Vector Calculator

time limit per test: 2 seconds
memory limit per test: 256 megabytes
input: standard input
output: standard output

points: 100 · problem code: vectorcalculator · author: caneToadCoad

Vector Calculator

Listen up soldier! You are going into the front lines, maggot! I don't wanna hear about "integers" or "real numbers anymore". Drop and give me twenty, and then implement a calculator that can calculate 2d vector operations!!!!!!.

Input format

The first line has one integer, n, the number of queries.

n lines follow, each with a query.

There are 3 query types with vector outputs.

Add queries are in format "$ADD\ a_x\ a_y\ b_x\ b_y$". The answer to this query is the sum of vectors (a_x, a_y) and (b_x, b_y)

Multiplication queries are in format "$MULT\ a_x\ a_y\ b$". The answer to this query is the multiplication of vector (a_x, a_y) and the scalar b

Rotation queries are in format "$ROT\ a_x\ a_y\ ang$". The answer to this query is the resulting vector after rotating vector (a_x, a_y) around the origin by ang degrees counterclockwise

There are 3 queries with scalar outputs.

Triangle queries are in format "$TRI\ a_x\ a_y\ b_x\ b_y\ c_x\ c_y$". The answer to this query is area of the triangle with points (a_x, a_y), (b_x, b_y) and (c_x, c_y)

Dot product queries are in format "$DOT\ a_x\ a_y\ b_x\ b_y$". The answer to this query is the dot product of vectors (a_x, a_y) and (b_x, b_y)

Cross product queries are in format "$CROSS\ a_x\ a_y\ b_x\ b_y$". The answer to this query is the cross product of vectors (a_x, a_y) and (b_x, b_y)

$1 < n < 1e5$

All numbers in queries are in range $[-1e5, 1e5]$

Output format

For each query, output the answer on a separate line. For vector outputs, print out the answer in format "$x\ y$", where x is the horizontal component, and y is the vertical component For scalar outputs, simply print out the answer in a separate line.

Tip: be careful of the print format, we don't the program to print in scientific form, or to be skimping out on decimal places. For c++ users, $cout << setPrecision(10)$, imported from the iomanip library can resolve this.

Answers within 1e-5 of the reference answer will be accepted.

Examples

Input

```
3
ADD 3.5 4.5 6 -3
ROT 1 4.5 90
MULT -3892 3 0.5
```

Output

```
9.5 1.5
-4.5 1
-1946 1.5
```

Input

```
2
TRI 0 0 1 0 0 1
DOT 3000 4000 4000 -3000
```

Output

```
0.5
0
```

Input

```
1
CROSS -3.5 -6.5 7 13
```

Output

```
0
```