GDB

GDB

- GDB (the **G**NU **D**e**b**ugger) is an amazingly powerful tool that we can use to root out glitches in our programs.

- As a general rule, the output and interaction with GDB can be a bit idiosyncratic and cryptic.
  - Fortunately, we've taken steps to solve this problem!

- If you aren't using the graphical debugger, it's useful to know how to work through your program with GDB's command line interface.

GDB

- To kick things off with GDB, type

```
gdb <program name>
```

- That will pull up the GDB environment. From there, the next two major commands you’ll likely use (in order) are:

```
b [function name, line number]
```

  - Makes it so that once your program begins, it will run uninterrupted until it encounters the function with that name or hits that line number, at which point the program will pause execution and await further input.

```
r [command-line arguments]
```

  - Runs the program with the command line arguments provided, if any.

GDB

`n`	Will step forward one block of code.
`s`	Will step forward one line of code.
`p [variable]`	Prints out the value of the variable given.
`info locals`	Prints out the values of all local variables.
`bt`	Shows you what series of function calls have led you to the current point in the program.
`q`	Quits GDB.

GDB

- Let’s try to use the text-based version of GDB inside CS50 IDE to debug a program called `buggy1` (compiled from `buggy1.c`) which is, as its name suggests, buggy!

GDB

```
#include <cs50.h>
#include <stdio.h>
#include <string.h>

int main(int argc, string argv[])
{
   if (strcmp(“”, argv[1]))
   {
      printf(“You figured it out!\n”);
   }
   else
   {
      printf(“Sorry :-(\n”);
   }
}
```