6278 Blind Problem Solving

This is an interactive problem. You are to solve a subset sum problem, a variant of a knapsack problem. You are given n items, each weighing w_i grams, and a knapsack with the maximum capacity of c grams. You are to find a subset of these items whose total weight does not exceed c and is maximal possible.

Easy? But you have to do it blindly!

At the beginning you know only two numbers n and c - the number of items and the maximal capacity of a knapsack. There are positive weights w_i and two bit strings inside the jury program: A and B, each having n bits. Each bit string represents a candidate solution to the problem: if the i-th bit is set to 1, then the i-th item is a part of the solution. A stores your previously selected candidate solution. B is a candidate solution that is proposed to you on each turn by flipping a random bit in A. You can either accept or decline this proposal. Your task is to stop when B encodes the optimal solution.

Initially, A is initialized with some value that is fixed for each test case but is not known to you.

At each turn the value of A is copied to B, and then a bit at a uniformly random position of B is flipped.

You are given the weight of B which is equal to $\sum_{i=1}^{n} w_i \cdot B(i)$, where $B(i)$ is the value of i-th bit in B.

As a reply, your program is allowed to perform one of the following three actions:

- `stop` — this is the final action. This means that B encodes the optimal solution. Your program must exit after performing this action.
- `accept` — the value of B is copied to A.
- `decline` — the value of B is thrown out.

After each non-final action next turn starts. You are allowed to perform at most 1000 actions.

Interaction protocol

Interaction starts with your program reading three integer numbers — the values of n, c and the weight of B for the first turn from the first line of the standard input. Then your program must write its action to the standard output, wait for the jury program to write the weight of the new value of B to the standard input and so on.

Your program must exit after writing the last action (`stop`) to the standard output. Your program must write end-of-line sequence and flush the standard output after each action, including the last one.

It is guaranteed that a jury program chooses the bits to flip randomly, i. e. using a random number generation algorithm with an initial seed fixed for each test case.

Input

The input will contain several test cases, each of them as described below.

The first line of the standard input contains n ($1 \le n \le 20$), c ($0 \le c \le 10^9$) and the weight of B. It is guaranteed that $1 \le w_i \le 10^8$ for any i.

Each of the following lines will contain one integer number — the weight of B at the corresponding turn.

Output

For each test case, the output must follow the description below.

The standard output consists of the actions your program performs. Each action is represented with a single line containing a single word: stop, accept, or decline.

Sample Input

```
2 5 3
2
5
```

Sample Output

```
decline
accept
stop
```