Eighth Annual
University of Oregon Programming Competition

Saturday, April 17, 2004

Many Happy Returns

Investment performance is usually measured by an annualized rate of return. For example, a mutual fund may quote its current one year performance by saying how large an investment would grow in a year: if you put $1,000 in the fund on January 1, you would have $1,070 on December 31. You could compare this to growth in a savings account if you put $1000 in a savings account with a 6.77% interest rate that was compounded daily, then after one year (365 days) your account would be $1,070. You can calculate this future value from the formula: $FV = P \times (1 + r/365)^k$, where P is the original amount, r is the annual interest rate, and k is the number of days in the period. This equation could also be solved for the rate when all the other values are known, and so for comparison purposes, we could determine that the mutual fund had an annual total return rate of 6.77%.

The trouble with comparisons like this is that the rate quoted for the mutual fund assumes that you make only one investment, and it must be made on January 1. Likewise, solving for the rate in the formula can only be done if you make a single investment in the mutual fund. But it is more likely that you start your investment on some date other than January 1, and you add to your investment throughout the year. If we did this in a savings account with a fixed interest rate, we could calculate the future value of each deposit at the end of the year and add them to get the total value of the account. But calculating the return rate for a mutual fund that is equivalent to a savings account is a much more difficult equation to solve.

In this problem, you are given the value of an investment as of the end of the year, and the dates and amounts of additions to the investment during the year. You are to calculate the interest rate that would have yielded the same value in a savings account with daily compounding and the same set of deposits. The input data will be for the year 2003 (so you do not have to worry about the number of days in a leap year).

The input will be a series of investment descriptions. The first line of the input will be the number of investments for which you will calculate the rates of return. Each investment will begin with a line that has the number of deposits, the value at the end of the year, and the date (for this problem that date will always be `12/31/2003`). This will be followed by the list of deposits, where each deposit is a line beginning with +, followed by the deposit date, and the deposit amount. Dates are in the form MM/DD/YYYY.

For each investment description, output the rate of return in the form shown, with accuracy to two decimal places. This rate should be equivalent to daily compounding from the date of each deposit to the end of the year, inclusive.

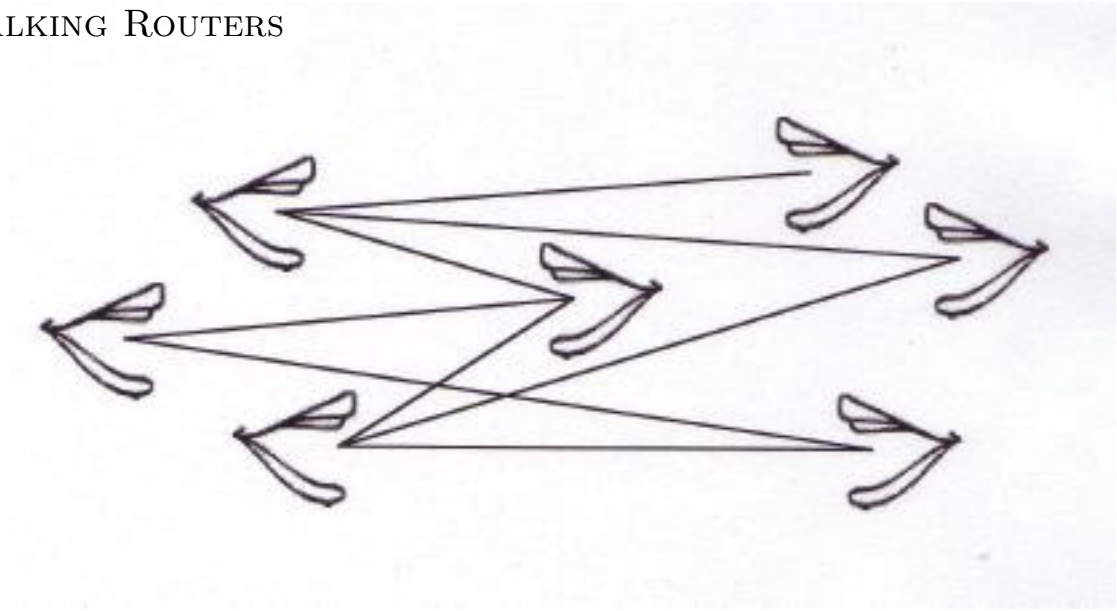
Sample Input

```
3
1 1070 12/31/2003
+ 1/1/2003 1000
2 2105 12/31/2003
+ 1/1/2003 1000
+ 6/30/2003 1000
4 7129.55 12/31/2003
+ 2/11/2003 1000
+ 4/15/2003 1500
+ 6/1/2003 999.90
+ 8/9/2003 3000
```

Sample Output

```
$1070.00 represents a return of 6.77%
$2105.00 represents a return of 6.77%
$7129.55 represents a return of 16.00%
```

Talking Routers



Suppose routers in a network talk to each other by exchanging control messages. A control message sent between a pair of routers traverses a path of edges and routers in the network. Every day each pair of routers talk to each other enough times so that control messages between them follow every possible shortest path (in terms of number of edges) between the pair. Each router records the number of distinct (unordered) pairs of routers that have exchanged messages through it, i.e., not counting pairs that include that router.

In this problem you must determine which routers record the most such pairs.

The first line of input will contain an integer n indicating how many problem instances will follow. Each instance then will start with a line indicating the number r of routers (assumed to be labeled 0 to $r-1$) and the number e of edges in the network; the next e lines will each contain two router labels representing an undirected edge of the network.

The output for each problem should be a list of routers observing a maximum number of router pairs followed by that maximum number, all on one line.

Sample Input

```
2
3 2
0 1
1 2
6 6
0 1
1 2
2 3
3 4
4 5
5 0
```

Sample Output

```
1  1
0  1  2  3  4  5  3
```

Spelling Suggestions

The first spelling checker programs simply determined whether each word in a document was present in a dictionary. Modern spelling checkers also suggest corrections to a misspelled word. One of the methods used to find possible corrections is to search for dictionary words within a given *edit distance* of a misspelled word.

The edit distance between two words is defined relative to a set of *edits*, which are operations on a word. For example, one possible edit might be inserting a letter. The edit distance between word A and word B, relative to the set of edits E, is the minimum number of operations from E needed to transform A to B. For example, if the edit operations E are limited to insertion and deletion of a single character, then the edit distance between "coast" and "court" is 4 (delete 'a' and 's', insert 'u' and 'r'). If the edit operations E also include substituting one character for another, then the edit distance between "coast" and "court" is 2 (substitute 'u' for 'a', 'r' for 's').

For this problem, the set of edits will include single character substitutions, deletions, and insertions, and your program must find all the words in a dictionary within edit distance 2 of a given word. The dictionary, which will be provided in a file named `dict`, may have up to 200,000 words, so some care may be necessary to check each word efficiently.

The first line of input to your program is an integer, n, indicating the number of words to follow, one to a line. The dictionary file `dict` will be a file consisting of at least one and no more than 200,000 lines of text, each line will consist of a single word, although some accented and non-alphabetic characters may be present.

For each of the n instances, the output consists of those dictionary words whose edit distance from the input word is 0, 1, or 2, where the possible edits are inserting a single character, deleting a single character, or substituting one character for another.

Each such word should be printed on the standard output, one word per line, in any order. There should be a blank line following each instance.

For this sample, assume the file `dict` consists of:

```
alpha
beta
gamma
almanac
almond
```

Sample Input	Sample Output
`3`	`alpha`
`alpo`	
`almand`	`almanac`
`grammar`	`almond`
	`gamma`

Modular Sums

Consider the numbers that can be obtained by summing a subset of $\{1, 2, \ldots, r\}$. Those 2^r sums are not all distinct. For example, with $r = 5$, the sum 6 can be obtained in 3 ways

$$6 \;=\; 1+5 \;=\; 2+4 \;=\; 1+2+3.$$

If we relax the condition and only ask how many subset sums are congruent to 6 modulo 9, then there are now 4 of these:*

$$6 \;\equiv\; 1+5 \;\equiv\; 2+4 \;\equiv\; 1+2+3 \;\equiv\; 1+2+3+4+5 \pmod 9.$$

In this problem, you are given integers r, s, and t, and must determine how many subsets of $\{1, 2, \ldots, r\}$ sum to s modulo t.

The first line of input will contain an integer n indicating how many problem instances will follow. The instances will then appear, one to a line, as a triple of integers

$$r \;\; s \;\; t$$

each between 1 and 50, separated by single spaces.

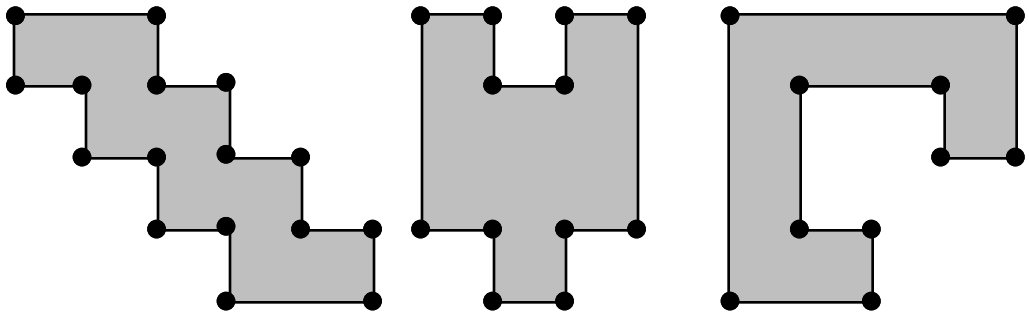
Your output for each instance r s t should be a line with a single number that indicates the number of subsets of $\{1, 2, \ldots, r\}$ that sum to s modulo t.

Note: The empty set is assumed to sum to 0.

Sample Input	Sample Output
4	4
3 3 3	4
5 6 9	16
4 7 1	0
2 4 5	

*We say that *a is congruent to b modulo m*, or $a \equiv b \pmod m$, if $a - b$ is a multiple of m.

Mondrian Land Rush



The planet Mondrian has been settled by a colony of artists. On the occasion of the recent Mondrian Land Rush, prospective settlers were not restricted to claiming dull rectangular parcels of land. Instead, they were permitted to use multiple stakes to mark the corners of their chosen properties. The perimeter of the property would then be comprised of line segments joining successive stakes with the only stipulations being that

- No more than 50 stakes may be used.
- The number of feet between successive stakes must be an integer ≤ 1000.
- the segment joining two successive stakes must run either North/South or East/West, with North/South segments alternating with East/West segments.

To register their property claims, the Mondrianese outline their lot shape by starting at the first stake, then indicating the compass direction and number of feet from each stake to the next, with the last segment necessarily ending at the starting stake. A bit less comfortable with the mathematical details than with the design, the artists occasionally have difficulty computing the area of their lots. Thus, you have the opportunity for some profitable consulting.

In this problem, you will be given settlers' outlines of their lots and must compute the area of the lot.

The first line of input to your program is an integer, n, indicating the number of Mondrianese lots to follow. Each instance is then described as follows : the first line will contain a single number m indicating the number of stakes used; the next m lines will then have a sequence of perimeter segments, each line indicating the compass direction and number of feet to the next stake, with one space between the direction (N,E,S,W) and the corresponding distance. The first segment starts at some initial stake and the last segment terminates at that initial stake; the perimeter does not otherwise intersect itself.

For each input lot, a line of your output should indicate the number of square feet in the lot.

Sample Input

```
3
4
N 200
E 200
S 200
W 200
6
W 400
N 100
E 300
N 200
E 100
S 300
8
E 100
N 100
E 100
S 100
E 100
N 200
W 300
S 200
```

Sample Output

```
40000 square feet
60000 square feet
50000 square feet
```