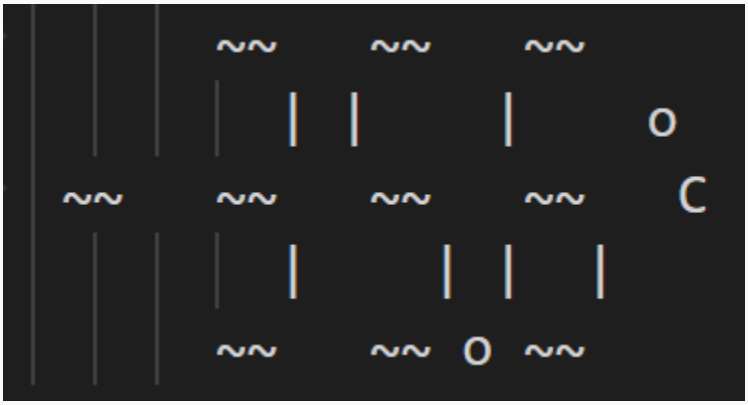
This project focused on Bit-level operations. A common skill in C and systems programming. This assignment features a problem in which shifting and bitwise AND/OR-ing are required to complete the requirements.

Debugging is also a critical skill enabled by the debugger. The second problem in the assignment makes use of the GNU Debugger, gdb, to work through a puzzle program requiring specific inputs to pass its "phases".

Part 1

Part 1 was to create a thermometer, the idea behind this part of the project was to explain the connection between software and hardware, with the idea that we had a binary reading from a temperature sensor and from there create a thermometer that converted and presented the temperature (either in Celsius or Fahrenheit). An example of this the display would be like:



For this part of the project we had to create the functions:

set_temp_from_ports()

Uses the two global variables (ports) THERMO_SENSOR_PORT and THERMO_STATUS_PORT to set the fields of `temp`. If THERMO_SENSOR_PORT is negative or above its maximum trusted value (associated with +45.0 deg C), this function sets the tenths_degrees to 0 and the temp_mode to 3 for `temp` before returning 1.

set_display_from_temp()

Alters the bits of integer pointed to by display to reflect the temperature in struct arg temp. If temp has a temperature value that is below minimum or above maximum temperature allowable or if the temp_mode is not Celsius or Fahrenheit, sets the display to read "ERR" and returns 1

thermo_update()

Called to update the thermometer display. Make use of set_temp_from_ports() and set_display_from_temp() to access the temperature sensor then set the display. Checks these functions and if they indicate an error, makes no changes to the display.

Part 2

Part 2 was all about an introduction to GDB debugging, the goal was to unlock a puzzle box by using the debugger to access certain lines of a compiled function and find out certain phrases or passcodes needed to unlock it. The puzzle box was unique for each student, the program did this by using the students ID to create a hash number, from there that hash number was used to modify the input to unlock the box. The general steps to complete this was to first see how many numbers needed to be imputed, from there I took a random guess. From there we would pass the first step of that phase and I could then look at how those numbers were modified. After that I would notice a math equation that the function used and they found the expected output, from there I would reverse engineer the equation to find the correct input. This was the general aspect of the project but it just got more and more complicated in each phase.

Gradescope

PROBLEM 1 (30.0/35.0)

```
gcc -Wall -Wno-comment -Werror -g  -c thermo_main.c
gcc -Wall -Wno-comment -Werror -g  -c thermo_update.c
gcc -Wall -Wno-comment -Werror -g  -c thermo_sim.c
gcc -Wall -Wno-comment -Werror -g  -o thermo_main thermo_main.o thermo_update.o thermo_sim.o
gcc -Wall -Wno-comment -Werror -g  -o test_thermo_update test_thermo_update.c thermo_sim.o thermo_update.o
./testy test_thermo_update.org
============================================================
== test_thermo_update.org : test_thermo_update and thermo_main tests
== Running 35 / 35 tests
1)  set_temp_from_ports() 0 C               : ok
2)  set_temp_from_ports() 0 F               : ok
3)  set_temp_from_ports() 128 C/F           : ok
4)  set_temp_from_ports() freezing C        : ok
5)  set_temp_from_ports() freezing F        : ok
6)  set_temp_from_ports() rounding C        : ok
7)  set_temp_from_ports() status nonzero    : ok
8)  set_temp_from_ports() sensor range      : ok
9)  set_temp_from_ports() status error      : ok
10) set_temp_from_ports() wide range        : ok
11) set_display_from_temp() 123 C           : ok
12) set_display_from_temp() 456 F           : ok
13) set_display_from_temp() 896 F           : ok
14) set_display_from_temp() 78 C            : ok
15) set_display_from_temp() -90 F           : FAIL -> results in file 'test-results/prob1-15-result.tmp'
16) set_display_from_temp() -234 C          : ok
17) set_display_from_temp() above 100       : ok
18) set_display_from_temp() extreme values  : ok
19) set_display_from_temp() error range     : ok
20) set_display_from_temp() error temp_mode : ok
21) set_temp() + set_display() negative     : ok
22) set_temp() + set_display() error        : ok
23) set_temp() + set_display() 1            : ok
24) thermo_update() positive temps          : ok
25) thermo_update() negative temps          : FAIL -> results in file 'test-results/prob1-25-result.tmp'
26) thermo_update() above 100 F             : ok
27) thermo_update() min/max                 : ok
28) thermo_update() status nonzeros         : ok
29) thermo_update() error range             : FAIL -> results in file 'test-results/prob1-29-result.tmp'
30) thermo_update() error status            : FAIL -> results in file 'test-results/prob1-30-result.tmp'
31) thermo_main 28544 F                     : ok
32) thermo_main 25333 C                     : ok
33) thermo_main 15333 C                     : ok
34) thermo_main 3430 F                      : ok
35) thermo_main -600 F                      : FAIL -> results in file 'test-results/prob1-35-result.tmp'
============================================================
RESULTS: 30 / 35 tests passed
Done
```

PROBLEM 2: Puzzlebox (55.0/50.0)

input.txt:

```
norma484 1 35 34
1 25 16
3 3 14 24
12345671234567 0 7 14
1000000 50
17 1629518194 16 1634625890
4.6893801601e+27
1000000 49
21 13 29 22 24
27797331633926381
```

```
gcc -Wall -Wno-comment -Werror -g  -c puzzlebox.c
gcc -Wall -Wno-comment -Werror -g  -o puzzlebox puzzlebox.o
./puzzlebox input.txt
=======================================
PROBLEM 2: Puzzlebox
UserID 'norma484' accepted: hash value = 1498045199
PHASE 1: A puzzle you say? Challenge accepted!
PHASE 2: That was cake by the ocean! Wait: the cake is a lie!
PHASE 3: Warm-up is over. This $#!^ just go real.
PHASE 4: Tired yet? Nope? There's more in phase four.
PHASE 5: You're doing well. But can you break through this secret technique of darkness?
PHASE 6: Watch out, here comes a wall of bricks! It's time for you to solve phase six.
PHASE 7: Next it's phase eleven! oops, seven. (off-by-4 errors don't lose credit, right?)
PHASE 8: You're doing great, now try phase 1000!
PHASE 9: Finally, the finish line; Can you solve phase nine?
PHASE 10: Rule #1: The doctor lies. Next time a message mentions 'finish line,' check the source code.
```

```
RESULTS: 55 / 50 points
```