Homework1

BY 唐志鹏 SA23011068

2.58

- 构造一个数它的首尾不同，然后看一下这个数的二进制表示的低8位，是和机器中的高8位相等（小端）还是低8位相等（大端）

```c
typedef unsigned char *byte_pointer;
typedef unsigned char byte;

int is_little_endian() {
    int testNum = 1;
    byte lowBit = (byte) (testNum & 0xFF);
    byte_pointer pointer = (byte_pointer) &testNum;
    if(*pointer == lowBit) {
        return 1;
    }
    else {
        assert(*(pointer + sizeof(int)-1)==lowBit);
        return 0;
    }
}
```

2.61

```c
int judge_C_Experiment(int x) {
    int res = 0;
    //任何位都是1，值就是-1
    res |= !(x - (-1));
    //任何位都是0，值就是0
    res |= !x;
    //最低位都是1，就是与0xFF或，还是原数
    res |= !((x | 0xFF) - x);
    unsigned int shift = (sizeof(int)-1)<<3;
    res |= !((x & ~(0xFF << shift)) - x);
    return res;
}
```

2.77

```c
int expression_A(int x) {
    return (x << 4) + x;
}

int expression_B(int x) {
    return x - (x << 3);
}

```

```c
int expression_C(int x) {
    return (x << 6) - (x << 2);
}

int expression_D(int x) {
    return (x << 4) - (x << 7);
}
```

2.84

- 对于整数，浮点数的位级表达对应的无符号数的大小关系是一致的，因此只需要关注符号位

```c
unsigned f2u(float x) {
    return *(unsigned*)&x;
}

int float_le(float x, float y) {
    unsigned ux = f2u(x);
    unsigned uy = f2u(y);
    unsigned sx = ux >> 31;
    unsigned sy = uy >> 31;
    return (sx && !sy) || (sx && sy && (ux>=uy)) ||
           (!sx && ! sy && (ux<=uy)) || (!sx && sy && !ux && !uy);
}
```

2.89

- A
  - 恒为真，因为虽然int转float可能会发生舍入，但是double转folat也会同样舍入
- B
  - 不恒为真，int转double没有精度损失，但是int运算会溢出，double运算不会溢出
- C
  - 恒为真，虽然浮点数加法没有结合律，但是double表示int的范围不会出现舍入。
  - x=INT_MAX/2-1, y= -INT_MAX/2+1, z=1时仍然符合
- D
  - 不恒为真，浮点数乘法没有结合律，而且double表示int的范围会出现舍入
- E
  - 不恒为真，有一个为0时会出现NaN

2.91

- A
  - 0x40490FDB
  - 0　10000000　10010010000111111011011
  - 二进制小数是11.00100100000111111011011
- B
  - 22/7=3+1/7
  - 从2.83可以看出，对于分母为2^k-1的循环小数，可以构造为：循环部分长度为k的二进制表示
  - 1/7　→　0.001[001]

    - 22/7 → 11.001[001]
- C
    - 0x40490FDB → 11.00100100001111111011011
    - 22/7 → 11.001001001...
    - 从第9位开始不同