameter. Since each student has a unique PID, some answers to the following questions will vary.

a) **[12 points]** When execution reaches the breakpoint, what are the values of the variables `limit` and `addend`? Display the values both in base-10 and hexadecimal.

There is a slim chance that the value of `addend` will be 0; in that case, use a `gdb` command to reset `addend` to be 172 before completing the following parts.

Now set a conditional breakpoint, at line 16, that will halt execution when the value of the variable `check` becomes larger than 30 million.

b) **[16 points]** Use a `gdb` command to display information about the breakpoints that are currently set.

c) **[16 points]** Step into the loop and display the value of `check` after the instructions in the loop body has been executed one time.

Use a `gdb` command to remove the first breakpoint (the one set for line 15).

d) **[8 points]** Use a `gdb` command to display information about the breakpoints that are currently set.

Now use a `gdb` command to run the program until the conditional breakpoint is triggered.

e) **[12 points]** What are the values of the variables `check` and `iter` now?

2. The directory `q2` contains the following files:

`q2.c`	C source file containing implementation of function `main()`
`Q2Test.h`	C header file containing declaration for function `Q2Test()`
`Q2Test.c`	C source file containing implementation of function `Q2Test()`
`Q2Fns.h`	C header file containing declarations for test functions called by `Q2Test()`
`q2`	64-bit CentOS executable built from files above, and `Q2Fns.c`

The questions below explore how to use `gdb` to trace the execution of code in a simple program involving multiple functions. You are not given the source code for some of the functions that will be called, so you cannot examine the tests they perform in order to understand why the program behaves as it does. You can, however, use `gdb` to follow the execution of the program and determine this.

Remember… this assignment is about using gdb. You must show relevant gdb output to justify your answers to the following questions. No credit will be given for answers that do not show that.

a) **[12 points]** Start `gdb` on `q2`. Use breakpoints, and other `gdb` commands, to show which of the functions called from `Q2Test()` returns true when you run `q2` with the parameters 500 and 100. Explain why your `gdb` output implies your conclusion.

b) **[12 points]** Start `gdb` on `q2`. Use breakpoints, and other `gdb` commands, to show which of the functions called from `Q2Test()` returns true when you run `q2` with the parameters 50 and 10. Explain why your `gdb` output implies your conclusion.

c) **[12 points]** Start `gdb` on `q2`. Use breakpoints, and other `gdb` commands, to show which of the functions called from `Q2Test()` experiences a runtime error when you run `q2` with the parameters 500 and 5. Explain why your `gdb` output implies your conclusion.