Quick Intro to the Gdb Debugger

GETTING HELP AND INFORMATION
---

*) Command: 'help'
   -- for list of help topics
   -- can specify a specific subtopic (e.g., data, breakpoints, etc.)

*) Command: 'apropos <topic>'
   -- lists matching commands about a topic

*) Exercises
   -- display all help topics
   -- display ALL help topics regarding data
   -- display ALL help topics regarding running the program
   -- display ALL help topics regarding the stack
   -- see commands related to the topic 'list'
   -- see commands related to the topic 'register'
   -- see commands related to the topic 'memory'

RUNNING THE PROGRAM
---

*) From the shell command line: gdb a.out
   -- loads executable, stops at first statement

*) Command: 'start'
   -- starts the program and stops at the first statement

*) Command: 'run' (or 'r')
   -- runs the program from the beginning

*) Command: 'continue' (or 'c')
   -- continues running the program from the current statement

*) Exercises
   -- Load a program into gdb (e.g., gdb a.out)
   -- Try the 'start' and 'run' commands repeatedly, several times!

```
----------------------------------------
VIEWING SOURCE CODE
----------------------------------------

*) Mode #1: Command-line Mode
   -- Command: 'list' (or 'l')
      -- scroll through the source code file
   -- Command: 'help list'
      -- display information about the list command

   *) Exercises for Command-line Mode
      -- list the next ten lines in the source code
      -- list the ten previous lines in the source code
      -- list code with a line number argument
      -- list code with a function argument
      -- set the listsize to 30 lines (type 'help list' to see how!)

*) Mode #2: Text User Interface (TUI mode)
   -- Command: 'tui enable'
      -- activates the text user interface
   -- Command: 'tui disable'
      -- deactivates the text user interface
   -- Command: Ctrl+x,a
      -- toggles between active/inactive TUI mode

   *) Exercises for an active TUI mode
      -- activate TUI mode
      -- scroll through code with up/down arrow, PageUp/PageDown


----------------------------------------
WALKING THROUGH CODE
----------------------------------------

*) Command: 'next' (or 'n')
   -- executes the next line of code
   -- does not trace into a function call

*) Command: 'step' (or 's')
   -- executes the next line of code
   -- will trace into a function call

*) Command: 'finish'
   -- completes current function and returns to caller

*) Exercises
   -- experiment with 'next' and 'step'
   -- descend into chain of function calls, use 'finish' to return to each caller
```

```
----------------------------------------
SETTING BREAKPOINTS
----------------------------------------

*) Command: 'break' (or 'b')
   -- can specify a target line number
   -- can specify a target function identifier
   -- use 'tbreak' to specify a temporary breakpoint

*) Command: 'info break'
   -- displays the breakpoint table (notice breakpoint numbers)

*) Command: 'enable <breakpoint number>'
   -- activates a breakpoint

*) Command: 'disable <breakpoint number>'
   -- deactivates a breakpoint

*) Exercises
   -- Type 'help break' to see general breakpoint help
   -- Type 'help breakpoint' to see detailed breakpoint commands
   -- Set four or five breakpoints using line numbers and function identifiers
   -- View the breakpoint table: type 'info break'
   -- Disable a specific breakpoint, verify the result
   -- Enable a specific breakpoint, verify the result
   -- Set a temporary breakpoint ('help tbreak'), verify the results

----------------------------------------
VIEWING THE STACK
----------------------------------------

*) Command: 'backtrace' (or 'bt')
   -- displays the call stack (note the frame numbers)

*) Command: 'frame <frame number>' (or 'f')
   -- activates the target frame

*) Exercises
   -- Trace program flow into nested function calls
   -- View the call stack up to that point
   -- Jump to some other frame
   -- View detailed info for the current frame ('info frame')


----------------------------------------
WORKING WITH VARIABLES
----------------------------------------

*) Command: 'print' (or 'p')
   -- displays the value of a variable
   -- can use to change a variable's value (e.g., 'print myInt = 55')
   -- to display artifical arrays: print ptr@<#elems> (e.g., 'print ptr@10')

*) Command: 'x' ("examine memory")
   -- syntax: type 'help x'
   -- to display a string: 'x/s charBuf'
```