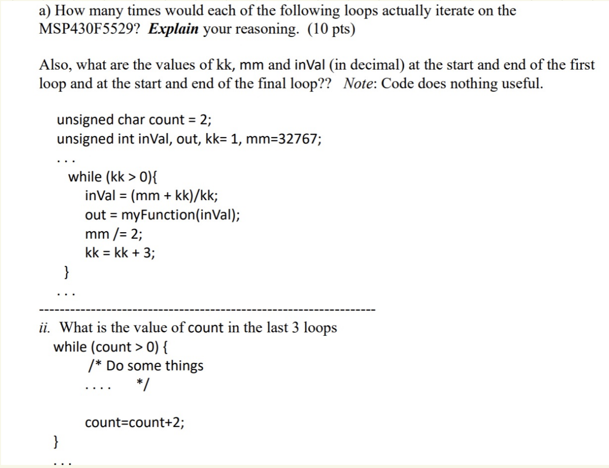
ECE 2049 A Term 2022

Sakshi Gauro
sgauro@wpi.edu

#1.



a) How many times would each of the following loops actually iterate on the MSP430F5529? ***Explain*** your reasoning. (10 pts)

Also, what are the values of kk, mm and inVal (in decimal) at the start and end of the first loop and at the start and end of the final loop?? *Note*: Code does nothing useful.

```
    unsigned char count = 2;
    unsigned int inVal, out, kk= 1, mm=32767;
    . . .
      while (kk > 0){
         inVal = (mm + kk)/kk;
         out = myFunction(inVal);
         mm /= 2;
         kk = kk + 3;
      }
    . . .
---------------------------------------------------------------------
```

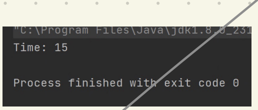
ii. What is the value of count in the last 3 loops

```
   while (count > 0) {
         /* Do some things
         . . . .      */

         count=count+2;
   }
   . . .
```

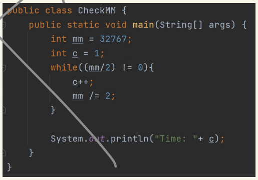
Rough:

infinite loop.

~~kk = ?~~	~~while (kk>0)~~	~~kk ± 3~~
~~1~~	~~1>0 (T)~~	



```
"C:\Program Files\Java\jdk1.8.0_231
Time: 15

Process finished with exit code 0
```



```
public class CheckMM {
    public static void main(String[] args) {
        int mm = 32767;
        int c = 1;
        while((mm/2) != 0){
            c++;
            mm /= 2;
        }

        System.out.println("Time: "+ c);
    }
}
```

a) The loop will iterate 21,845 times because the value of kk is constantly increase and fulfil the loop condition. Since, kk is an unsigned int, the max value is 65,536. There fore, it'll over flow until it reaches the max.

First Loop:

Start:

kk: 1

mm: 32767

inVal: 1

End:

kk: 4

mm: 16383

inVal: 32768

Final Loop:

Start:

kk: 65,533

mm: 0

inVal: $\frac{0 + \cancel{kk}}{\cancel{kk}} = 1$

End:

kk: 0

mm: 0

inVal: 0

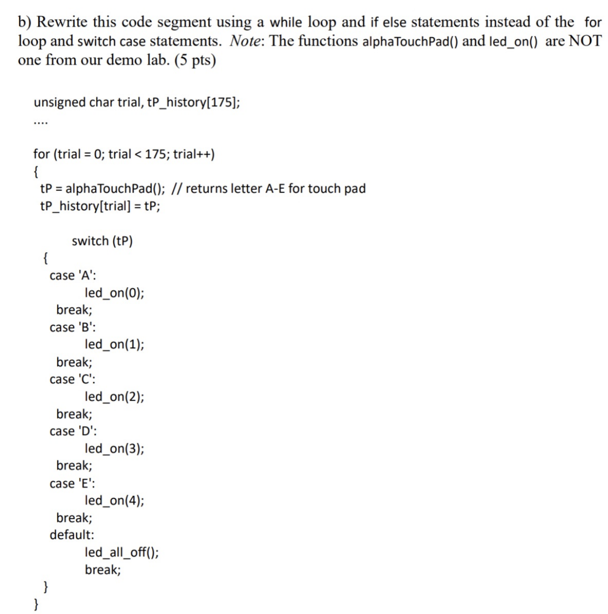
ii) unsigned char count = 2 (8-bit)

$\therefore 0$ to $2^8 = 256$ //

256 − 2 = 254

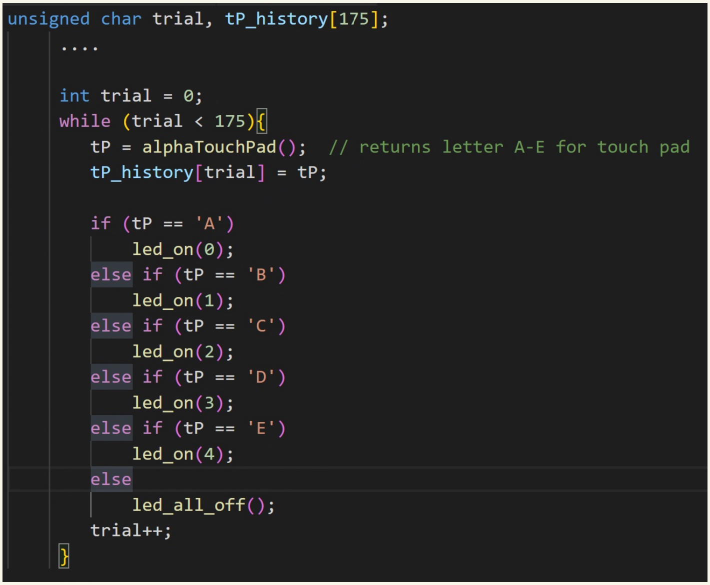
254 − 2 = 252

252 − 2 = 250 //

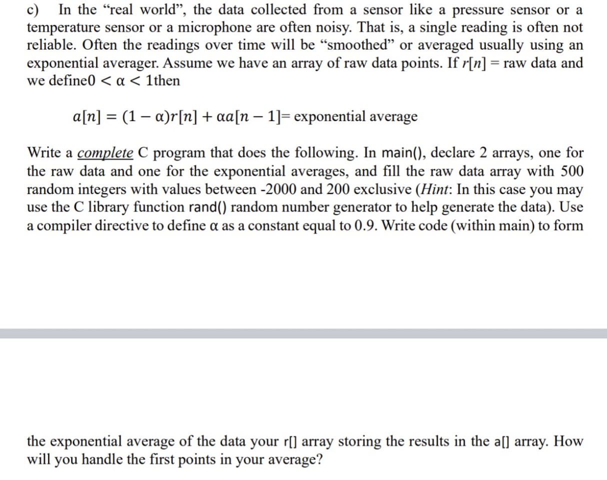


b) Rewrite this code segment using a while loop and if else statements instead of the for loop and switch case statements. *Note*: The functions alphaTouchPad() and led_on() are NOT one from our demo lab. (5 pts)

```
unsigned char trial, tP_history[175];
....

for (trial = 0; trial < 175; trial++)
{
 tP = alphaTouchPad();  // returns letter A-E for touch pad
 tP_history[trial] = tP;

      switch (tP)
  {
   case 'A':
         led_on(0);
    break;
   case 'B':
         led_on(1);
    break;
   case 'C':
         led_on(2);
    break;
   case 'D':
         led_on(3);
    break;
   case 'E':
         led_on(4);
    break;
   default:
         led_all_off();
         break;
  }
}
```



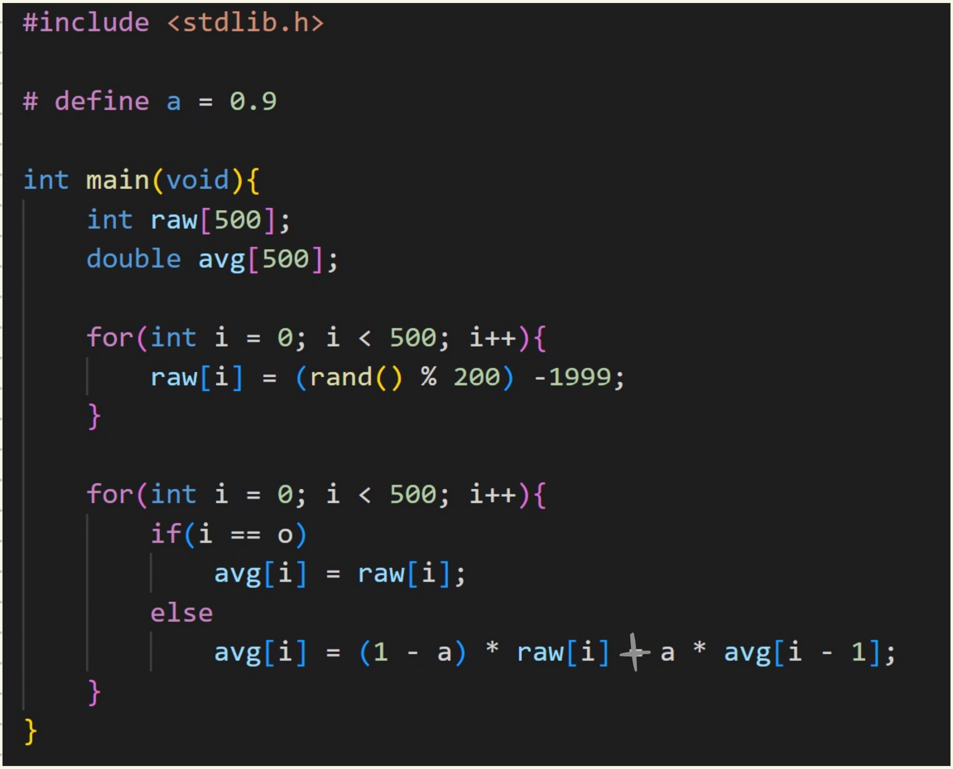
```
unsigned char trial, tP_history[175];
    ....

    int trial = 0;
    while (trial < 175){
       tP = alphaTouchPad();  // returns letter A-E for touch pad
       tP_history[trial] = tP;

       if (tP == 'A')
           led_on(0);
       else if (tP == 'B')
           led_on(1);
       else if (tP == 'C')
           led_on(2);
       else if (tP == 'D')
           led_on(3);
       else if (tP == 'E')
           led_on(4);
       else
           led_all_off();
       trial++;
    }
```



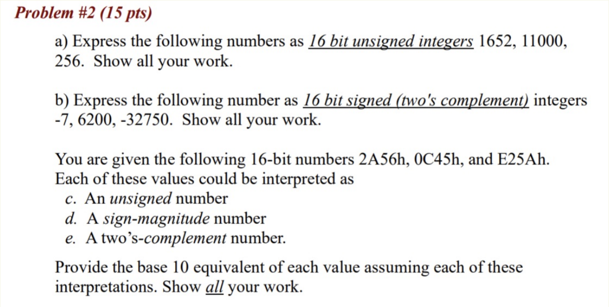
c) In the “real world”, the data collected from a sensor like a pressure sensor or a temperature sensor or a microphone are often noisy. That is, a single reading is often not reliable. Often the readings over time will be “smoothed” or averaged usually using an exponential averager. Assume we have an array of raw data points. If $r[n]$ = raw data and we define$0 < \alpha < 1$then

$a[n] = (1 - \alpha)r[n] + \alpha a[n-1]$= exponential average

Write a *complete* C program that does the following. In main(), declare 2 arrays, one for the raw data and one for the exponential averages, and fill the raw data array with 500 random integers with values between -2000 and 200 exclusive (*Hint*: In this case you may use the C library function rand() random number generator to help generate the data). Use a compiler directive to define α as a constant equal to 0.9. Write code (within main) to form the exponential average of the data your r[] array storing the results in the a[] array. How will you handle the first points in your average?



```
#include <stdlib.h>

# define a = 0.9

int main(void){
    int raw[500];
    double avg[500];

    for(int i = 0; i < 500; i++){
        raw[i] = (rand() % 200) -1999;
    }

    for(int i = 0; i < 500; i++){
        if(i == o)
            avg[i] = raw[i];
        else
            avg[i] = (1 - a) * raw[i] + a * avg[i - 1];
    }
}
```



Problem #2 (15 pts)

a) Express the following numbers as *16 bit unsigned integers* 1652, 11000, 256. Show all your work.

b) Express the following number as *16 bit signed (two's complement)* integers -7, 6200, -32750. Show all your work.

You are given the following 16-bit numbers 2A56h, 0C45h, and E25Ah. Each of these values could be interpreted as

c. An *unsigned* number
d. A *sign-magnitude* number
e. A two's-*complement* number.

Provide the base 10 equivalent of each value assuming each of these interpretations. Show *all* your work.

Soln

a) 1652 Deci → 16 Bit

1652/2 → R0
826/2 → R0
413/2 → R1
206/2 → R0
103/2 → R1
51/2 → R1
25/2 → R1
12/2 → R0
6/2 → R0
3/2 → R1
1/2 → R1
0

∴ 1652 = 0000 0110 0111 0100

b) 11000 Dec → 16 Bit

11000/2 → R0
5500/2 → R0
2750/2 → R0
1375/2 → R1
687/2 → R1
343/2 → R1
171/2 → R1
85/2 → R1
42/2 → R0
21/2 → R1
10/2 → R0
5/2 → R1
2/2 → R0
1/2 → R1
0

∴ 11000 =
0010 1010 1111 1000

c) 256 Dec → 16 Bit

256/2 → R0
128/2 → R0
64/2 → R0
32/2 → R0
16/2 → R0
8/2 → R0
4/2 → R0
2/2 → R0
1/2 → R1
0

∴ 256 = 0000 0001 0000 0000

b) i) -7

7/2 = R1
3/2 → R1
1/2 → R1
0

∵ 7 = 0000 0000 0000 0111
= 1111 1111 1111 1000
+1

∴ -7 = 1111 1111 1111 1001

ii) 6200

6200/2 → R0
3100/2 → R0
1550/2 → R0
775/2 → R1
387/2 → R1
193/2 → R1
96/2 → R0
48/2 → R0
24/2 → R0
12/2 → R0
6/2 → R0
3/2 → R1
1/2 → R1
0

0000 0011 1000 0011 ⇒ 6200

iii) −32750

32750/2 → R0
16375/2 → R1
8187/2 → R1
4093/2 → R1
2046/2 → R0
1023/2 → R1
511/2 → R1
255/2 → R1
127/2 → R1
63/2 → R1
31/2 → R1
15/2 → R1
7/2 → R1
3/2 → R1
1/2 → R1
0

0011 1011 1111 1111 → 32750

1100 0100 0000 0000 → −32750

c) An unsigned number

i) 2A56 h

$= 2 \times 16^3 + 10 \times 16^2 + 5 \times 16^1 + 6 \times 16^0$

$= 10838$

ii) 0C45 h

$= 0 \times 16^3 + 12 \times 16^2 + 4 \times 16^1 + 5 \times 16^0$

$= 3141$

iii) E25A h

$= 14 \times 16^3 + 2 \times 16^2 + 5 \times 16^1 + 10 \times 16^0$

$= 57946$

d) A sign-magnitude number

i) 2A56 h

+ve number

0010 1010 0101 0110

∴ 2A56h = 10838

ii) 0C45 h

0 ⇒ +ve number

∴ 0C45h = 3141

iii) E25Ah

= 1110 0010 0101 1010

−ve number → MSB = 1

110 0010 0101 1010

$= 2^{14} + 2^{13} + 2^{9} + 2^{6} + 2^{4} + 2^{3} + 2^{1}$

$= 25178$

∴ E25Ah = −25178

e) A two's complement number

i) 2A56 h

+ve number

0010 1010 0101 0110

∴ 2A56h = 10838

ii) 0C45 h

0 ⇒ +ve number

∴ 0C45h = 3141

iii) E25Ah

= 1110 0010 0101 1010

−ve number → MSB = 1

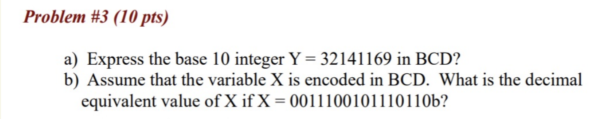
110 0010 0101 1010

$$\begin{array}{r} 0001\ 1101\ 1010\ 0101 \\ +1 \\ \hline 0001\ 1101\ 1010\ 0110 \end{array}$$

$= 2^{12} + 2^{11} + 2^{10} + 2^{8} + 2^{7} + 2^{5} + 2^{2} + 2^{1}$

$= 7590$

∴ E25Ah = −7590



Problem #3 (10 pts)

a) Express the base 10 integer Y = 32141169 in BCD?

b) Assume that the variable X is encoded in BCD. What is the decimal equivalent value of X if X = 0011100101110110b?

3)

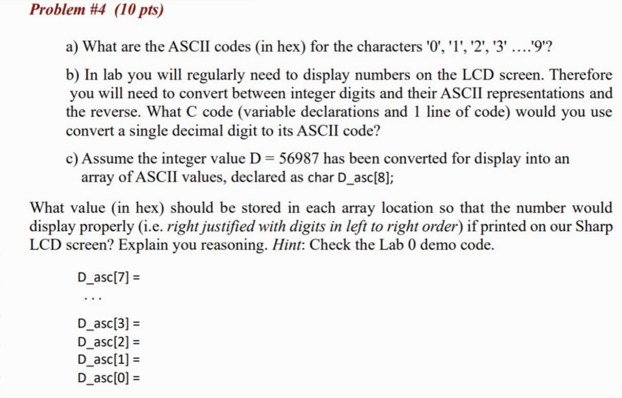
a) Y = 32141169 in BCD

③	②	①	④	①	①	⑥	⑨
0011	0010	0001	0100	0001	0001	0110	1001 //

b) X = 0011 1001 0111 0110 b → deci

0011	1001	0111	0110
↓	↓	↓	↓
3	9	7	6

∴ 3976 //



Problem #4 (10 pts)

a) What are the ASCII codes (in hex) for the characters '0', '1', '2', '3' ….'9'?

b) In lab you will regularly need to display numbers on the LCD screen. Therefore you will need to convert between integer digits and their ASCII representations and the reverse. What C code (variable declarations and 1 line of code) would you use convert a single decimal digit to its ASCII code?

c) Assume the integer value D = 56987 has been converted for display into an array of ASCII values, declared as char D_asc[8];

What value (in hex) should be stored in each array location so that the number would display properly (i.e. *right justified with digits in left to right order*) if printed on our Sharp LCD screen? Explain you reasoning. *Hint*: Check the Lab 0 demo code.

D_asc[7] =
. . .

D_asc[3] =
D_asc[2] =
D_asc[1] =
D_asc[0] =

a) '0' → 48 '1' → 49 '2' → 50 '3' → 51 '4' → 52 '5' → 53

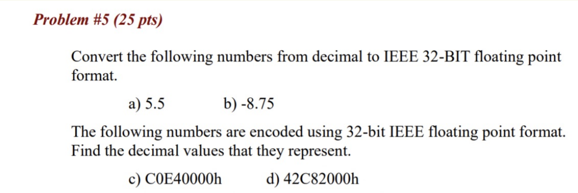
'6' → 54 '7' → 55 '8' → 56 '9' → 57

b) When a integer digit is stored in a variable declared as 'char', we can use it to convert a single decimal digit to its ASCII code.
char variable = 'digit'.

c) D = 56987

$$\underbrace{5 \to 35 \quad 6 \to 36 \quad 9 \to 39 \quad 8 \to 38 \quad 7 \to 37,}_{\text{(in hex)}}$$

D-asc[0] = 35
D-asc[1] = 36
D-asc[2] = 39
D-asc[3] = 38
D-asc[4] = 37
D-asc[5] = 00
D_asc[6] = 00
D-asc[7] = 00



Problem #5 (25 pts)

Convert the following numbers from decimal to IEEE 32-BIT floating point format.

a) 5.5 b) -8.75

The following numbers are encoded using 32-bit IEEE floating point format. Find the decimal values that they represent.

c) C0E40000h d) 42C82000h

5. IEEE 32-Bit Base = 127 [always]

a) 5.5

5/2 → R1
2/2 → R0
1/2 → R1
0

5.5 = 101.101
→ 1.01101 × 2②

0/1
(Sign bit) (127+2) (01101)
8 exp. bit 23 Fraction bit

129/2 → R1
64/2 → R0
32/2 → R0
16/2 → R0
8/2 → R0
4/2 → R0
2/2 → R0
1/2 → R1
0

10000001

$$\boxed{\underset{\text{Sign}}{0}\ \underbrace{1000\ 0001}_{\text{8-exp Bit}}\ \underbrace{0110\ 1000\ 0000\ 0000\ 0000\ 000}_{\text{23 fraction Bit}}}$$

b) -8.75

8/2 → R0
4/2 → R0
2/2 → R0
1/2 → R1

8.75 = 1000.1001011

1.0001001011 · 2^3

75/2 → R1
37/2 → R1
18/2 → R0
9/2 → R1
4/2 → R0
2/2 → R0
1/2 → R1
0

Sign bit (sign) 8 exp. bit (127+3 = 130) Fraction bit (0001001011)

∴ $\underbrace{1}\ \underbrace{1000\ 0110}\ \underbrace{0001\ 0010\ 1100\ 0000\ 0000\ 000}$

130/2 → R0
65/2 → R1
32/2 → R1
16/2 → R0
8/2 → R0
4/2 → R0
2/2 → R0
1/2 → R1
0

c) C0E40000 h

(C) (0) (E) (4) (0) (0) (0) (0)
1100 0000 1110 0100 0000 0000 0000 0000

$$\underset{\overset{\uparrow}{\text{sign}}}{1} \quad \underbrace{1000\ 0001}_{8\text{ exp bit}} \quad \underbrace{1100\ 1000\ 0000\ 0000\ 0000\ 000}_{23\text{ fraction bit}}$$

$2^0 + 2^7 = 128 + 1 = 129$

$1.11001 \cdot 2^2$

$\longrightarrow 111.001$

$127 + x = 129$

$\therefore x = 2$

$111 = 2^0 + 2^1 + 2^2$
$= 7$

$001 = 2^{-3}$
$= .125$

$$\boxed{\therefore -7.125}$$

d. 42C82000 h

(4) (2) (C) (8) (2) (0) (0) (0)
0100 0010 1100 1000 0010 0000 0000 0000

$$\underset{\overset{\uparrow}{\text{sign}}}{0} \quad \underbrace{1000\ 0101}_{8\text{ exp bit}} \quad \underbrace{1001\ 0000\ 0100\ 0000\ 0000\ 000}_{23\text{ fraction bit}}$$

$2^0 + 2^2 + 2^7 = 133$

$1.1001\,0\,0\,0001 \cdot 2^6$

$= 1100100.0001$

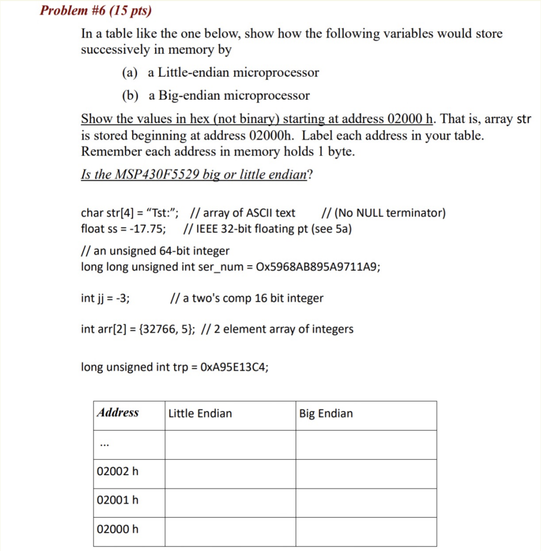
$127 + x = 133$

$\therefore x = 6$

$1100100 = 2^2 + 2^5 + 2^6$
$= 100$

$0.0001 = 2^{-4}$
$= 0.0625$

$$\boxed{100.0625}$$



Problem #6 (15 pts)

In a table like the one below, show how the following variables would store successively in memory by

(a) a Little-endian microprocessor

(b) a Big-endian microprocessor

Show the values in hex (not binary) starting at address 02000 h. That is, array str is stored beginning at address 02000h. Label each address in your table. Remember each address in memory holds 1 byte.

Is the MSP430F5529 big or little endian?

```
char str[4] = “Tst:”;   // array of ASCII text        // (No NULL terminator)
float ss = -17.75;      // IEEE 32-bit floating pt (see 5a)

// an unsigned 64-bit integer
long long unsigned int ser_num = Ox5968AB895A9711A9;

int jj = -3;             // a two's comp 16 bit integer

int arr[2] = {32766, 5};  // 2 element array of integers

long unsigned int trp = 0xA95E13C4;
```

Address	Little Endian	Big Endian
...		
02002 h		
02001 h		
02000 h		

the MSP430 is Little Endian.

Memory		Little Endian	Big Endian
0200	Tst:	54	54
0201		73	73
0202		74	74
0203		3A	3A
0204	-17.75 ★→	00	C1
0205		00	8E
0206		8E	00
0207		C1	00
0208	0x5968AB89 5A9711A9	Aa	59
0209		11	68
020A		97	AB
020B		5A	89
020C		89	5A
020D		AB	97
020E		68	11
020F		59	A9
0210	-3 #→	FD	FF
0211		FF	FD
0212	{32766,5}	FE	7E
0213		7F	FE
0214		05	00
0215		00	05
0216	0x1795E13C4	C4	A9
0217		13	53
0218		53	13
0219		A9	C4

−17.75 → 17.75
⇓
10001.11

Expo = 4
mant = 000111
↳ 127+4 = 131 ⟶ 10000011

1100 0001 1000 1110 0000 0000 0000 0000
C 1 8 E

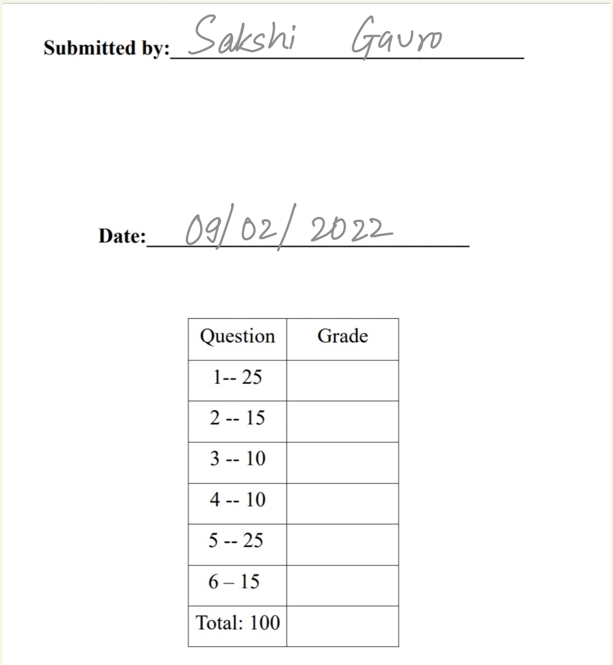
−3 ⟶ 0000 0000 0000 0011 → 3
↳

$$\begin{array}{r} 1111\ 1111\ 1111\ 1100 \\ +1 \\ \hline 1111\ 1111\ 1111\ 1101 \end{array}$$ → −3

F F F D

32766/2 → R0
16383/2 → R1
8191/2 → R1
4095/2 → R1
2047/2 → R1
1023/2 → R1
511/2 → R1
255/2 → R1
127/2 → R1
63/2 → R1
31/2 → R1
15/2 → R1
7/2 → R1
3/2 → R1
1/2 → R1
0 ↑

= 7FFE



Submitted by: Sakshi Gauro

Date: 09/02/2022

Question	Grade
1-- 25	
2 -- 15	
3 -- 10	
4 -- 10	
5 -- 25	
6 – 15	
Total: 100	