Description:

Write a program that can input an integer, a float, or double number and print out its bit pattern and vice versa (input a 32-bit or 64-bit pattern and output its value).

The program should be able to transfer six mode of input data:

1. (20%) 32-bit pattern → integer & float number
2. (10%) integer (< $2^{31}-1$) → 32-bit pattern
3. (10%) float number → 32-bit pattern
4. (20%) 64-bit pattern → integer & double number
5. (10%) integer (< $2^{63}-1$) → 64-bit pattern
6. (10%) double number → 64-bit pattern

For identifying the six mode above, you should input $M(=1~6)$ first, which represents each mode, and then input the number you want to transfer. Finally,

7. (20%) Report with right format should be both printed out and updated on server. Please emphasize your program “how to work”, or partial credit will be given.

Remark

1. If you want to capture the first bit while you input 32-bit pattern in
argv[n], you can use argv[n][0], and argv[n][1] for the second bit, and so on.

2. If you want to use an integer pointer to float or double to perform the bitwise operations, following code can be referred.

```c
float a = 111; // determine a float number you want to transfer
int *x; // x is a pointer to int, representing a memory address
x = &a; // x fetches a's memory address
printf("test %d %d\n", x, *x); // *x is the integer type value in memory address x
```

Command Line:

./hw3 M transfer_num

(illegal input is not considered)

Output:

Output one or two values following “integer: “,”float: “,”double: “,”32-bit: “,”64-bit: “(lower case and note space) depending M.

Example

> ./hw3 1 1110000000000000000000000000000000
> integer: -536870912
> float: -36893488147419103232.000000

> ./hw3 2 111
> 32-bit: 000000000000000000000000000000001101111
> .hw3 3 111
> 32-bit: 01000010110111100000000000000000

(Note: If you output 32-bit / 64-bit pattern, you must print 32 / 64 bits even if all bits are 0.)