Tutorial 2
Floating-Point Numbers

Exercise 1

Convert the following decimal numbers into their binary **single-precision** floating-point representations.

1. 128
2. –32.75
3. 18.125
4. 0.0625

Exercise 2

Convert the following decimal numbers into their binary **double-precision** floating-point representations.

1. 1
2. –64
3. 12.06640625
4. 0.2734375

Exercise 3

Convert the following **single-precision** floating-point numbers into their decimal representations.

1. $1011\ 1101\ 0100\ 0000\ 0000\ 0000\ 0000\ 0000_2$
2. $0101\ 0101\ 0110\ 0000\ 0000\ 0000\ 0000\ 0000_2$
3. $1100\ 0001\ 1111\ 0000\ 0000\ 0000\ 0000\ 0000_2$
4. $1111\ 1111\ 1000\ 0000\ 0000\ 0000\ 0000\ 0000_2$
5. $0000\ 0000\ 0100\ 0000\ 0000\ 0000\ 0000\ 0000_2$

Exercise 4

Convert the following **double-precision** floating-point numbers into their decimal representations.

1. $403D\ 4800\ 0000\ 0000_{16}$
2. $C040\ 0000\ 0000\ 0000_{16}$
3. $BFC0\ 0000\ 0000\ 0000_{16}$
4. $8000\ 0000\ 0000\ 0000_{16}$
5. $FFF0\ 0001\ 0000\ 0000_{16}$

Exercise 5

Assuming that the mantissa is normalized, answer the following questions for both single- and double-precision formats.

1. Calculate the smallest and largest absolute values of a floating-point number.
2. What is the smallest number (greater than 0) which, when added to 1, gives a different result from 1?

Exercise 6

Let us consider the following program:

```
#include <stdio.h>

void main()
{
    float f1, f2, f3, r;

    f1 = 1E25;
    f2 = 16;

    f3 = f1 + f2;
    r = f3 - f1;

    printf("r = %f\n", r);
}
```

Indication : $10^{25} \approx 2^{83}$

1. Once the program has run through, what will the value of *r* be? Explain your line of reasoning.
2. Assuming that `f1=10`^{n} where *n* is a natural number, what is the largest value of *n* that still gives a correct value of *r*?
3. Assuming that *f1*, *f2*, *f3* and *r* are declared as `double`, what is the largest value of *n* that still gives a correct value of *r*?

Exercise 7

Assuming that your C compiler supports the IEEE-754 standard, write a short function in C language that converts a single-precision floating-point number into its 32-bit IEEE hexadecimal representation. The floating-point number will be passed to the function as an argument and the 32-bit hexadecimal representation will be displayed on the screen.