Homework 1

T1

1. Convert these decimal numbers to **8-bit** 2's complement numbers:
   - -114
   - +81
2. Convert the following **8-bit** 2's complement numbers to decimal.
   - 0011 0010
   - 1111 1101

T2

1. What's the smallest and largest number that can be represented by an **8-bit** 2's complement number? (Answer in decimal)
2. Try to determine the range that an **N-bit** 2's complement number can represent. (Answer in decimal)

T3

Is there a negative integer that has identical 2's complement representation and original code in binary (8-bit)? If so, what is it? (Answer in decimal)

T4

The C code below takes two integers and prints out whether the first number is less than the second one. (The numbers given are guaranteed to be in the range of `int`.)

```c
#include <stdio.h>

int main(void) {
    int a, b;
    scanf("%d %d", &a, &b);
    if (a - b < 0) {
        printf("a < b\n");
    } else {
        printf("a >= b\n");
    }
    return 0;
}
```

1. Under what circumstances will the program print `a < b` while actually $a \geqslant b$?
2. What if we change the code to the following? (Also, the numbers given are guaranteed to be in the range of `unsigned int`.)

```c
#include <stdio.h>

int main(void) {
    unsigned int a, b;
    scanf("%d %d", &a, &b);
    if (-a > -b) {
        printf("a < b\n");
    } else {
        printf("a >= b\n");
    }
    return 0;
}
```

T5

Write the decimal equivalents for the IEEE floating point number below.

0 10001011 00000000001000000001000

T6

What is the **smallest number** that can be represented in IEEE floating point format with 32 bits regardless of infinity? What about the **smallest positive number**? (Answer in binary)

> We define a number is smaller as it is on the lefter side of the number axis.

T7

Can you list all the integers whose IEEE floating point representations are exactly the same as their 2's complement integer representations? (Answer in decimal)

> Hint: You can write a program to find the answer. Type `float` in C language follows the rule of 32bits IEEE floating point format.

T8

The code below uses three XOR operation to swap two integers.

```c
void swap(int *a, int *b) {
    __ = __ ^ __;
    __ = __ ^ __;
    __ = __ ^ __;
}
```

1. Fill in the blanks to complete the code.
2. Is there anything wrong to use the `swap` function in the sorting function below? If so, how can you fix it?

```c
void sort(int *a, int n) {
    // sort a[0] ~ a[n - 1]
    for (int i = 0; i < n - 1; i++) {
        int min = i;
        for (int j = i; j < n; j++) {
            if (a[j] < a[min]) {
                min = j;
            }
        }
        swap(a + i, a + min);
    }
}
```

T9

We've got 2 blackboxes, each of which takes two numbers as input and produces a number as the output. The first one is capable of adding, while the second one is capable of multiplying. (As shown in Figure 1.10, (a) and (b)) We can combine these blackboxes to calculate $(m + n) \times p$. (As shown in Figure 1.10, (c))

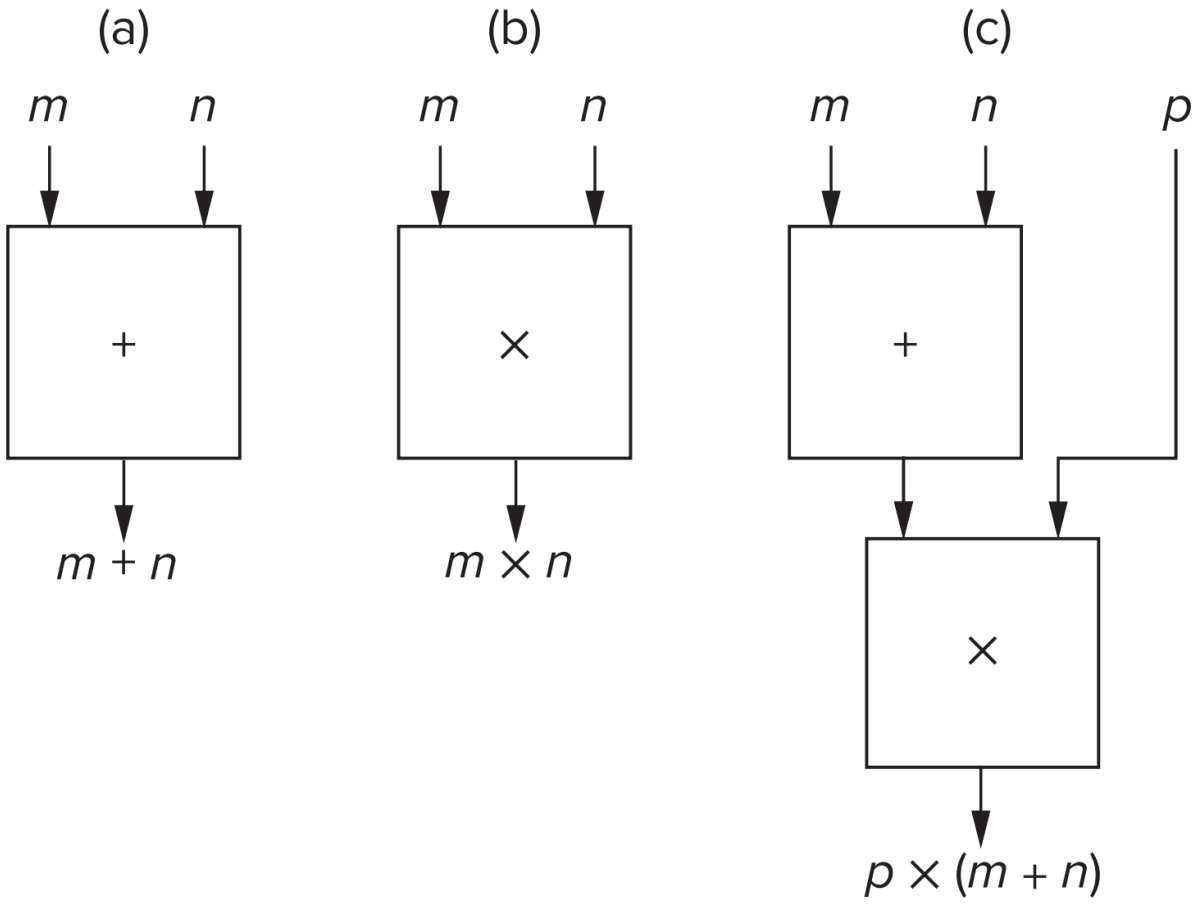

Figure 1.10, P20

You're required to draw circuits that can calculate the following expressions:

1. $y = kx + b$
2. $p = (m + n)(m - n)$

3. **Square** of the length of the longest edge of a **right triangle** (noted as c), given the lengths of the other two edges (noted as a, b).
4. The weighted average m of:

Value	Weight
v_1	w_1
v_2	w_2

> You can draw the circuits on a physical paper, on a tablet app, or, preferably, on flowchart makers like https://app.diagrams.net/ .

T10

We'd like to use binary to represent the following characters:

- `A` to `Z`
- `a` to `z`
- `0` to `9`
- 2 special characters: ` ` (space) and `.`

1. How many bits do we need to represent a single character?
2. How many bits do we need to represent a string of N characters?
3. Assume that we use 0 to represent `A`, 1 to represent `B`, and so on. So we use 63 to represent `.`. What is the binary representation of `Hello World.`?