Tutorial 1 - C and Data Representation Week 3

1. Variable Range Limit & Data Representation

(a). 2 147 483 647 = $2^{31} - 1$ ✓

↳ Largest value an int (32-bit) can take using complement representation

$\underbrace{\overbrace{0111 \ldots 1111}^{\text{31 1's}}}$

(b). Output : Start, i is 0x7fff ffff ✓

↳ Basically the hexadecimal representation of i

$\underbrace{\overbrace{1000 \ldots 0000}^{\text{31 0's}}}$

(c). Output : What ?! i is 0x 8000 0000 ✓

↳ Overflow has occurred (i is represented as a negative integer)

(d). 0x80000000 representation :

S&M : -0

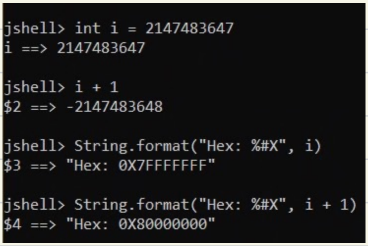
1s : - 2 147 483 647 $(2^{31}-1)$

2s : - 2 147 483 648 (2^{31})

Excess : unlikely

2s ? Not sure why though Print i and check value

(e). Java : - Python : BigNum

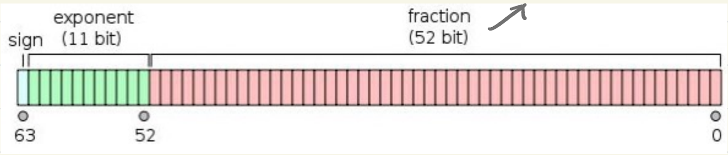


```
jshell> int i = 2147483647
i ==> 2147483647

jshell> i + 1
$2 ==> -2147483648

jshell> String.format("Hex: %#X", i)
$3 ==> "Hex: 0X7FFFFFFF"

jshell> String.format("Hex: %#X", i + 1)
$4 ==> "Hex: 0X80000000"
```

Clearly using 2s-complement

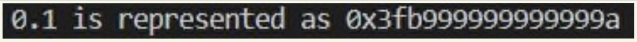
2. Floating Point Representation

(a).



mantissa

(b). Output :

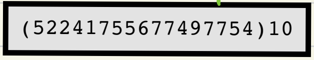


```
0.1 is represented as 0x3fb999999999999a
```

sign : 0x0 (+ve)

exponent : 0x 3fb = $(1019)_{10}$ ⇒ actual exp : 1019 − 1023 = −4

mantissa : 0x $\underbrace{999\ldots9}_{12}$ a =



(52241755677497754)10

$(2^{11-1} - 1)$

According to Google, representation of exp : excess − 1023

32-bit : excess $-(2^{8-1} - 1)$

(c). $\underbrace{1.1001\ 1001\ 10_2}_{\text{actually recurring}} \times 2^{-4} = 0.0001\ 1001\ 1001\ 10_2$

$= 0.999755859375_{10}$

(d). 0.1_{10} cannot be represented accurately in binary

⇒ inaccuracy

(e).
- We only take 10 bits
- Actual rep : 52 bits ⇒ greatly reduces inaccuracy
- Loop used to magnify error

(f). Most HLLs use IEEE 754 single / double represention

(g). Depends on whether the number can be converted to binary accurately within the limited mantissa bits