5867 Finding Feasible Paths

You are all excellent programmers and should be good at tracing program running behaviors for debugging purposes. The following problem is to request you to verify if the program runs in the feasible ways.

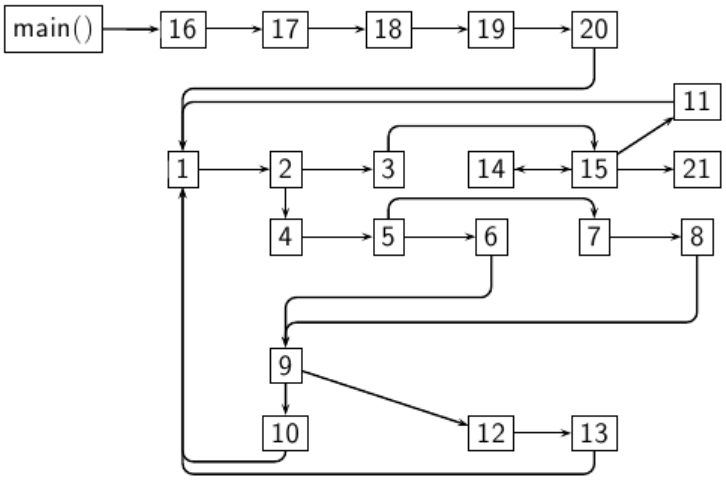
A program execution with function invocation can be modeled as a control flow graph. The control flow can be identified as an intra-function flow or inter-function flow. The inter-function flows will occur at function invocation and function return. For example, if we trace the following function `simpleRecFunc(x,y,z)` written in the syntax of the C programming language:

```
1: void simpleRecFunc(int a,int b, int *v) {
2:    if (a <=2) {
3:          *v = b + 1;
4:    } else {
5:        if (a>b)
6:          *v = *v+1 ;
7:        else
8:          *v = *v -1 ;
9:        if (a < b + 1) {
10:               simpleRecFunc(a-1, b, v);
11:               simpleRecFunc(a-2,*v,v);
12:       } else
13:               simpleRecFunc(a-3,*v,v);
14:    }
15: }
16: void main() { srandom(getpid());//set a random seed
17:   int x =random()%21;//obtain the x value between 0 and 20
18:   int y =random()%101;//obtain the y value between 0 and 100
19:   int z;
20:   simpleRecFunc(x,y,&z);
21: }
```

Each statement is annotated with a line number n, where n is a positive integer. After a function invocation, the return line number is $n + 1$. An exception is that after a function invocation at line 11, the return line number is 14. The formal parameter `v` in the `simpleRecFunc()` function is always passed in a valid integer reference, containing an integer value.

For example, if at line 2, the condition $(a \leq 2)$ is true, the run-time flow path is 2 3 15; if the above condition is false, and at line 5, the condition $(a > b)$ is true, the run-time path is 2 4 5 6 9 12 13. We won't trace the control flow and step into the standard library function of `srandom()`, `getpid()`, and `random()`, but only trace the flow into the function of `simpleRecFunc()`. According to the above program, we can depict a corresponding flow graph as follows:

Each node in the flow graph is labeled with the corresponding statement line number. The run-time path will be varied, due to different parameters of `a`, `b`, and `v`.



Given a path from node 16 to node 21, you are to determine if the path is feasible. For example, the path 16 17 18 19 20 1 2 3 15 21 is feasible. The path 16 17 18 19 20 1 2 4 5 7 8 9 12 13 1 2 3 15 21 is infeasible for we cannot find any possible integers or zero for `a`, `b` and `*v` to execute the above path. The path 16 17 18 19 20 1 2 4 5 7 8 9 10 1 2 3 15 21 is also infeasible because after the function invocation, the control must return to the next statement of the caller. The caller here being at line 10, the next statement line number is 11. Please note that there is no direct edge between 10 and 11 due to that we step into the function of `simpleRecFunc()` and the next node of 10 is 1 instead of 11. Assume the program is given enough memory to execute.

Technical Specification

1. The number of path data is N , where $0 < N \leq 1024$.
2. The number of statement line numbers in an path is M, where $0 < M \leq 2^{16}$.
3. In the invocation of `simpleRecFunc(x,y,z)`, $0 \leq x \leq 20$ and $0 \leq y \leq 100$

Input

The test data begins with a positive integer N, where N is the number of path data. The subsequence N lines are the input paths, specified by a sequence of statement line numbers, each separated with one or more spaces, from the starting statement line 16 to the end statement line 21. Each input path is read up to the end of the line, for example:

```
16 17 18 19 20 1 2 3 15 21
```

Output

If the given path is feasible according to the above control flow graph, output feasible, otherwise output infeasible.

Sample Input

```
4
16 17 18 19 20 1 2 3 15 21
16 17 18 19 20 1 2 4 5 7 8 9 12 13 1 2 3 15 21
16 17 18 19 20 1 2 4 5 7 8 9 10 1 2 3 15 21
16 17 18 19 20 1 2 4 5 6 9 12 13 1 2 3 15 14 15 21
```

Sample Output

```
feasible
infeasible
infeasible
feasible
```