Solution of exercise 2

```
// Check agreement with IEEE-754
// 1|10000000|10010010000111001010110

    int i;
    printf("1"); // Sign bit
    print_bits(1 + 127, 8); // Exponent = 1 + excess
    printf("1"); // 11 (=3), but omitting the hidden bit
    // Fractional part:
    f = 0.1415;
    for (i = 1; i <= 22; i++) {
        f *= 2;
        if (f >= 1) {
            printf("1");
            f--;
        } else
            printf("0");
    }
    printf("\n");
```

```
#include <stdio.h>

// print the n rightmost bits of i
void print_bits(int i, int n) {
    if (n > 0) {
        print_bits(i >> 1, n - 1);
        printf("%d", i & 1);
    }
}

int main() {
    float f = -3.1415;
    print_bits(*(int *) &f, 32);
    printf("\n");
}

// Output: 11000000010010010000111001010110
```

Solution of exercise 3

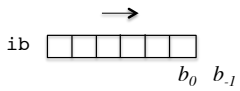
```
int mult1(short a, short b) {
   int ia = a, ib = b;
   return ia * ib;
}
```

```
int mult2(short a, short b) {
    int ia = a, ib = b;
    int p = 0, i;
    for (i = 1; i <= 15; i++) {
         if ((ib & 1) == 1)
             p += ia;
         ia <<= 1;
         ib >>= 1;
     }
    return p;
}
```


```
int mult3(short a, short b {
    int ia = a, ib = b, p = 0, i, sign = 1;
    if (ia < 0) {
        sign = -1;
        ia = -ia;
    }
    if (ib < 0) {
        sign = -sign;
        ib = -ib;
    }
    for (i = 1; i <= 15; i++) {
        if ((ib & 1) == 1)
            p += ia;
        ia <<= 1;
        ib >>= 1;
    }
    return sign == 1 ? p : -p;
}
```

```
int mult4(short a, short b) {
    int ia = a, ib = b, p = 0, i;
    int b0, bm1 = 0;

    for (i = 1; i <= 16; i++) {
        b0 = ib & 1;
        if (b0 == 1 && bm1 == 0)
            p -= ia;
        else if (b0 == 0 && bm1 == 1)
            p += ia;
        bm1 = b0;
        ia <<= 1;
        ib >>= 1;
    }
    return p;
}
```



```
int main() {
    short a, b;
    int overhead_time, total_time, start_time, i;
    const int iterations = 100000000;

    srand(7913);
    start_time = clock();
    for (i = 1; i <= iterations; i++) {
        a = rand() & 0xFFFF;
        b = rand() & 0xFFFF;
        multx(a, b);
    }
    total_time = clock() - start_time;

    srand(7913);
    start_time = clock();
    for (i = 1; i <= iterations; i++) {
        a = rand() & 0xFFFF;
        b = rand() & 0xFFFF;
    }
    overhead_time = clock() - start_time;

    printf("Multx: Time used = %f s.\n",
            (float) (total_time - overhead_time) / CLOCKS_PER_SEC);
}
```

Results

```
Mult1: Time used = 0.024970 s.
Mult2: Time used = 10.874304 s.
Mult3: Time used = 11.751448 s.
Mult4: Time used = 14.091684 s.
```