P1 Sequence This

You are given an integer k>0 and a series of k+2 numbers n[1], n[2], …, n[k+2]. You are told that the numbers in the series are calculated using an equation of the following form for n>0:

$$n[x] = \frac{(x + a[1]) * (x + a[2]) * \ldots * (x + a[k])}{d}$$

You do not know the value of d. You do not know the values for a[1] ... a[k]. You only know that each (a[i] ≥ 0) and (d > 0) and you can assume that the product in the numerator is evenly divisible by the integer value d. You can assume that the numerator can be represented by a 32-bit integer. But you know that the formula for n[x] is a polynomial function of degree k and you know the value of k+2 points for this function. Based on this information, you actually have enough information to calculate the next number n[k+3] in the sequence! This is your task.

Input Format

Your program will read from standard input. The first line will contain a single positive integer by itself that represents the value k. The second line will consist of (k+2) integer values separated from each other by a single space. The values on the second line represent the series n[1] through n[k+2].

Output Format

Your program must write to standard output a single integer on a line by itself representing the value for n[k+3].

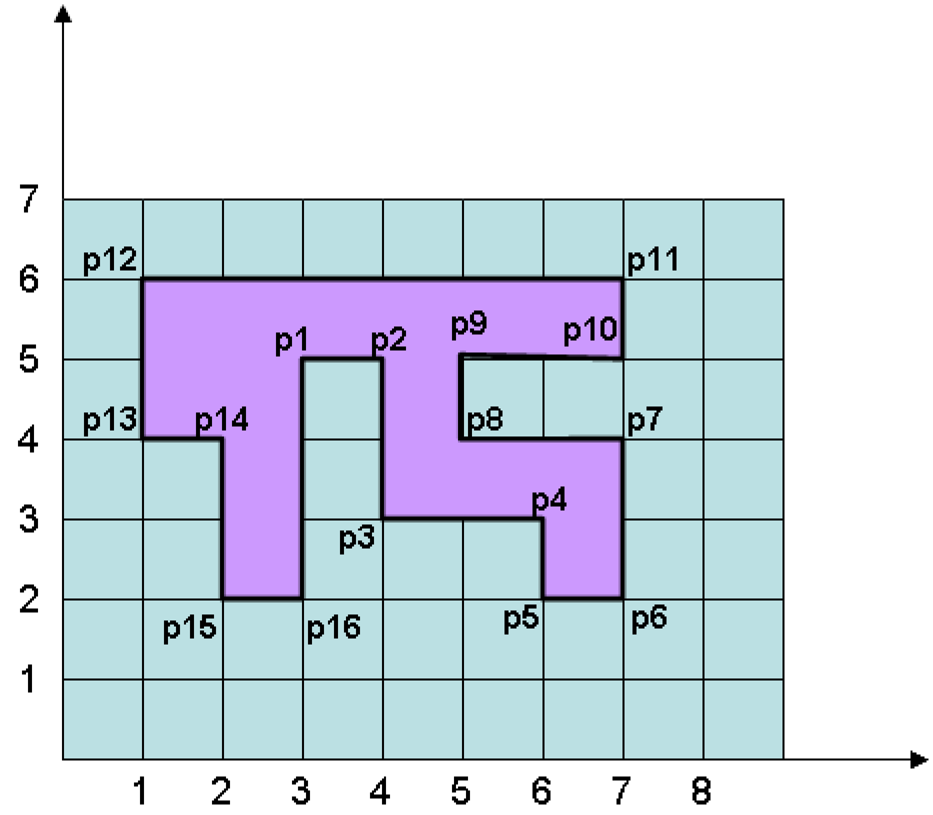
Sample Input and Corresponding Sample Output

Sample Input	Sample Output
1 3 4 5	6
2 1 3 6 10	15
4 170 405 798 1400 2268 3465	5060
5 1764 4160 8400 15300 25872 41344 63180	93100
4 1 16 81 256 625 1296	2401

P2 Calculate This!

You are given a planar polygon formed of horizontal and vertical lines and you must determine the total area of the enclosed polygon. The polygon does not intersect itself and there are no “holes” in the polygon. The (x,y) coordinates of each of its endpoints is an integer value > 0. One of the points on the polygon is identified as the starting point p1; in counter-clockwise order around the polygon, each successive point appears until you return to the original point. You can assume that three consecutive points $\mathbf{p_i}$, $\mathbf{p_{i+1}}$, $\mathbf{p_{i+2}}$ in the polygon are non-colinear.

Consider the following Polygon:



The area of this polygon is 15.

Input Format

Your program will read from standard input a sequence of lines. The first line will contain a single positive integer by itself that represents the number of points N that define the polygon. Each of the next N lines contains two positive integers separated by a space, where the first value is the x-coordinate of the polygon point and the second value is the y-coordinate of the polygon point.

Output Format

Your program will write a single line containing a positive integer that is the enclosed area of the polygon.

Sample Input and Corresponding Sample Output

Sample Input	Sample Output
16 3 5	15

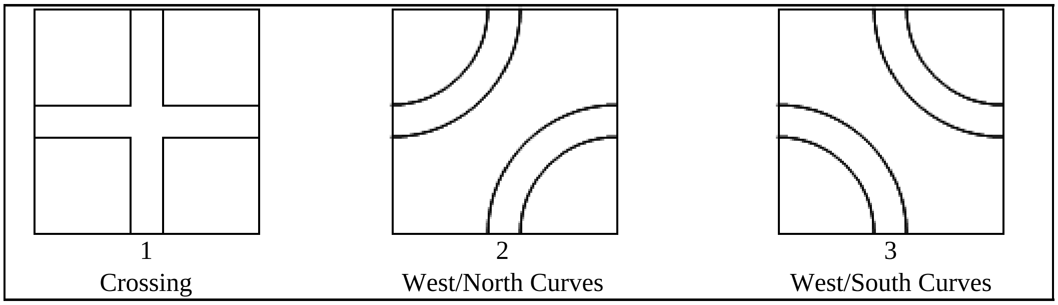
4 5
4 3
6 3
6 2
7 2
7 4
5 4
5 5
7 5
7 6
1 6
1 4
2 4
2 2
3 2	

4
2 2
6 2
6 6
2 6
	16

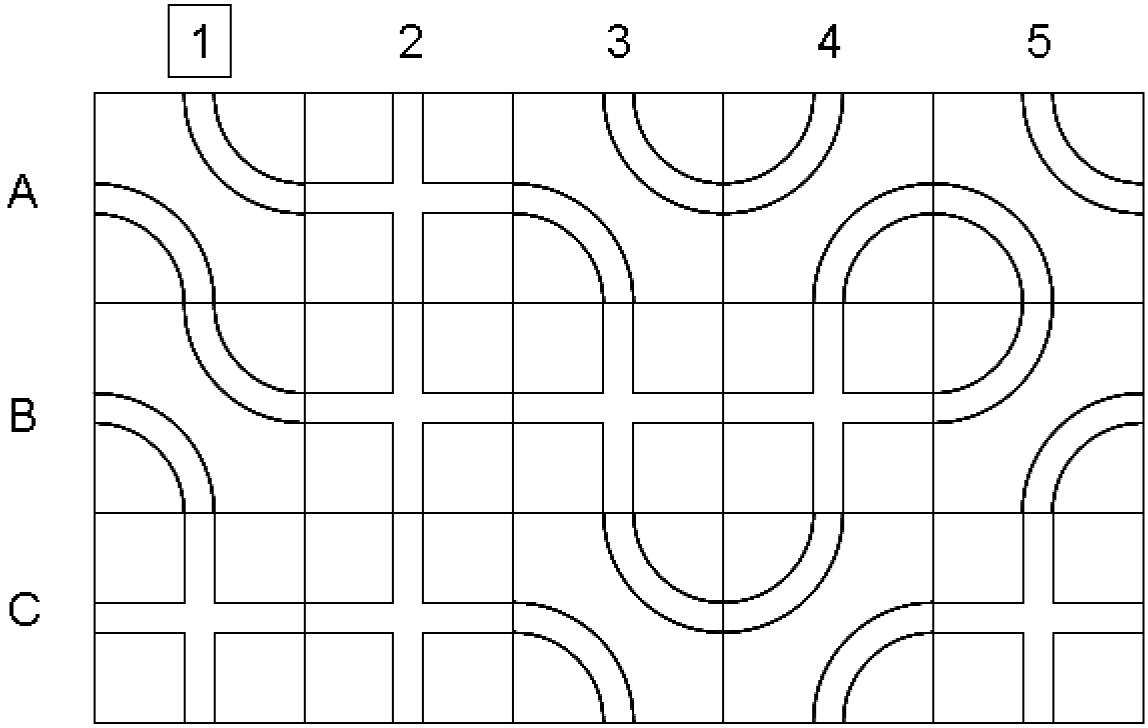
6
4 4
4 3
6 3
6 2
7 2
7 4
	4

P3 Locate This!

You are given a two-dimensional rectangular board consisting of N columns (labeled 1, 2, … N) and M rows (labeled A, B, C and so on) that contain an arrangement of three types of board pieces. You can assume M and N are both less than or equal to 10.



A board consists of an arrangement of these pieces (there are no gaps). For example, the following is a valid board arrangement. Note that the columns are labeled left to right while the rows are labeled from top to bottom. Each square has a label, thus the upper left square in the board is “A1”.



A train will enter the board at square A1 from the top and proceed through the board following the track as determine by the individual pieces until the train exits the board. **Your task is to output on a single line the sequence of squares in the order that the train encounters them**. In the above board configuration, the train traverses the following path: **A1 A2 A3 B3 C3 C4 B4 A4 A5 B5 B4 B3 B2 B1 A1**

Input Format

Your program will read from standard input a sequence of lines. The first line will contain a single positive integer by itself that represents the Number of Columns (let’s call this C). The second line will contain a single positive integer by itself that represents the Number of Rows (let’s call this R). The rest of the input will consist of R lines of strings of length C that contains digits 1, 2, or 3 representing the three possible board pieces described earlier. The first line of input is row A with the digits reading left-to-right as shown in the board; the second line of input is row B, and so on.

You can assume that the input is properly formatted

Output Format

Your program shall output to standard output on a single line by itself a sequence of squares (represented by Row and Column) separated by spaces. **Note that there must not be a trailing space in your output!**

Sample Input and Corresponding Sample Output

Sample Input	Sample Output
5 3 31323 31112 11321	A1 A2 A3 B3 C3 C4 B4 A4 A5 B5 B4 B3 B2 B1 A1
2 2 13 22	A1 B1
2 3 13 11 13	A1 B1 C1

P4 Manipulate This!

You are given the task of decoding a string whose input has been processed according to a series of directives. A string consists of a sequence of *n* characters (only uppercase letters of the alphabet and spaces). Index position 1 of the string is its first character while the last character of the string has index position *n*.

There are five directives:

1. `REVERSE a b` – reverse the order of characters from index position *a* to *b*. *a* and *b* are positive integers (where *a* < *b*) whose value £ *n* where *n* is the length of the string being processed.
2. `SWAP a b` – swap the two characters at index positions *a* and *b*. *a* and *b* are different positive integers whose value £ *n* where *n* is the length of the string being processed.
3. `INSERT str1 a` – insert string str1 so it appears just before the character at index position *a* and just after the character at position *a-1;* you can be assured that *0* < *a* £ *n* where *n* is the length of the string being processed.
4. `SUBSTITUTE str1 str2` – replace the first occurrence of substring str1 (when reading the string from left to right) with the string str2; if str1 does not exist as a substring then no change occurs. Note that substitution is CASE SENSITIVE and neither str1 or str2 can contain a space.
5. `DELETE a b` – delete the substring starting at index position *a* and completing at index position *b*; *a* and *b* are positive integers (where *a* < *b*) whose value £ *n* where *n* is the length of the string being processed.

Your program will read in a string and a set of properly formed directives (you can assume all values of *m* and *n* are valid). The output of your program is the string after all directives have been processed.

Input Format

Your program will read from standard input. The first line will contain a string of characters (uppercase letters of the alphabet and spaces) on a line by itself (there will be no trailing spaces in the input). The second line will be a single positive integer by itself that represents the number of directives *k* to be processed. The remaining input consists of *k* lines, each of which contains a valid directive. Each directive is formatted as “OPERATION param1 param2” where operation is either {REVERSE, SWAP, INSERT, SUBSTITUTE, or DELETE} and the two parameters param1 and param2 have appropriate values based upon the directive. Note that there is a single space separating the operation from param1, and param1 from param2.

Output Format

Your program must write to standard output a string on a line by itself representing the final value of the string after all directives have been processed.

Sample Input and Corresponding Sample Output

Sample Input	Sample Output
UNITED STATES OF AMERICA 6 REVERSE 6 13 SWAP 2 7 SWAP 16 1 INSERT NOT 5 DELETE 10 18 SUBSTITUTE ITN R	FEROTESU AMERICA
AIRPLANE 3 REVERSE 2 6 SWAP 6 2 DELETE 3 4	AIRANE
TIME FLIES 3	TTF EMELUNIMIES

```
SUBSTITUTE I MINUTE
REVERSE 2 12
SWAP 8 2
```