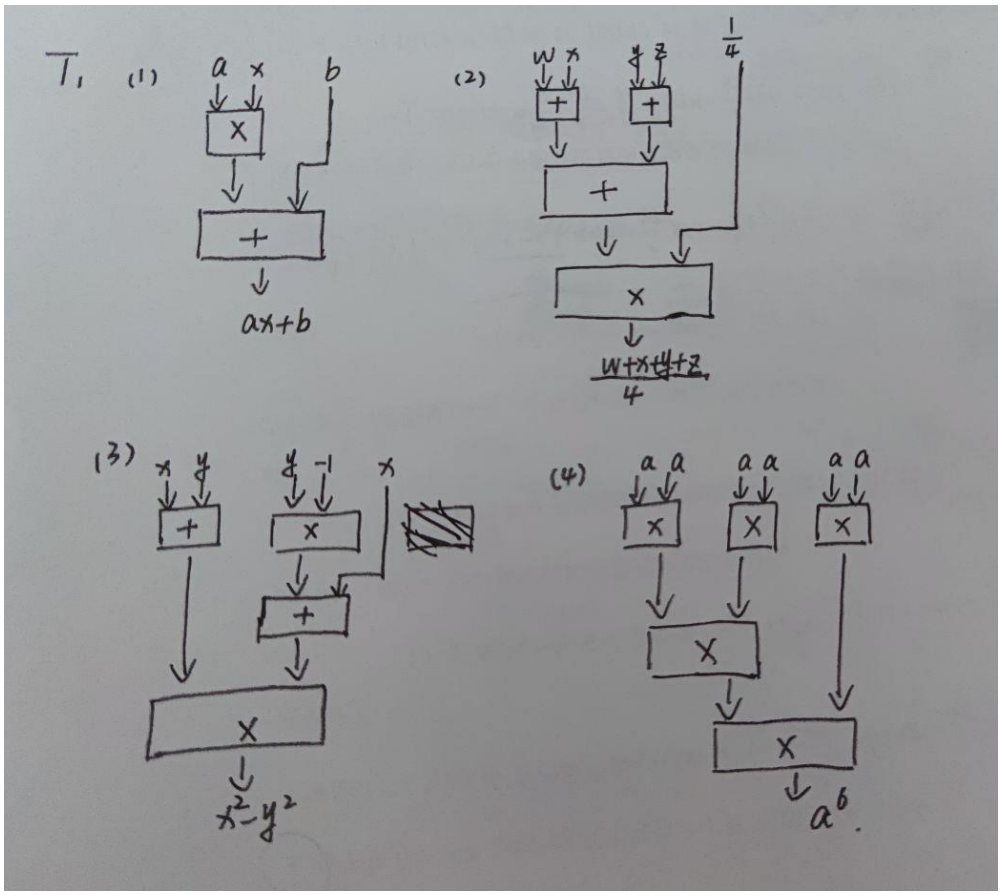
T1



T2

(1) a.10011110

b.00010111

(2) c.-62

d.17

T3

a. 110100

b. 000110

c. 10111

d. 00100

T4

a. 11101011

b. 01111110

c. 0

d. 01110001

T5

IEEE 表示：0 10000001 00010011001100110011001

实现：

```
#include <iostream>
#include <stdio.h>
//integer:整数 radix:输出的进制 result:结果数组 num:生成r进制所占位数
void transfer_int(int integer, int radix, int result[], int& num)
{
    int i, j, n;
    for (i = 0; integer > 0; i++)//除r取余
    {
        result[i] = integer % radix;
        integer = integer / radix;
    }
    num = i;//r进制位数
    for (j = 0; j < i / 2; j++)//输出顺序与计算存入数组的顺序相反
    {
        n = result[j];
        result[j] = result[i - 1 - j];
        result[i - 1 - j] = n;
    }
}

//小数转r进制(对8进制等均可使用)
//decimal:小数 radix:进制 result:结果数组 num:生成r进制所占位数
void transfer_dec(double decimal, int radix, int result[], int& num)
{
    int i;
    for (i = 0; decimal > 0 && i < 50; i++)//乘r取整
    {
        result[i] = decimal * radix;
```

```
        decimal = decimal * radix;
        decimal = decimal - int(decimal);
    }
    num = i;
}

int main()
{
    int integer;//定义整数部分
    double decimal;//定义小数部分
    int s, E[8] = { 0 }, m[50] = { 0 };//定义浮点数符号位，阶码数，尾数

    while (true)
    {
        double number;//输入浮点数
        printf("请输入一个浮点数：");
        scanf_s("%lf", &number);
        int i=0, j=0;
        if (number >= 0)
        {
            s = 0;
        }
        else
        {
            s = 1;
            number = -number;
        }//按照输入数据的正负分类
        integer = (int)number;
        decimal = number - integer;
        int m_n1, m_n2;
        transfer_int(integer, 2, m, m_n1);
        transfer_dec(decimal, 2, m + m_n1, m_n2);
        //规格化，计算阶数，尾数
        int pn = 0;//阶数

        if (integer > 0)//小数点左移
        {
            pn = m_n1 - 1;//阶数
            for (i = 0; i < 23; i++)//去掉首位默认的1
            {
                m[i] = m[i + 1];
            }
        }
        else//小数点右移
```

```
    {
        for (i = 0; m[i] == 0; i++)
        {
        }
        pn = -i - 1;
        for (j = 0; j < 23; j++)//去掉左边无效的0和第一个的1
        {
            m[j] = m[j + 1 + i];
        }
    }//计算阶码
    int p1[8], pn1;
    transfer_int(pn + 127, 2, p1, pn1);//阶数转二进制
    if (pn1 < 8)//不足8位左边补0
    {
        for (j = 0; j < 8 - pn1; j++)
        {
            E[j] = 0;
        }
        for (int k = 0; k < pn1; k++)//得出完整p
        {
            E[j + k] = p1[k];
        }
    }
    else
    {
        for (i = 0; i < 8; i++)//得出完整p
        {
            E[i] = p1[i];
        }
    }

   //依次输出符号位，阶码和尾数
    printf("%d ", s);
    for (i = 0; i < 8; i++)
    {
        printf("%d", E[i]);
    }
    printf(" ");
    for (i = 0; i < 23; i++)
    {
        printf("%d", m[i]);
    }
    printf("\n");
}
```

```
    return 0;
}
```

T6

1011.28125

T7

(1)

a. 10000101

b. 11111111

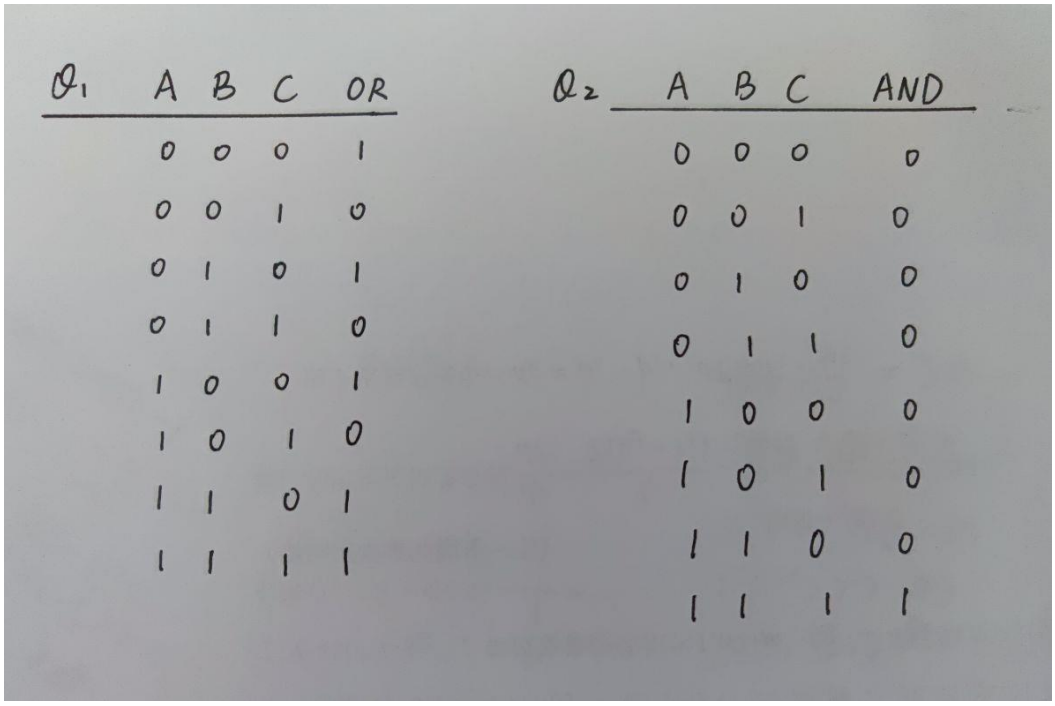
c. 10101111

(2)

d. X9BFD

e. X5007

T8



Q_1	A	B	C	OR
	0	0	0	1
	0	0	1	0
	0	1	0	1
	0	1	1	0
	1	0	0	1
	1	0	1	0
	1	1	0	1
	1	1	1	1

Q_2	A	B	C	AND
	0	0	0	0
	0	0	1	0
	0	1	0	0
	0	1	1	0
	1	0	0	0
	1	0	1	0
	1	1	0	0
	1	1	1	1

T9

（1） BQoN

（2） 用于加密传输 8bit 字节码

T10

$$(2^{23}-1)\times 2^{8}=2^{31}-2^{8}$$

T11

“没做出来”