Problem 158: Shifty Bits

Difficulty: Hard

Author: Matt Marzin, King of Prussia, Pennsylvania, United States

Originally Published: Code Quest 2021

Problem Background

Ground stations that command and control constellations of satellites must also be responsible for interpreting data from those satellites. To make data easier to transmit, data values from multiple sources are often combined into a single data string through a process called "commutation." Once received by a ground station, this data must be broken apart again so it can be properly analyzed; this process is called "decommutation" and the individual values are referred to as "measurands."

Problem Description

You're working with Lockheed Martin's Space Systems division to develop a new data decommutation algorithm. Your system will receive commutated data as a hexadecimal string; this must be translated into binary in order to properly separate the different measurands you'll be looking for. For example, the hexadecimal string 0xB312C675 would be converted like so:

Hex	B				3				1				2				C				6				7				5			
Decimal	11				3				1				2				12				6				7				5			
Binary	1	0	1	1	0	0	1	1	0	0	0	1	0	0	1	0	1	1	0	0	0	1	1	0	0	1	1	1	0	1	0	1

There are three main elements needed to identify and decommutate a measurand:

1. A data type, which determines how the information will be interpreted.
2. An offset marking where the data starts. The right-most bit will have an index of 0; the offset indicates at which index your data begins.
3. The length of the measurand, or the number of bits it contains.

The data types you'll encounter in this problem are listed below:

Data Type	Length	Minimum Value	Maximum Value
int	2 to 32 bits	-2,147,483,648	2,147,483,647
uint (unsigned int)	1 to 24 bits	0	16,777,215
float	32 bits	3.4E-38	3.4E38
double	64 bits	1.7E-308	1.7E308

Your programming language should have built-in functionality for converting binary values to float and decimal numbers; we strongly recommend using that. For int and uint values, however, the bit length

will vary, and may require your own interpretation. For int values, remember that regardless of the length of data, the leftmost bit will always indicate the sign of the value (positive = 0, negative = 1). Uint values will always be positive.

For example, let's use the data string presented above to extract an int measurand with a length of 6 and an offset of 4:

1	0	1	1	0	0	1	1	0	0	0	1	0	0	1	0	1	1	0	0	0	1	1	0	0	1	1	1	0	1	0	1

$$\left(-1 \times (1 \times 2^5)\right) + (0 \times 2^4) + (0 \times 2^3) + (1 \times 2^2) + (1 \times 2^1) + (1 \times 2^0) = ?$$
$$-32 + 0 + 0 + 4 + 2 + 1 = -25$$

Sample Input

The first line of your program's input, received from the standard input channel, will contain a positive integer representing the number of test cases. Each test case will include:

- A line containing a hexadecimal string, prefaced with '0x', and followed by any number of uppercase hexadecimal characters.
- A line containing a positive integer, M, representing the number of measurands contained in the data string.
- M lines, each containing the following information about one of the measurands to decommutate, separated by spaces:
  - A string indicating the data type of the measurand; one of int, uint, float, or double
  - A non-negative integer indicating the index offset at which the measurand begins
  - A positive integer indicating the length of the measurand in bits

```
2
0xDEADFACEBEEFBABE
3
double 0 64
int 8 16
uint 24 8
0x0123456789ABCDEF
2
int 5 6
float 16 32
```

Sample Output

For each test case, your program must print the value of each measurand in the order they were presented, one per line. Print doubles and floats to a precision of 5 decimal places; use lowercase scientific notation (as shown below) if the absolute value is greater than 99999.99999 or less than 0.00001. When scientific notation is required, print the exponent as e+### or e-### (as applicable), where ### is the three-digit exponent used, including any leading zeroes.

```
-1.19794e+148
-4166
190
-17
3704.60425
```