GDB Cheat Sheet

setup (do this before running)

`gdb ./filename`: opens file in gdb

`layout src`: displays C code

`break function_name`: sets breakpoint at start of function

break main (sets break point at the start of main function)

running

`run`: runs program from the beginning

(can run as many times as you like and break points are preserved)

`run other_file_name`: runs with file argument (soln.txt)

run floats.txt (run the program with floats.txt passed as an argument)

`continue`: runs program from current breakpoint

`next`: executes current line and pauses before next line of code

`step`: steps into a function

`quit`: kills program

`exit`: leave gdb

other helpful commands:

`p variablename`: prints a variable

`where`: shows you which stack frames you are in and called your current function

`frame #`: lets you look at a frame other than the current frame you are in

`info break`: see what break points you have set

`del 2`: delete listed #2 break point

`help gdbcommand`: provides information about that command

(**note:** hitting enter will repeat the last gdb command,
use the up/down arrows to find previous commands)

***use keys “ctrl-l” to reset the graphics after print output when screen output gets funky

GDB for assembly code debugging (weeks 6 & 7)

Set up:

`gdb ./filename`: opens file in gdb
`layout asm`: displays assembly code
`layout register`: displays registers
`break function_name`: sets breakpoint at start of function
`break *functionname+n`: sets breakpoint n bytes into a function

running

`run`: runs program from the beginning
`ni`: execute the next instruction in the program and pause
`si`: steps into a function at current instruction

examining state

`display $register`: shows contents of register when breakpoint is hit
`p $register`: prints value of register (/d=decimal,/c=char,/t=binary (0b),/x=hex(0x))
`x/s $register+0xn`: examines value at memory address as string
`x/wd $register+0xn`: examines value at memory address as decimal
`x/wx $register+0xn`: examines value at memory address as hexadecimal
`info registers`: displays information about registers