Problem 126: Endian Handler

Difficulty: Hard

Author: Louis Ronat, Denver, Colorado, United States

Originally Published: Code Quest Australia 2019

Problem Background

Most data structures are stored in multiple bytes, and the order of these bytes is somewhat arbitrary. The arrangement of bytes is usually managed by the operating system, and the system expects the bytes of each data structure to be in the supported order. When transferring multi-byte data structures (e.g. integers not represented as ASCII) this can result in the data being completely misread.

For 32-bit (4-byte) integers, the bytes may be arranged starting with the most significant (largest byte) and descending to the least significant (smallest byte), or by starting with the least significant and ascending to the most significant. These storage arrangements are called big-endian and little-endian, respectively. The name comes from *Gulliver's Travels*, where Gulliver encounters a nation where civil war breaks out between factions that disagree on whether to open hard-boiled eggs from the small end or the large end.

Problem Description

As an intermediary between the warring factions, you must translate arrays of 4 bytes from each faction (given as hexadecimal strings) into standard unsigned base-10 integers.

Below is a table showing examples of converting unsigned base-10 integers to their big- and little-endian hexadecimal values:

Base-10 Integer	Hexadecimal Bytes (in order of significance)				Big Endian Hexadecimal	Little Endian Hexadecimal
	Most	>	>	Least		
1	00	00	00	01	00000001	01000000
16,777,216	01	00	00	00	01000000	00000001
277	00	00	01	15	00000115	15010000
1,782,954,783	6A	45	BB	1F	6A45BB1F	1FBB456A

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 consists of a single line containing an 8-character hexadecimal string, a space, and an endian type (either "BIG" or "LITTLE").

```
3
00000001 BIG
00000001 LITTLE
6A45BB1F LITTLE
```

Sample Output

For each test case, your program must output the unsigned base-10 integer equivalent to the input hexadecimal string, interpreted according to the specified endianness.

```
1
16777216
532366698
```