Answer 1

A1

1

- -114 原码为 1111 0010，反码为 1000 1101，补码为 1000 1110 (负数补码为原码的反码 + 1)
- +81 原码与补码均为 0101 0001 (正数补码与原码一致)

2

- 0011 0010 是正数，原码与补码均为 0011 0010，因此为 50
- 补码为 1111 1101，是负数，反码为 1000 0010，原码为 1000 0011，因此是 -3 (补码的补码即为原码)

A2

1. 最小的补码为 1000 0000，即 -128；最大的补码为 0111 1111，即 127.
2. N 位 2 进制补码的范围为 $-2^{N-1} \sim 2^{N-1}-1$.

A3

-64。（原码为 1100 0000，反码为 1011 1111，补码为 1100 0000）

A4

1. `int` 型的范围为 $-2^{31} \sim 2^{31}-1$，因此两个 `int` 型变量相减得到的结果范围也最多为 $-2^{31} \sim 2^{31}-1$，如果变量 `a` 比变量 `b` 大超过 2^{31}，则 `a - b` 会溢出，结果为负数，此时会输出 `a < b`.
2. `unsigned int` 型取负数也是取反加一，不过结果会被作为 `unsigned int` 看待，因此 $-a$ 相当于 $2^{32}-a$，不过有一个特殊的数 `0`，其取负之后依然为 `0`，因此如果 `b = 0`，而 `a` 非零，就会输出 `a < b`.

A5

$10001011_B = 139_D$

因此为

$$
\begin{aligned}
&(1.00000000001000000001000) \times 2^{139-127} \\
=&1000000000010.00000001_B \\
=&4098.00390625_D
\end{aligned}
$$

A6

最小可以表示的浮点数为 $-(2-2^{-23}) \times 2^{127}$，也就是
1 11111110 11111111111111111111111

最小可以表示的正数为 $2^{-23} \times 2^{-126}$，也就是 0 00000000 00000000000000000000001

A7

```c
#include <limits.h>
#include <stdio.h>

union my_union {
    int a;
    float b;
};

int main(void) {
    union my_union t;
    for (int i = INT_MIN; i < INT_MAX; i++) {
        t.b = i;
        if (t.a == i) {
            printf("%d\n", i);
        }
    }
    return 0;
}
```

运行程序，得到结果-834214802, 0, 1318926965.

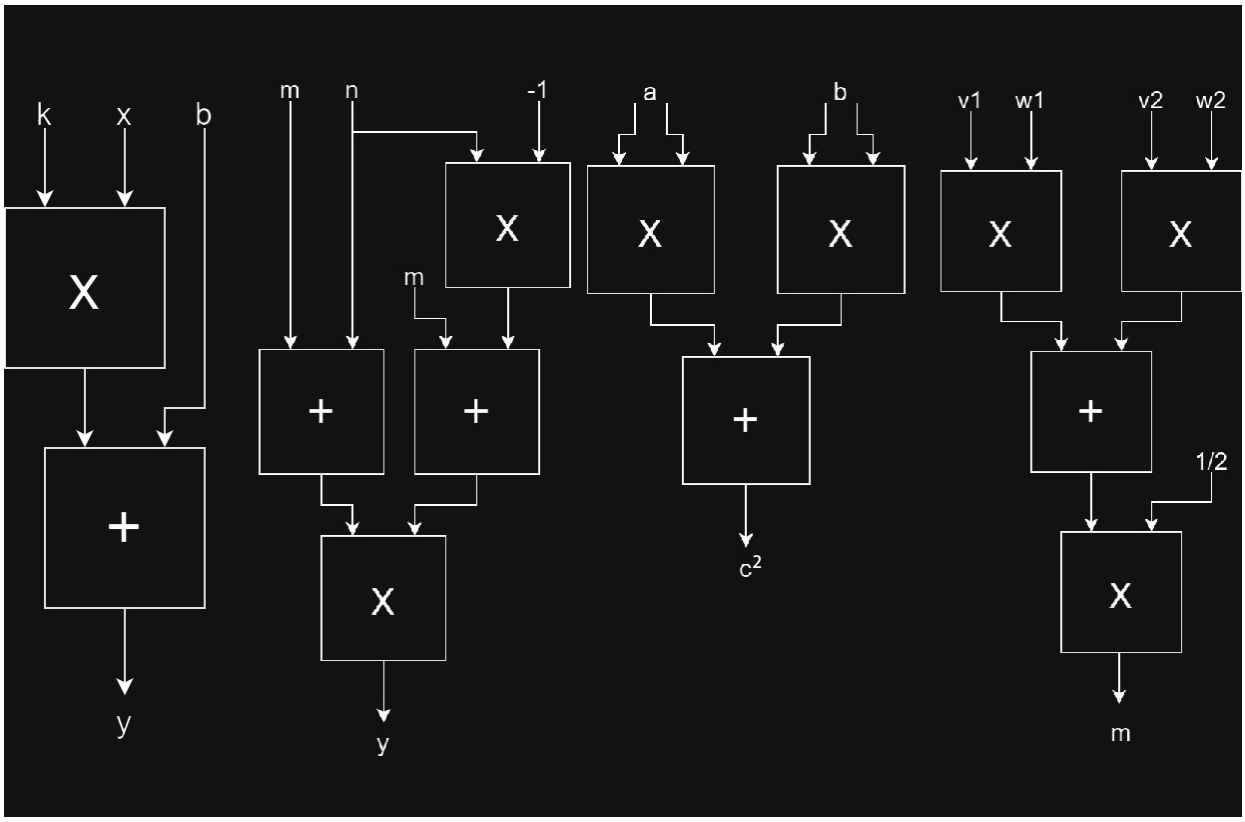
A8

1. 答案如下

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

2. 如果 `swap` 函数中的 `a` 和 `b` 指向相同的地址，则第一行异或后该地址就会变成 `0`，第二行异或会使该地址变成 `1`，第三行再异或就得到 `0`，也就是如果 `a == b` 时，会使地址上的值变成 `0`，而不再是原来数组里的值。要改正的话只需在 `swap` 中或是 `sort` 中对要交换的两个指针判断是否相等即可。

A9



- (2) 使用 $m^2 - n^2$ 计算也算对
- (4) 不乘 1/2，乘了也算对，乘了 $1/(w_1 + w_2)$ 也算对

A10

1. 共有 64 个字符，因此需要 6 位二进制数来表示。
2. $6N$
3. 000111 011110 100101 100101 101000 111110 010110 101000 101011 100101 011101 111111