ACM International Collegiate Programming Contest
2001 East Central Regional Practice Contest
Ashland University
University of Cincinnati
Western Michigan University
Sheridan University
November 9, 2001

Sponsored by IBM

Rules:

1. All questions require you to read the test data from standard input and write results to standard output. You cannot use files for input or output. Additional input and output specifications can be found in the General Information Sheet.
2. All programs will be re-compiled prior to testing with the judges' data.
3. Non-standard libraries cannot be used in your solutions. The Standard Template Library (STL) and C++ string libraries are allowed.
4. Programming style is not considered in this contest. You are free to code in whatever style you prefer. Documentation is not required.
5. All communication with the judges will be handled by the PC^2 environment.
6. The allowed programming languages are C, C++ and Java.
7. Judges' decisions are to be considered final. No cheating will be tolerated.
8. There are **three** questions to be completed in **one hour and twenty minutes**.

Problem A: The Worm Turns

Worm is an old computer game. There are many versions, but all involve maneuvering a "worm" around the screen, trying to avoid running the worm into itself or an obstacle.

We'll simulate a very simplified version here. The game will be played on a 50×50 board, numbered so that the square at the upper left is numbered $(1, 1)$. The worm is initially a string of 20 connected squares. Connected squares are adjacent horizontally or vertically. The worm starts stretched out horizontally in positions $(25, 11)$ through $(25, 30)$, with the head of the worm at $(25, 30)$. The worm can move either East (`E`), West (`W`), North (`N`) or South (`S`), but will never move back on itself. So, in the initial position, a `W` move is not possible. On each move, the worm moves to the square in the direction given and will maintain its length. Thus the only two squares occupied by the worm that change in any move are at its head and tail. Note that the head of the worm can move to the square just vacated by the worm's tail.

You will be given a series of moves and will simulate the moves until either the worm runs into itself, the worm runs off the board, or the worm successfully negotiates its list of moves. In the first two cases you should ignore the remaining moves in the list.

Input

There will be multiple problem instances. The input for each problem instance will be on two lines. The first line is an integer n (< 100) indicating the number of moves to follow. (A value of $n = 0$ indicates end of input.) The next line contains n characters (either `E`, `W`, `N` or `S`), with no spaces separating the letters, indicating the sequence of moves.

Output

Generate one line of output for each problem instance. The output line should be one of the follow three:

```
The worm ran into itself on move m.
The worm ran off the board on move m.
The worm successfully made all m moves.
```

where m is for you to determine and the first move is move 1.

Sample Input

```
18
NWWWWWWWWWWSESSSWS
20
SSSWWNENNNNNWWWWSSSS
30
EEEEEEEEEEEEEEEEEEEEEEEEEEEEEE
13
SWWWWWWWWWNEE
0
```

Sample Output

```
The worm successfully made all 18 moves.
The worm ran into itself on move 9.
The worm ran off the board on move 21.
The worm successfully made all 13 moves.
```

Problem B: Undercut

Undercut is a card game where two players each have five cards numbered one through five. At each round, each player selects a card, then simultaneously reveals it. If the cards are of equal value, there is no score. Otherwise, there are two cases: the two cards are exactly one point apart (this is called an undercut), or the cards are more than one point apart. In the latter case, the person revealing the larger of the cards gets the number of points on the larger card. In the case of an undercut the player with the lower card gets the sum of the two cards. The exception to this is when the cards are 1 and 2, in which case the player with the lower card gets 6 points (instead of only 3 points). After each round, the cards are returned to the hands and they play another round.

For example, if there are 5 rounds and player A plays (in this order) 5, 3, 1, 3, 5 and player B plays 3, 3, 3, 3, 4, then the scoring for each round would be: A gets 5 points, no points, B gets 3 points, no points, B gets 9 points. The totals would be A: 5 , B: 12.

In this problem you will be given card plays for both players and must determine the final scores.

Input

There will be multiple input instances. Each instance will be one game. The first line of input for a game will be an integer $n \leq 20$. (A value of $n = 0$ terminates input.) The next two lines will each contain n integers between 1 and 5 inclusive indicating the cards played on each of n rounds. The first line are player A's card plays and the second line are player B's card plays.

Output

Each input instance should generate one line of output of the form:

`A has` a `points.  B has` b `points.`

where the values of a and b are for you to determine. A blank line should separate output lines.

Sample Input

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

Sample Output

```
A has 5 points. B has 12 points.

A has 0 points. B has 21 points.
```

Problem C: Currency Exchange

When Issac Bernand Miller takes a trip to another country, say to France, he exchanges his US dollars for French francs. The exchange rate is a real number such that when multiplied by the number of dollars gives the number of francs. For example, if the exchange rate for US dollars to French francs is 4.81724, then 10 dollars is exchanged for 48.1724 francs. Of course, you can only get hundreths of a franc, so the actual amount you get is rounded to the nearest hundreth. (We'll round .005 up to .01.) All exchanges of money between any two countries are rounded to the nearest hundredth.

Sometimes Issac's trips take him to many countries and he exchanges money from one foreign country for that of another. When he finally arrives back home, he exchanges his money back for US dollars. This has got Issac thinking about how much of his unspent US dollars is lost (or gained!) to these exchange rates. You'll compute how much money Issac ends up with if he exchanges it many times. You'll always start with US dollars and you'll always end with US dollars

Input

The first 5 lines of input will be the exchange rates between 5 countries, numbered 1 through 5. Line i will give the exchange rate from country i to each of the 5 countries. Thus the jth entry of line i will give the exchange rate from the currency of country i to the currency of country j. The exchange rate from country i to itself will always be 1 and country 1 will be the US. Each of the next lines will indicate a trip and be of the form

$n\ c_1\ c_2\ \ldots\ c_n\ m$

where $1 \leq n \leq 10$ and $c_1, \ldots, c_n$ are integers from 2 through 5 indicating the order in which Issac visits the countries. (A value of $n = 0$ indicates end of input, in which case there will be no more numbers on the line.) So, his trip will be $1 \rightarrow c_1 \rightarrow c_2 \rightarrow \cdots \rightarrow c_n \rightarrow 1$. The real number m will be the amount of US dollars at the start of the trip.

Output

Each trip will generate one line of output giving the amount of US dollars upon his return home from the trip. The amount should be given to the nearest cent, and should be displayed in the usual form with cents given to the right of the decimal point, as shown in the sample output. If the amount is less than one dollar, the output should have a zero in the the dollars place.

Sample Input

```
1 1.57556 1.10521 0.691426 7.25005
0.634602 1 0.701196 0.43856 4.59847
0.904750 1.42647 1 0.625627 6.55957
1.44616 2.28059 1.59840 1 10.4843
0.137931 0.217555 0.152449 0.0953772 1
3 2 4 5 20.00
1 3 100.00
6 2 3 4 2 4 3 120.03
0
```

Sample Output

```
19.98
99.99
120.01
```