P9

In the movie "Die Hard 3", Bruce Willis and Samuel L. Jackson were confronted with the following puzzle. They were given a 3-gallon jug and a 5-gallon jug and were asked to fill the 5-gallon jug with exactly 4 gallons. This problem generalizes that puzzle.

You have two jugs, A and B, and an infinite supply of water. There are three types of actions that you can use: (1) you can fill a jug, (2) you can empty a jug, and (3) you can pour from one jug to the other. Pouring from one jug to the other stops when the first jug is empty or the second jug is full, whichever comes first. For example, if A has 5 gallons and B has 6 gallons and a capacity of 8, then pouring from A to B leaves B full and 3 gallons in A.

A problem is given by a triple (Ca, Cb, N), where Ca and Cb are the capacities of the jugs A and B, respectively, and N is the goal. A solution is a sequence of steps that leaves exactly N gallons in jug B. The possible steps are

```
fill A
fill B
empty A
empty B
pour A B
pour B A
success
```

where "`pour` A B" means "pour the contents of jug A into jug B", and "`success`" means that the goal has been accomplished.

You may assume that the input you are given does have a solution.

Input

Input to your program consists of a series of input lines each defining one puzzle. Input for each puzzle is a single line of three positive integers: Ca, Cb, and N. Ca and Cb are the capacities of jugs A and B, and N is the goal. You can assume $0 < Ca \leq Cb$ and $N \leq Cb \leq 1000$ and that Ca and Cb are relatively prime to one another.

Output

Output from your program will consist of a series of instructions from the list of the potential output lines which will result in either of the jugs containing exactly N gallons of water. The last line of output for each puzzle should be the line '`success`'. Output lines start in column 1 and there should be no empty lines nor any trailing spaces.

Sample Input

```
3 5 4
5 7 3
```

Sample Output

```
fill B
pour B A
empty A
```

```
pour B A
fill B
pour B A
success
fill A
pour A B
fill A
pour A B
empty B
pour A B
success
```

P10

Running a taxi station is not all that simple. Apart from the obvious demand for a centralised coordination of the cabs in order to pick up the customers calling to get a cab as soon as possible, there is also a need to schedule all the taxi rides which have been booked in advance. Given a list of all booked taxi rides for the next day, you want to minimise the number of cabs needed to carry out all of the rides.

For the sake of simplicity, we model a city as a rectangular grid. An address in the city is denoted by two integers: the street and avenue number. The time needed to get from the address a, b to c, d by taxi is $|a - c| + |b - d|$ minutes. A cab may carry out a booked ride if it is its first ride of the day, or if it can get to the source address of the new ride from its latest, at least one minute before the new ride's scheduled departure. Note that some rides may end after midnight.

Input

On the first line of the input is a single positive integer N, telling the number of test scenarios to follow. Each scenario begins with a line containing an integer M, $0 < M < 500$, being the number of booked taxi rides. The following M lines contain the rides. Each ride is described by a departure time on the format $hh : mm$ (ranging from `00:00` to `23:59`), two integers a b that are the coordinates of the source address and two integers c d that are the coordinates of the destination address. All coordinates are at least 0 and strictly smaller than 200. The booked rides in each scenario are sorted in order of increasing departure time.

Output

For each scenario, output one line containing the minimum number of cabs required to carry out all the booked taxi rides.

Sample Input

```
2
2
08:00 10 11 9 16
08:07 9 16 10 11
2
08:00 10 11 9 16
08:06 9 16 10 11
```

Sample Output

```
1
2
```