Dylan Delgado HW #1

Problem 1: Create a subdirectory called “Homework1” in your personal github fork, and upload it to github. Execute the command “git branch -v” and store the output in a text file in that directory called “Problem1.txt”.

This problem is complete, the resultant file may be found at the following address: https://github.com/Dylan1496/PHY410/blob/HW1/Homework1/Problem1.txt.

Problem 2: What are the two’s complement representations for the following (decimal) numbers?

a) 10
b) 436
c) 1024
d) -13
e) -1023
f) -1024

We need to make sure we have enough bits to store these numbers. The largest number we have is 1024. So we need enough bits to represent that. 1024 is 2^{10}. So we need as many bits as needed to store that. The first bit is the one’s place, the second is the 2’s place, the third is the 4’s place, etc., until we get to the 1024’s place. This is the eleventh bit, so we need 11 bits.
We can now represent these numbers:

a) $10 = 8 + 2 = 2^3 + 0*2^2 + 2^1 + 0*2^0$

→ 10 is 00000001010 in binary.

b) 436

Clearly, $436 > 256 = 2^8$

$436 - 256 = 180$

$180 > 128 = 2^7$

$180 - 128 = 52$

$52 < 64 = 2^6$ but $52 > 32 = 2^5$

$52 – 32 = 20$

20 is $2^4 + 2^2 = 16 + 4$

so 436 is 00110110100 in binary.

c) 1024 is 2^{10}, so 1024 is 10000000000 in binary.

d) -13 is negative, so we’ll need to use 2’s complement here.

13 in binary: 13 is $8 + 4 + 1 = 2^3 + 2^2 + 0*2^1 + 2^0$. In 11 bits, 13 is 00000001101.
Flip the bits: 11111110010
Add 1: 11111110011 = -13 in binary

e) -1023 is negative.
1023 is 2^{10} -1, so 1023 = 01111111111
Flip the bits: 10000000000
Add 1: 10000000001 = -1023 in binary

f) -1024 is negative.

1024 is 2^{10}, so 1024 = 10000000000
Flip the bits: 01111111111
Add 1: 10000000000 = -1024 in binary

3. Suppose I need to compute the series $f_n = f_{n-1}^2$. If the value $f_0 = 2$, what is the maximum n that can be stored in the following C++ data types, assuming that an int is 2 bytes, a long int is 4 bytes, and each byte stores 4 bits?

a) int
b) long int
c) unsigned long int

As a note, this means int's are 8 bits, and long int's are 16 bits.

a) For int's, we have 8 bits. Also, an int is signed. This means we can represent the following range of numbers: -128 to 127.

Starting from n = 0,

$f_0 = 2$
$f_1 = 2^2 = 4$
$f_2 = 4^2 = 16$
$f_3 = 16^2 = 256$ (too big)

So the largest n that can stored in this data type is n = 2.

b) How about a long int?
Like int, long int is signed. However a long int can use 16 bits. So the allowed range is -32768 to 32767.

Starting from n = 0,

$f_0 = 2$
$f_1 = 2^2 = 4$
$f_2 = 4^2 = 16$
$f_3 = 16^2 = 256$
$f_4 = 256^2 = 65536$ (too big)

Here, the maximum n we can fit is n = 3.

c) How about an unsigned long int?

An unsigned long int can go up to $2^{16} - 1 = 65535$. This is one less than $f_4 = 65536$, so the largest n that we can store is n = 3.