“计算机组织结构”作业 02 参考答案

1. 假定变量 int i = 1234567890、float f = 1.23456789e10，sizeof(int)=4，判断以下表达式的结果（True / False）：

   1) i == (int)(float)i；i == (int)(double)i

False；True

第一个表达式：因为 IEEE754 的 float 类型中，尾数的小数部分只有 23 个二进位和一位隐藏位，共 24 位有效位数，理论上 float 的十进制有效位数为 7 位，而 i 中有 9 位十进制有效位,用二进制表示为 11101011011111100000000111 1101B,从二进制有效值的编码来看，将 i 转换为 float 类型时会发生 3 位有效数字的丢失，再转换为 int 类型时，其值无法还原了。

第二个表达式：因为 IEEE754 的 double 类型中，尾数的小数部分有效位数为 52+1=53 个二进位，而 int 类型的有效位数只有 31 个二进位，因此对于任何一个 int 类型的变量，转换为 double 后，其精度不会有损失，再转换回 int 类型时，其值不变。

   2) f == (float)(int)f；f == (float)(double)f

False；True

第一个表达式：第一个表达式：因为变量 f 的值超过了 int 类型可表示的最大值，因而将 f 转换为 int 类型后再转换回 float 类型时，其值已经改变了。

第二个表达式：因为 double 类型的精度比 float 类型高，任何 float 类型的变量转换为 double 后再转换回 float 类型时，其值不变。

2. 下图是某个 java 程序及其该程序的若干组执行结果。请根据 IEEE754 标准的舍入规定对运行结果进行解释说明，并通过分析得出 float 变量的有效十进制位数。

```
public static void main(String[] args) {
    float f;
    BufferedReader br = new BufferedReader(new InputStreamReader(System.in));
    while (true) {
        System.out.print("please enter a number: ");
        try {
            f = Float.parseFloat(br.readLine());
            System.out.printf("%f\n", f);
        } catch (IOException e) {
            e.printStackTrace();
        }
    }
}
```

```
please enter a number: 61.419997
61.419998
please enter a number: 61.419998
61.419998
please enter a number: 61.419999
61.419998
please enter a number: 61.42
61.419998
please enter a number: 61.420001
61.420002
please enter a number:
```

6 位

该程序的功能非常简单，就是从键盘上输入一个实数，赋给一个 float 型变量后再从屏幕上输出。从运行结果来看，61.419998 和 61.420002 是两个可表示数，两者之间相差 0.000004。当输入数据是一个不可表示数时，机器将其转换为最邻近的可表示数。

因为 61 = 111101B = 1.11101B×25，如果将 float 型数据的规格化正数的表示范围以 2^i (-126≤i≤127) 为分割点划分成若干区间，61.419998 应该位于区间 [2^5,2^6]，该区间相邻可表示数之间的间隔为 2^-23 × 2^5 = 2^-18 = 0.0000038… ≈ 0.000004，从上述分析结果来看，该区间相邻两个可表示数之间的间隔就是 0.000004。因此，在 61.419998 前面的可表示数为 61.419994，后面的可表示数为 61.420002。

从直观的角度看，输入 61.419997 和 61.419999 时，其最靠近的可表示数为 61.419998；而 61.420001 的最邻近可表示数为 61.420002。

但在特殊的中点值情况下，如输入为 61.42 时，十进制形式的 61.420000 位于可表示数 61.419998 和 61.420002 的中点。这时与机器的舍入机制有关，需要根据机器内部的二进制表示来判断。

61.42 的 double 表示为 1 10000000100 1110101101011100001010001111010111000010100011110110。

从二进制表示形式来看，在 float 能表示的有效值部分的后一位为 0，因此后续数字被舍弃，因此，61.42 对应输出的可表示数为 61.419998。

实际上，浮点数所有的可表示数都是根据二进制来选择的，因此中点值的可表示数需要根据实际情况来判断。

目前几乎所有机器中 float 型变量都是采用 IEEE754 单精度浮点数格式表示，其二进制有效位数为 24 位，因此能精确表示的十进制有效位数为 6 位：

对于题干中出现的例子，61.419998 位于区间 [2^5，2^6]，该区间相邻可表示数之间的间隔为 2^-23 × 2^5 = 2^-18 = 0.0000038…，间隔大小没有超过第 7 位有效数字。但对于某些特殊情况，区间内可表示数的间隔的值超过该指数区间的第 7 位有效数字时，则不能保证第 7 位精确表示。例如，9.007203e15 = 9007203e9 位于区间 [2^53，2^54]，该区间相邻可表示数之间的间隔为 2^-23 × 2^53 = 2^30 = 1.073741824e9，间隔超过了第七位有效数字。反映在二进制中时，该值的第 24 位为 0，因此后续的值被舍弃，于是出现了第 7 位有效数字不准确的情况。这种情况只会出现在两个可表示数的间隔大于第七位有效数字时，两个可表示数位于刚好四舍五入的左右两端，且小的可表示数更接近实际数字。

[熊丘桓，201250172]

3. 【2015 统考真题】由 3 个“1”和 5 个“0”组成的 8 位二进制补码，能表示的最小整数是（　　）。

A. -126　　B. -125　　C. -32　　D. -3

B

根据补码真值计算的公式，3 个“1”应该分别位于最高位和最低的两位。因此，最小的整数为 1000011，换算成真值为：-2^7+2^1+2^0 = -125。

4. 【2021 统考真题】已知带符号整数用补码表示，变量 x、y、z 的机器数分别为 FFFDH，FFDFH，7FFCH，下列结论中，正确的是（　　）。

A. 若 x,y 和 z 为无符号整数，则 z<x<y

A. 若 x,y 和 z 为无符号整数，则 x<y<z

A. 若 x,y 和 z 为带符号整数，则 x<y<z

A. 若 x,y 和 z 为带符号整数，则 y<x<z

D

若 x,y 和 z 为无符号整数，则 z<y<x；若 x,y 和 z 为带符号整数，则 y<x<z。

5. 【2010 统考真题】假定变量 i、f 和 d 的数据类型分别为 int、float 和 double（int 用补码表示，float 和 double 分别用 IEEE 754 单精度和双精度浮点数格式表示），已知 i=785、f=1.5678E3、d=1.5E100，若在 32 位机器中执行下列关系表达式，则结果为“真”的是（　　）。

I. i==(int)(float)i；II. f==(float)(int)f；III. f==(float)(double)f；IV. (d+f)-d==f

A. 仅 I 和 II　　B. 仅 I 和 III　　C. 仅 II 和 III　　D. 仅 III 和 IV

B

int 为 32 位，转化为 float 时只保存 1+23 位，有可能造成精度丢失；但由于 i<1024（10 位），此时精度无丢失，所以 I 正确。

将 float 转换为 int 时，小数部分会被丢掉，因此 II 不正确。

double 的表示范围大于 float，float 转化为 double 时不会有精度丢失，所以 III 正确。

d+f 时需要对阶，造成 f 的精度丢失，因此 IV 不正确。

6. 【2011 统考真题】float 型数据通常用 IEEE 754 单精度格式表示。若编译器将 float 型变量 x 分配在一个 32 位浮点寄存器 FR1 中，且 x=-8.25，则 FR1 的内容是（　　）。

A. C104 0000H　　B. C242 0000H　　C. C184 0000H　　D. C1C2 0000H

A

X = -8.25 = -1000.01B = -1.00001B×2^3。根据 IEEE 754 表示，E = 3+127=130，转化为二进制为 10000010；尾数的最高位为隐藏位，因此为 000010…0。因此，FR1 的内容为：1 10000010 000010…0，转换为十六进制表示为 C104 0000H。

7. 【2012 统考真题】float 类型（即 IEEE 754 单精度浮点数格式）能表示的最大正整数是（ ）。

A. 2^126 - 2^103 B. 2^127 - 2^104 C. 2^127 - 2^103 D. 2^128 - 2^104

D

IEEE 754 单精度能表示的最大正整数为：1.1…1×2^(254-127)=2^128-2^104。

8. 【2013 统考真题】某数采用 IEEE 754 单精度浮点数格式表示为 C640 0000H，则该数的值是（ ）。

A. -1.5×2^13 B. -1.5×2^12 C. -0.5×2^13 D. -0.5×2^12

A

该数的二进制表示为：1100 0110 0100 0000 0000 0000 0000 0000。因此，符号位为 1；阶码为 10001100，真值为 13；尾数（含隐藏位）为 1.1B，即 1.5。所以该数的值为-1.5×2^13。

9. 【2014 统考真题】float 型数据常用 IEEE 754 单精度浮点格式表示。假设两个 float 型变量 x 和 y 分别存放在 32 位寄存器 f1 和 f2 中，若(f1)=CC90 0000H，(f2)=B0C0 0000H，则 x 和 y 之间的关系为（ ）。

A. x<y 且符号相同 B. x<y 且符号不同 C. x>y 且符号相同 D. x>y 且符号不同

A

x 的二进制表示为 1100110010010…0，符号位为 1；阶码为 10011001，尾数（含隐藏位）为 1.001；y 的二进制表示为 10110000110…0，符号位为 1；阶码为 01100001，尾数（含隐藏位）为 1.1。可见，x 和 y 的符号均为负，且 x 的阶码大于 y 的阶码，因此选 A。

10. 【2018 统考真题】IEEE 754 单精度浮点格式表示的数中，最小的规格化正数是（ ）。

A. 1.0×2^-126 B. 1.0×2^-127 C. 1.0×2^-128 D. 1.0×2^-149

A

IEEE 754 单精度能表示的最大正整数为：1.0…0×2^(1-127)=1.0×2^-126。

11. 【2020 统考真题】已知带符号整数用补码表示，float 型数据用 IEEE 754 标准表示，假定变量 x 的类型只可能是 int 或 float，当 x 的机器数为 C800 0000H 时，x 的值可能是（ ）。

A. -7×2^27 B. -2^16 C. 2^17 D. 25×2^27

A

C8000000H 的二进制表示为 1100 1000 0…0。
如果是 int 型，则真值为-7×2^27，选 A。
如果是 float 型，则真值为-2^17。

12. 【2021 统考真题】下列数值中，不能用 IEEE 754 浮点格式精确表示的是（ ）。

A. 1.2 B. 1.25 C. 2.0 D. 2.5

A

1.2 采用 IEEE 754 表示时，尾数是无限循环小数 1.001100110011…。
注：本题也可采用排除法。

13. 存储器中有一个 8 位字 11000010，假设在海明码中采用偶校验，请写出加入校验码后的数据。（示例：000000000000）

110000010010

因为是 8 位字，校验码的长度为 4，分别为 C4C3C2C1

根据公式计算可得：
C1=0⊕1⊕0⊕0⊕1=0
C2=0⊕0⊕0⊕0⊕1=1
C3=1⊕0⊕0⊕1=0
C4=0⊕0⊕1⊕1=0

12	11	10	9	8	7	6	5	4	3	2	1
1100	1011	1010	1001	1000	0111	0110	0101	0100	0011	0010	0001
D8	D7	D6	D5	C4	D4	D3	D2	C3	D1	C2	C1
1	1	0	0	0	0	0	1	0	0	1	0

因此，加入校验码后的数据为：110000010010

14. 一个 8 位字 00111001，采用海明码生成校验位后存储。假定由存储器读出数据时，计算出的校验位是 1101，那么由存储器读出的数据字是什么？（示例：00000000）

00011001

假设采用偶校验，00111001 计算出的校验码为 0111（方法见题 13），而读出的校验码为 1101。因此，数据字读出时发生了错误，而校验码读出时没有发生错误（这建立在最多只有一位发生错误的假设上）。

计算出故障字为 0111⊕1101=1010。可见是第 10 位（D6）出错。原先的数据字为 00111001，所以读出的数据字为 00011001。

注：本题也可以假设采用奇校验。

15. 已知下列字符的 ACSII 编码：A=1000001，a=1100001，0=0110000，求：

a) E 在最前面加入奇校验位后的 8 位编码；
b) e 在最前面加入奇校验位后的 8 位编码；
c) 7 在最前面加入奇校验位后的 8 位编码；
d) g 在最前面加入奇校验位后的 8 位编码；
e) Z 在最前面加入奇校验位后的 8 位编码；
f) 5 在最前面加入奇校验位后的 8 位编码。

示例（均为英文标点）:00000000,00000000,00000000,00000000,00000000,00000000

01000101,11100101,00110111,01100111,11011010,10110101

a）字母是根据排序编码的，若将 A 看作第 1 个，E 为第 5 个，即 E 的编码为 100 0101。加入奇校验后的 8 位编码为 0100 0101。

同理可知：

b）e：110 0101（7 位），1110 0101（8 位）
c）7：011 0111（7 位），0011 0111（8 位）
d）g：110 0111（7 位），0110 0111（8 位）
e）Z：101 1010（7 位），1101 1010（8 位）
f）5：011 0101（7 位），1011 0101（8 位）

16. 某计算机在信息传输中采用基于偶校验的海明码，对每个字节生成校验位。假设所传输信息的十六进制表示为 8F3CAB96H，且将信息与校验码按照故障字的顺序排列后一起传输。如果传输中没有发生任何错误，写出所接收到信息（含校验码）的十六进制表示。

示例：FFFFFFFFFFFH

8F7362A5F93AH

根据海明码的计算规则：

C1 = D1 ⊕ D2 ⊕ D4 ⊕ D5 ⊕ D7
C2 = D1 ⊕ D3 ⊕ D4 ⊕ D6 ⊕ D7
C3 = D2 ⊕ D3 ⊕ D4 ⊕ D8
C4 = D5 ⊕ D6 ⊕ D7 ⊕ D8

对各个字节计算出校验码：

8FH = 1000 1111B，校验码（C4C3C2C1）为 1011

3CH = 0011 1100B，校验码（C4C3C2C1）为 0010
ABH = 1010 1011B，校验码（C4C3C2C1）为 0111
96H = 1001 0110B，校验码（C4C3C2C1）为 0110
所以，将信息和校验码按照故障字的顺序排列后的二进制表示为：
1000 1111 0111 0011 0110 0010 1010 0101 1111 1001 0011 1010
十六进制表示为：8F7362A5F93AH

17. 假设要传送的数据信息为 100011，若约定的生成多项式位$G(x)=x3+1$。如果传输中没有出现错误，接收到的信息是什么？

示例：000000000

100011111

生成多项式 G(x)为 1001，所以将数据左移 3 位后，进行模 2 除法：

```
      100111
1001 / 100011000
       1001
        0011
        0000
         0111
         0000
          1110
          1001
           1110
           1001
            1110
            1001
             111
```

校验码为 111。

如果传输中没有出现错误，接收到的信息是：100011111。

[吴超月，131250168]

========== 分割线：以下内容不在小程序上提交 ==========

18. 设某浮点数格式为：1 位数符、5 位阶码、6 位尾数。参照 IEEE754 浮点数的解释方式，写出：

1） 规格化数的非零正数的最小值、最大值

IEEE754 为：2^-126 ~ (2-2^-23)*2^127

2^-14 ~ (2-2^-6)*2^15

2） 非规格化数的非零正数的最小值、最大值

IEEE754 为：2^-(126+23) ~ (1-2^-23)*2^-126

2^-(14+6) ~ (1-2^-6)*2^-14

3） 写出 9/16 的二进制表示。

0011 1000 1000B

9/16 = 0.1001B = 1.001 * 2^(-1)，符号位为 0，指数为-1，5 位阶码表示的指数偏置常数为 2^(5-1)-1 = 15，故移码表示为 -1 + 15 = 14 = 0 1110B，尾数部分为 00 1000B。

[徐晨，201250213]

19. 设一个变量的值为 2049，在程序中将其转换为 32 位整数补码、IEEE754 单精度浮点数

格式并打印该变量(用二进制字符串表示),找出两种编码中表示有效值的二进制序列,并说明这段序列不同的原因及浮点数表示有效值的方式的优势。

2049 = 1000 0000 0001B = +1.000 0000 0001B × 2^11

32 位补码整数表示为 0000 0000 0000 0000 0000 1000 0000 0001B

IEEE754 单精度浮点数格式表示为 0100 0101 0000 0000 0000 0001 0000 0000 0000B

上述两种表示中,存在相同的二进制序列 000 0000 0001。因为 2049 被转换为规格化浮点数后,有效值部分中最前面的 1 被隐藏,而 32 位补码整数中最前面的 1 没有被隐藏,所以除了这个 1 之后的二进制有效值序列是相同的。

这意味着浮点数的尾数可表示的位数比实际编码多一位,因而使浮点数在相同有效值位数的情况下精度能够更高。