~~~Few GDB debugging commands~~~

***break*:** Sets a breakpoint at a specified line number or function. You can use this command to pause execution of the program at a certain point and examine its state. Example: **break main** sets a breakpoint at the beginning of the **main** function.

run: Starts execution of the program from the beginning. You can use this command to run your program and stop it at a breakpoint or when it crashes.

next: Executes the next line of code in the program. If the line is a function call, it will execute the function without stopping at the individual lines of code inside the function.

step: Similar to **next**, but will stop at the individual lines of code inside a function.

continue: Continues execution of the program until it reaches the next breakpoint or it finishes executing.

print: Prints the value of a variable or expression. Example: **print x** will print the value of the variable **x**.

backtrace: Displays the current call stack, showing the sequence of functions that were called to reach the current point in the program.

watch: Sets a watchpoint on a variable. A watchpoint will stop execution of the program when the value of the variable changes. Example: **watch x** sets a watchpoint on the variable **x**.

info: Displays various pieces of information about the program, such as the values of registers or the loaded libraries. Few of the sub commands are:

info registers: This command will display the current values of all registers.

info source: This command will display information about the current source file.

info frame: This command displays information about the current stack frame, including the function name, arguments, and local variables. For example, use info frame to see information about the current stack frame.

info locals: This command displays information about the local variables in the current stack frame. For example, use info locals to see the values of all the local variables in the current function.

info breakpoints: This command lists all the breakpoints that have been set. For example, use info breakpoints to see a list of all the breakpoints that are currently active.

quit: Exits gdb and terminates the debugging session

unwatch: This command removes a watchpoint that was previously set. For example, to remove the watchpoint on **x**, use **unwatch x**.

tbreak: This command is similar to break, but it only stops the program once. After the program stops, the breakpoint is automatically removed. For example, to stop the program once it reaches line **10**, use **tbreak 10**.

return: This command is used to finish the current function and return to the calling function. You can use it when stepping through code to quickly move out of a function. For example, if you are inside the function **foo** and want to **return** to the calling function, use the return command.

finish: This command is similar to return, but it finishes the current function and returns all the way to the main function. For example, if you are several levels deep in function calls and want to quickly return to the main function, use the **finish** command.

disable: This command disables a breakpoint. For example, to disable breakpoint number **1**, use **disable 1**.

enable: This command re-enables a disabled breakpoint. For example, to enable breakpoint number **1**, use **enable 1**.

set: This command is used to set the value of a variable or register. For example, **set $eax = 10** will set the value of the **eax** register to **10**.

display: This command is used to print the value of a variable or register every time gdb stops at a breakpoint. For example, **display $eax** will print the value of the **eax** register every time gdb stops at a breakpoint.

condition: This command is used to set a condition for a breakpoint. For example, **break main if argc > 2** will set a breakpoint at the beginning of the main function only if **argc** is greater than **2**.

x: This command is used to examine memory. For example, **x/10x $esp** will display the **10 memory locations** starting at the address pointed to by the **esp** register.

stepi and ***nexti***: These commands are used to step through the program one instruction at a time. stepi will **step into** function calls, while nexti will **step over** them.

finish: This command is used to continue execution of the program until the current function is finished.

tui: This command is used to enable the Text User Interface mode in gdb. This mode provides a split screen display with the source code on one side and the gdb console on the other.

attach: This command is used to attach gdb to a running process. For example, attach 1234 will attach gdb to the process with the process ID of 1234

continue: This command allows you to continue running the program after it has been stopped. This can be useful during loops when you want to continue running the program until a certain condition is met.

until: This command continues execution of the program until a specific line of code is reached, which can be helpful for stepping out of loops or functions.
where: Prints a stack trace of the current execution.

unarg: This command is used to run a program with command line arguments.
(gdb) runarg argument1 argument2 argument3
Starting program: /path/to/program argument1 argument2 argument3

record: This command is used to record the execution of a program for later replay.
(gdb) record full
Record all instructions in full.

replay: This command is used to replay the execution of a program that was previously recorded.

For certain versions of gdb:
reverse-continue: This command is used to continue execution of the program in reverse direction.
reverse-stop: This command is used to stop execution of the program in reverse direction.
reverse-next: This command is used to step backwards by one line in the program
reverse-finish: This command is used to run the program in reverse direction until the current function returns.