Scavenger Hunt

You are playing a scavenger hunt game organized by your school.

Assume the school is an infinite 2D field. The rules of the game are quite simple: There are *n* people from the organization, numbered from *1* to *n*. Person *i* is located at position (x_i, y_i). All you have to do is go to (x_i, y_i) and let the *i*-th person see you, for all $1 \le i \le n$. You can run from a position (x_a, y_a) to a position (x_b, y_b) in $dist((x_a,y_a),(x_b,y_b))$ seconds, where *dist(P,Q)* is the euclidian distance between points *P* and *Q*.

Initially, you are at the same position of person *1*, who already saw you. Also, initially you only know the position of person *1*. To find out the position of some other person, you need to ask someone. Every person knows the position of all the others. However, if you ask person *i* where person *j* is, it will take him $W_{i,j}$ seconds to answer. Your task is to find a sequence of questions and runnings that let you finish the game quickly. This is a partial problem - you do not need to find an optimal solution, but your score will be proportional to the solution you find.

Input/Output

This is an iterative problem. Your program will "talk" to the judge directly via the standard input and output.

There are no more than 20 test cases. For each test case, the judge will start by printing

START n

$\mathbf{x_1\ y_1}$

$\mathbf{W_{1,1}\ W_{1,2} \ldots\ W_{1,n}}$

$\mathbf{W_{2,1}\ W_{2,2} \ldots\ W_{2,n}}$

...

$\mathbf{W_{n,1}\ W_{n,2} \ldots\ W_{n,n}}$

(Assume $2 \le n \le 40$ and $0 \le x_i, y_i, W_{i,j} \le 10^4$).

After reading that, your program should print commands. A command must be one of these:

- **ASK i** ask the person that saw you last where the person *i* is. The judge will reply by printing **ANSWER** $\mathbf{x_i\ y_i}$. You'll get "Runtime Error" if $(1 \le i \le n)$ doesn't hold.
- **GO i** go to the position of the person *i* and let him see you. The judge will reply by printing **MOVED** . You'll get "Runtime Error" if $(1 \le i \le n)$ doesn't hold or if you don't know the position of the person *i* yet.
- **FINISH** finish the game. The judge will reply by printing **OK length** , where **length** is the total time spent by you in the game of the test case in seconds, with exactly 3 fractional digits (this is just a "bonus"; your program should be able to calculate **length** by itself). You'll get "Wrong Answer" if you haven't been to everyone's position at least once.

- You'll get "Runtime error" if your program prints something different from the commands described above.

Imediatally after printing **OK length** , the judge will start another test case by printing **START ...** . If there are no more test cases, the judge will print **END** instead. In this case, your program should terminate.

Please notice that you are allowed to ask more than one question or to not ask any questions at all to a person.

Attention: *the program should clear the output buffer after printing each line. It can be done using fflush(stdout) command or by setting the proper type of buffering at the beginning of the execution - setlinebuf(stdout).*

Also, while reading data from the judge in C/C++ using scanf, avoid hardcoding the commands you known you'll receive. For instance, prefer to use scanf("%s %d",start_string,&n) instead of scanf("START %d",&n).

Score

For each test case, $p_i = min(1.0, length/greedy)$, where *length* is the total time spent by you, and *greedy* is the length obtained by a program that prints ASK 2 GO 2 ASK 3 GO 3 ... ASK n GO n FINISH.

Your final score will be equal to $ceil[\ (\Sigma(p_i*n_i)/\Sigma n_i)*100\ ]$, where n_i is the value of *n* in the *i*-th test case. The *lower* your score is, the *better* you're ranked.

Exemple

This is an exemple of the executation of a program that obtains *greedy* as the total spend time (and thus gets the score 100):

Judge:

START 3

0 0

1 2 3

4 5 6

7 8 9

Program: ASK 2

Judge: ANSWER 2 2

Program: GO 2

Judge: MOVED

Program: ASK 3

Judge: ANSWER 1 1

Program: GO 3

Judge: MOVED

Program: FINISH

Judge:

OK 12.243

END

Program: <terminates>