Problem 103: LMCoin

Difficulty: Medium

Author: Ben Fenton, Faslane, Helensburgh, United Kingdom

Originally Published: Code Quest 2019

Problem Background

Bitcoin and other "cryptocurrencies" are fast becoming very popular in today's world. They are vastly different from traditional ways of paying for things, in that they don't' require a bank or credit card company to act as a mediator between the buyer and seller. This avoids having to give the bank or other organization a transaction fee.

Instead, bitcoin is given directly to the other party. However, this leaves a big problem - how do you prove you've paid for something? Or that you even have the money to pay in the first place without someone vouching for you? This is known as the "double spending problem."

Instead of a bank recording all transactions in a central ledger, all users of bitcoin record all of the transactions at the same time. This means that any attempt to fool the system would be noticed and the transaction rejected. This is done through something called "blockchain." In this problem, we will build a simple blockchain for LMCoin, our very own digital currency.

As the name implies, a blockchain consists of several "blocks" of data, each representing a separate transaction. Each block is identified by a unique "hash", a value that is generated using all of the information stored in a block; this includes the hash of the previous block in the chain. Therefore, as we go along the chain, the integrity of the chain increases and ensures past transactions cannot be altered. Bitcoin and other cryptocurrencies are created by "mining" them; adding a new block to the chain that produces a unique "hash" value within a certain range - this involves a lot of guesswork!

Problem Description

You will need to write a program that implements the LMCoin blockchain. As the name implies, a blockchain consists of several "blocks" of data, each representing a separate transaction. Besides the start block (which doesn't include the last item), each block has four pieces of information:

1. A timestamp indicating when the block was created
2. Some data (such as a pizza order, for example)
3. An index (this block's position in the chain)
4. The hash of the previous block in the chain

The hash algorithm used by our currency will work as follows:

$$H_n = \frac{(T_n + V_n + n + H_{n-1}) * 50}{147}$$

Where:

- n is the index of the block within the chain; the first block in the chain has index 1.
- H_n is the hash for the block at index n. ($H_0 = 0$)
- T_n is the timestamp of the block at index n.
- V_n is the numeric value of the data in the block at index n (explained below).

A block's data is converted to a numeric value by adding the values of each letter in the data string, as shown below:

Letter	a	b	c	d	e	f	g	h	i	j	k	l	m
Value	1	2	3	4	5	6	7	8	9	10	11	12	13
Letter	n	o	p	q	r	s	t	u	v	w	x	y	z
Value	14	15	16	17	18	19	20	21	22	23	24	25	26

For example, the value of the string "cheese" is 3 + 8 + 5 + 5 + 19 + 5 = 45.

Timestamps will be in the format DDMMYYHHMM (two digits each for day, month, year, hour, and minute, respectively); for example, 8:30 AM on 27 April 2019 would be written as 2704190830.

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 the following lines of input:

- A line containing a space-separated list of data values for the first ten blocks in a chain. Data values will contain only lowercase letters.
- A line containing a space-separated list of timestamps for the first ten blocks in a chain. Timestamps will be in the format described above.

```
2
pepperoni veggie ham peppers cheese olives mushroom chicken beef bacon
2704191000 2704191030 2704191100 2704191130 2704191200 2704191230 2704191300
2704191330 2704191400 2704191430
candy candy salad chips pretzel icecream apple fries cookie sandwich
2602201200 2602201300 2702201200 2702201300 2802201200 2802201300 2902201200
2902201300 0103201200 0103201300
```

(Note that there are only four lines of input above following the number of test cases; the timestamps are too long to fit on this page.)

Sample Output

For each test case, your program must output the hash of the tenth block in the chain, calculated using the provided values. Printed hash values should be rounded to the nearest whole number. Do not round any intermediate hash values used for calculations.

```
1393884230
219309065
```