1. Encode the **decimal value 33** into each of the following representations (use minimum number of required bits for parts a and b)

   a. **(2 points)** unsigned representation: 100001

   b. **(2 points)** 2's complement representation: 0100001

   c. **(2 points)** Hexadecimal: $(21)_{16}$

2. Given the **bit pattern 0111 0100,** calculate the **decimal equivalent** of this bit pattern when it is interpreted in each of the following representations:
(Note: Truncate the most significant bit if the representation has 7-bit)

a. **(2 points) 7-bit** unsigned : 116 (decimal equivalent) 64+32+16+4

b. **(2 points) 7-bit** 2's complement : −12 (decimal equivalent)

c. (**2 points**) **8-bit** 2's complement : 116 (decimal equivalent)

3. In this problem, you can refer to the IEEE 754 single-precision format provided on the last page of this exam booklet. You will encode the decimal fraction **-11.71875** using IEEE 754 single-precision floating-point.

a. **(3 points)** Begin by converting **11.71875** into binary.

Answer: 1011·10111

b. **(2 points)** Rewrite your answer to **part (a)** using normalized binary scientific notation, for example, 1.001011×2^{13}.

Answer: **1.** 0111 10111 × **2**^{3}

UIN:

Exponent = 127+3 = 130

c. **(2 points)** Fill in the bits below to represent **-11.71875** in IEEE 754 single-precision floating-point. Some bits have been filled in for you already.

FIRST GROUP OF 16 BITS

1	1	0	0	0	0	0	1	0	0	1	1	1	0	1	1

~~0~~ 1	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0

SECOND GROUP OF 16 BITS

3. In this problem, you can refer to the IEEE 754 single-precision format provided on the last page of this exam booklet. You will encode the decimal fraction **-11.71875** using IEEE 754 single-precision floating-point.

a. **(3 points)** Begin by converting **11.71875** into binary.

Answer: 1011·10111

b. **(2 points)** Rewrite your answer to **part (a)** using normalized binary scientific notation, for example, 1.001011×2^{13}.

Answer: **1.** 01110111 × **2**^{3}

UIN:

Exponent = 127+3 = 130

c. **(2 points)** Fill in the bits below to represent **-11.71875** in IEEE 754 single-precision floating-point. Some bits have been filled in for you already.

FIRST GROUP OF 16 BITS

1	1	0	0	0	0	0	1	0	0	1	1	1	0	1	1

~~0~~ 1	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0

SECOND GROUP OF 16 BITS

4. **(4 points)** The numbers A and B below are represented in IEEE 754 single-precision floating-point.

FIRST GROUP OF 16 BITS for A

Exponent = 126

1	0	1	1	1	1	1	1	0	0	0	1	0	1	0	1

0	0	1	1	1	1	0	1	0	1	1	1	0	1	1	0

SECOND GROUP OF 16 BITS for A

FIRST GROUP OF 16 BITS for B

Exponent = 170

1	1	0	1	0	1	0	1	0	0	0	1	0	1	0	1

0	0	1	1	1	1	0	1	0	1	1	1	0	1	1	0

SECOND GROUP OF 16 BITS for B

An arithmetic operation was applied on A to obtain B. In other words,

$$\boxed{A \times 2^{44} = B}$$

$$A \boxed{\times 2^{44}} = B$$

Fill in the blank by writing an arithmetic operator and a number, e.g., $\boxed{+\mathbf{1}}$

1. Perform the following bit extension for the bit pattern 101011. The number represented should not change.

   a. **(1 point)** From 6-bit unsigned to 8-bit unsigned 00101011

   b. **(1 point)** From 6-bit 2's complement to 8-bit 2's complement: 11101011

   c. **(1 point)** How do you extend bit patterns in general for 2's complement representation? Explain using no more than 15 words.

   To extend an n-bit 2's complement to (n+k)-bit representation, we need to append the sign-bit k times to the left position.

2. a. **(0.5 points)** What is the bit pattern for the smallest number in 4-bit unsigned representation?

0000

b. **(0.5 points)** What is this smallest number from 2(a) in decimal value?

$(0)_{10}$

c. **(0.5 points)** What is the bit pattern for the largest number in 4-bit unsigned representation?

1111

d. **(0.5 points)** What is this largest number from 2(c) in decimal value?

$(15)_{10}$

e. **(1 point)** What is the numerical range of a *k*-bit unsigned representation? Express your answer as a function of *k* in the form of [*min*, *max*].

$[\ 0\ ,\ 2^{k}-1\ ]$

3. a. **(0.5 points)** What is the bit pattern for the smallest number in 4-bit 2's complement representation? (Note: -1 > -2, meaning -2 is smaller than -1.)

1000

UIN:

b. **(0.5 points)** What is this smallest number from 3(a) in decimal value?

$(-8)_{10}$

c. **(0.5 points)** What is the bit pattern for the largest number in 4-bit 2's complement representation?

0111

d. **(0.5 points)** What is this largest number from 3(c) in decimal value?

$(7)_{10}$

e. **(1 point)** What is the numerical range of a *k*-bit 2's complement representation? Express your answer as a function of *k* in the form of [*min*, *max*].

$[-2^{k-1}, 2^{k-1}-1]$

3. a. **(0.5 points)** What is the bit pattern for the smallest number in 4-bit 2's complement representation? (Note: -1 > -2, meaning -2 is smaller than -1.)

1000

UIN:

b. **(0.5 points)** What is this smallest number from 3(a) in decimal value?

$(-8)_{10}$

c. **(0.5 points)** What is the bit pattern for the largest number in 4-bit 2's complement representation?

0111

d. **(0.5 points)** What is this largest number from 3(c) in decimal value?

$(7)_{10}$

e. **(1 point)** What is the numerical range of a *k*-bit 2's complement representation? Express your answer as a function of *k* in the form of [*min*, *max*].

$[-2^{k-1}, 2^{k-1}-1]$

4. a. **(0.5 points)** For an odd non-negative integer, the rightmost bit in its unsigned representation is always:

"1"
____

b. **(0.5 points)** For an even negative integer, the leftmost bit in its 2's complement representation is always:

"1"
____

5. Perform the following 8-bit binary arithmetic operations. Indicate whether each operation results in an overflow when the bit patterns are interpreted using the indicated representation by circling the corresponding YES or NO. Show your work.

Let G **= xE6,** H **= xA9, P** = 01001011 **and Q** = 01010111

a. **(2 points) G + H =**

G = 1110 0110
H = 1010 1001
G+H = ~~1~~| 1000 1111

b. **(1 point)** Does overflow occur in (a) when the numbers have an unsigned representation?

[✓ Yes] No

UIN:

c. **(1 point)** Explain your reasoning for (b) in no more than 10 words.

The addition operation results in a carry out from the MSB position.

d. **(1 point)** Does overflow occur in (a) when the numbers have a 2's complement representation?

Yes [✓ No]

e. **(1 point)** Explain your reasoning for (d) in no more than 10 words.

The sign bits of the addend and the result are Same that is why no overflow.

f. **(2 points)** What is P +Q in 2's complement representation?

P = 0100 1011
Q = 0101 0111
P + Q = P+Q = 1010 0010

g. **(1 point)** Does overflow occur in (f)? [✓ Yes] No

h. **(1 point)** Explain your reasoning for (g) in no more than 10 words.

The addion of two positive numbers Results in a negative number. [sign bits are - different]

5. Perform the following 8-bit binary arithmetic operations. Indicate whether each operation results in an overflow when the bit patterns are interpreted using the indicated representation by circling the corresponding YES or NO. Show your work.

Let G = **xE6,** H = **xA9, P** = 01001011 **and Q** = 01010111

a. **(2 points) G + H =**

G= 1110 0110
H= 1010 1001
G+H= ~~1~~| 1000 1111

b. **(1 point)** Does overflow occur in (a) when the numbers have an unsigned representation?

[✓ Yes] No

UIN:

c. **(1 point)** Explain your reasoning for (b) in no more than 10 words.

The addition operation results in a carry out from the MSB position.

d. **(1 point)** Does overflow occur in (a) when the numbers have a 2's complement representation?

Yes [✓ No]

e. **(1 point)** Explain your reasoning for (d) in no more than 10 words.

The Sign bits of the addend and the result are Same that is why no overflow.

f. **(2 points)** What is P +Q in 2's complement representation?

P= 0100 1011
Q= 0101 0111

P + Q = P+Q= 10100010

g. **(1 point)** Does overflow occur in (f)? [✓ Yes] No

h. **(1 point)** Explain your reasoning for (g) in no more than 10 words.

The addion of two positive numbers Results in a negative number. [Sign bits are different]

6. Let **W** = 10111100 and **Z** = 1001

a. **(2 points)** Compute **W - Z** using 8-bit **2's complement representation for W, Z and the result**. (Show your work.)

W = 10111100

Z = 11111001

-Z = 00000111

W - Z = -Z = 00000111

-Z = 00000111

W-Z = 11000011

b. **(1 point)** Does overflow occur in (a)? **Yes** [✓ **No**]

c. **(1 point)** Explain your reasoning for (b) in no more than 10 words.

Adding a positive number to a negative number will always produce no overflo

7

5. (2 points) Binary, Characters, and ASCII

Using the attached ASCII table write the sequence of 7-bit ASCII codes that corresponds to the string **`Hey!`**

Hex sequence 48 65 79 21

Binary sequence 1001000 1100101 1111001 010 0001

8 ~~6.~~ **(3 points)** In a C program, suppose we declare: `int A = -2`. What do the following expressions evaluate to? Write the answer in **the requested** form and show your work. (Assume int is a 32-bit 2's complement integer.)

a. A >> 4 Answer in decimal: $(-1)_{10}$

b. 1 << 5 Answer in hex: $2^5 = (32)_{10}$ = x 00000020

c. A | (~(1<<3)) Answer in hex: x FFFFFFFF

Problem 3 (10 points): Boolean Functions and Truth Table

(**4 points**) Calculate the outputs of the following two Boolean functions of three variables:

$$S(A, B, C) = (A'B'C) + (A'BC')+(AB'C')+(ABC)$$

$$P(A, B, C) = (A+B+C)(A+B'+C')(A'+B+C')(A'+B'+C)$$

C	B	A	S	P
0	0	0	0	0
0	0	1	1	1
0	1	0	1	1
0	1	1	0	0
1	0	0	1	1
1	0	1	0	0
1	1	0	0	0
1	1	1	1	1

1. (**2 points**) What conclusion you can draw about the functions S and P from the completed truth table? Answer in less that 10 words.

S and P are equivalent.

SOP and POS expressions of the same function.

2. (**4 points**) Perform the following logical operations. Express your answers in hexadecimal notation.

a. NOT((NOT(xDEFA)) AND (NOT(xFFFF))) = xFFFF

b. x00FF XOR x325C = x 32A3

show your work:

a.

NOT(xFFFF) = 0000 0000 0000 0000

NOT(xDEFA) = 0010 0001 0000 0101

AND ⇒ 0000 0000 0000 0000

NOT (AND..) = xFFFF

b.

x00FF → 0000 0000 1111 1111

x325C → 0011 0010 0101 1100

0011 0010 1010 0011

= x 32A3

1. What happens when the user inputs **f** as the **choice** and **4** as the integer **n**?

Answer:

```
Enter choice (f for factorial, s for sum, p for prime sum):
Enter a positive integer n :
Factorial of 4 is 24
```

2. What will the program output if the user inputs **s** as the **choice** and **5** as the integer **n**?

Answer:

```
Enter choice (f for factorial, s for sum, p for prime sum):
Enter a positive integer n :
Sum of first 5 natural number is 15
```

3. What will the program output if the user inputs **s** as the **choice** and **n** as the integer **n**?

Answer:

```
Enter choice (f for factorial, s for sum, p for prime sum):
Enter a positive integer n :
Invalid Input
```

4. If the user enters **p** as the **choice** and **-2** as **n**, what will be the output?

Answer:

```
Enter choice (f for factorial, s for sum, p for prime sum):
Enter a positive integer n :
Welcome to ECE 120! n must be positive.
```

5. What output does the program produce when the user inputs **x** as the **choice** and **3** as **n**?

Answer:

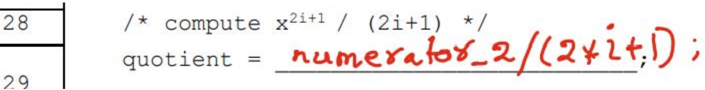
```
Enter choice (f for factorial, s for sum, p for prime sum):
Enter a positive integer n :
Welcome to ECE 120! Invalid choice.
```

```
#include <stdio.h>
int main()
{
   int N, M, i, j;
   float x, quotient;
   float numerator_1, numerator_2 = 1;
  float sum = 0.0, product = 1.0;
  printf("Enter x, M and N: \n");
   scanf("%f %d %d", &x, &M, &N);

  for (j = 1; j ≤ M; j=j+1)
   {
     /* Compute 4j² */
        numerator_1 = 4*j*j;

     /* Update product */
     product = product * numerator_1/(numerator_1-1);
   }

  for (i = 0; i < N+1; i=i+1)
  {
    if (i==0) { /* compute x^(2i+1) for i=0 */
       numerator_2 = numerator_2 * x;
    }
    else {        /* compute x^(2i+1) */
    numerator_2 = numerator_2*x*x;
    }
    /* compute x^(2i+1) / (2i+1) */
    quotient = numerator_2/(2*i+1);

      if (i % 2 == 1){    /* (-1)^i */
         quotient = (-1) * quotient;
       }
    sum = sum + quotient;
  }
  printf("For x=%f,the result is %f\n", x, sum);
  return 0;
}
```

```
printf("Enter x, M and N: \n");
scanf("%f %d %d", &x , &M , &N );
```

```
for (j = 1; j ≤ M; j=j+1)
{
```

20	`for (i = 0 ; i < N+1; i=i+1)`



```
/* compute x$^{2i+1}$ / (2i+1) */
quotient = numerator_2/(2*i+1) ;;
```

```
if (____i_____ % 2 == 1){    /* (-1)^i */
```

```
            }
        sum = sum + quotient;
    }
```

2

5.8

%f, %f, product-sum

Correct

2

5.8

%f, %f, product-sum

Correct