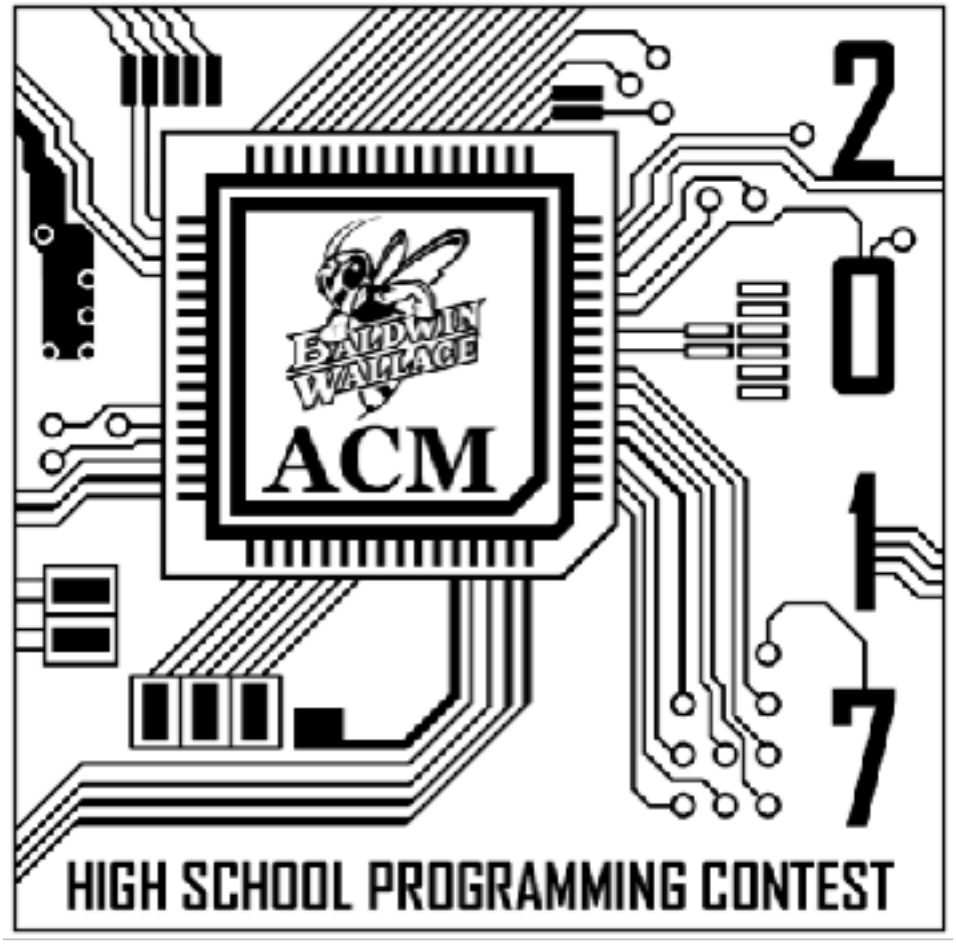
BALDWIN WALLACE UNIVERSITY

2017 HIGH SCHOOL PROGRAMMING CONTEST



DO NOT OPEN UNTIL INSTRUCTED TO DO SO

Why Did the Minions Cross the Road?

The Penguin is trying to cross the street with all of his penguin minions in order to rob the bank. Although The Penguin is able to fly, thanks to his umbrella, he is not able to carry any of his minions with him. All of his minions have to cross the street by foot. Each lane in the street has a stream of traffic with a vehicle arriving at some regular interval (e.g., every 5 seconds). Initially, the Penguin and minions are standing on the curb on the left side of the street and seeing a vehicle in very lane. The Penguin begins to give the minions directions, one per second. The directions tell the minions to stay where they are, move forward one lane, or move back one lane. The command is completed in time for the next command to be heard. Due to the speed of the vehicles, the minions cannot react fast enough in order to dodge a vehicle that arrives in the same lane they do at the same time, so they unfortunately get hit if this happens. The minions are very trusting and follow the instructions from The Penguin without hesitation. If they are lucky, they make it across the street, but they could also get hit by a car, get stranded in a lane if there aren't enough commands, or return to the starting spot. If at any point the minions get to the other side of the street, they ignore any additional commands the Penguin might give them. Also if the minions are standing before the first lane, then any backwards commands will result in the minion staying. The minions can not move away from the street.

Details of the Input:
The first line of input consists of a single positive integer, *c*, representing the number of cases to be processed. For each case, there are three lines of input. The first line contains an integer, *l*, representing the number of lanes in the street. The next contains *l* integers, separated by spaces, representing the frequency of traffic (in seconds) for each lane. The final line contains a string comprised of some combination of the letters 'F','B', and 'S' which represent commands of forward, back, or stay. The commands are to be followed in the given order.

Details of the output:
For each case, one line of output is generated. The line begins with a label of the form `Case i:` (*i*=1…*c*) and then prints what happened to the minion. Options are `Crossed the street`, `Is stranded`, `Was hit`, or `Did not move` (if they end up back in the starting spot).

Sample input:

```
4
6
2 3 4 5 6 7
FFFFFFF
6
7 6 5 4 3 2
FFFFBBBB
6
2 3 4 5 6 7
FFFBBB
4
5 5 5 5
FSSF
```

Sample out

```
Case 1: Crossed the street
Case 2: Was hit
Case 3: Did not move
Case 4: Is stranded
```

No Joking Around

The Joker wants to pass secret messages about his devious plans, so he concocts a scheme to encrypt his messages as follows. Each original message is a string of digits (0-9). The encrypted string is formed by "verbally" describing the structure of the message. As an example, the string 122344111 can be described as "one 1, two 2's, one 3, two 4's, three 1's". Therefore, the encrypted message is 1122132431. Turns out, that the joke is on the Joker in this situation as it is generally not possible to uniquely identify the original message from its encrypted form. As an example, a string comprised of 112213243 1's also encrypts as 1122132431.

Details of the input

The first line of input consists of a single positive integer, *c*, which is the number of test cases to be processed. For each case, there is a single line of input containing up to 1000 digits.

Details of the output

For each test case a single line of output is printed. The output starts with the label "`Case i:` " (*i* = 1… *c*) and ends with the encrypted message.

Sample Input

```
3
122344111
1111111111
12345
```

Sample Output

```
Case 1: 1122132431
Case 2: 101
Case 3: 1112131415
```

Mad Hatter Goes to School

The Mad Hatter has enslaved everyone at the Gotham High School. All of the students and staff are under the Mad Hatter's mind control. Everyone is trying to solve simple math problems, but the Mad Hatter has forced them to forget one specific digit. While Batman eventually removes the chip enslaving everyone, he must first recite a fraction with the smallest denominator that doesn't use the number they forgot. For example, if Batman needs to represent "15" but cannot use a "5", then the answer would be "30/2". This is now acceptable as neither the numerator nor denominator contains "5". If the original number doesn't contain the bad digit, then it still needs to be represented as a fraction with a "1" in the denominator.

Details of the input

The first line of the input will be a positive integer c, indicating the number of test cases to be processed. For each case, there will be two integers, n ($1 \leq n \leq 99$) which is the number that must be represented, and d ($0 \leq d \leq 9$) which is the digit that is not allowed to appear in either the numerator or denominator of the output.

Details of the output

For each case there is a single line of output. The line begins with a label of the form `Case i:` ($i = 1 \ldots c$) and then displays the numerator, followed by a forward-slash "/" and then the denominator.

Sample input

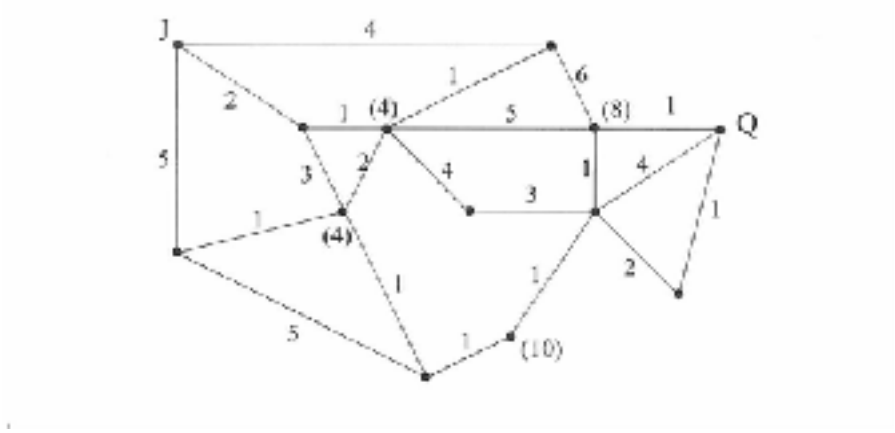
```
3
20 2
15 5
16 5
```

Sample output

```
Case 1: 60/3
Case 2: 30/2
Case 3: 16/1
```


Batman vs. Croc

Killer Croc made off from the Gotham Bank with a sack of gold bars and hid them in the connected network of sewers below the city. Batman has a map of the sewers and knows the travel time along each stretch of sewer pipe, so is going to try to retrieve the gold bars. However, Killer Croc, expecting that Batman would try to play hero again, placed bombs at some of the intersections of the tunnels. Batman was able to do a satellite scan of the sewer system and discovered exactly where the gold is hidden, where each bomb is located and at what point it will explode. Once a bomb explodes, that intersection is destroyed and is henceforth impassable. It goes without saying that Batman does NOT want to be at an intersection when a bomb explodes. But because of his finely honed athletic abilities, he will escape unscathed if he misses the explosion by even a second. Batman wants to get to the gold in the least amount of time possible. Your job is to find that minimal time.



For example, in the following situation, Batman (starting at the intersection marked J) can get to the gold (marked Q) in a minimum of 13 seconds. The integers on each sewer line indicate travel time and the integers at some nodes (surrounded by parentheses) indicate the time a bomb goes off at that node. Batman starts running at time 0 and all timing on the bombs is relative to time 0.

Details of the input

The first line of input contains a single integer, c, which is the number of test cases to be processed. For each case, its first line of input contains a single integer, n ($n \leq 40$). There are $n+1$ nodes in the input set numbered 0,1,…,n. Batman (J) is located at node 0 and the gold (Q) is located at node n. The next line has two integers, p and b ($0 \leq b \leq 10$), indicating the number of sewer pipes and bombs, respectively. The next p lines each contain three integers, n_1, n_2, and t, indicating that there is a pipe from node n_1 to node n_2 that takes time t to travel. The next b lines will contain two integers, N and T, indicating there is a bomb at node N that is set to go off at time T. There will be at most one bomb at any intersection.

Details of the output

For each input set, output a line of the form

```
Batman can reach the gold in time t.
```

where *t* is the minimal time required, if Batman can reach the gold at all. If Batman cannot reach the gold, output

```
Batman cannot reach the gold.
```

Sample input

```
1
12
20 4
0 1 4
0 2 2
3 0 5
2 4 1
2 5 3
5 4 2
1 4 1
1 6 6
4 6 5
4 7 4
7 8 3
9 5 1
3 9 5
6 8 1
9 10 1
10 8 1
8 11 2
11 12 1
8 12 4
6 12 1
5 4
4 4
10 10
6 8
```

Sample output

```
Batman can reach the gold in time 13.
```

Invasion of the Mutant Plants

For Posion Ivy's latest scheme to green up Gotham, she has created several varieties of mutant plants. The plants are growing throughout the streets of Gotham, creating havoc everywhere they spawn. All plants start to grow in the center of a street and branch out very systematically and symmetrically. When a branch grows to a certain length, it spawns two branches of half the size that grow away from the splitting point at a 90˚ angle. Each variety of plant has a minimum length of branch that will grow. In order for Batman to clean up the streets, he has to know what the plant would look like fully grown. The BatWing has a built in BatScanner which can detect the length of the starting branch (i.e., trunk) of a plant, but needs your help in generating a depiction of the fully grown plant.

Details of the input
The first line of input contains a single integer, *c*, which is the number of test cases to be processed. For each test case, there is a single line containing two integers, *l* and *m* ($0 < l,m < 40$), represents the length of the starting branch and the minimum length that a branch can be.

Details of the output
For each test case, the first line of output is of the form the form "`Case i:`" ($i = 1 \ldots c$). The remaining lines for the case depict a square grid comprised of spaces, vertical lines ('|' (shift back slash)) and horizontal lines ('-') with dimensions (2**l* +1) x (2**l* +1) where the vertical and horizontal lines represents the branches of plant in their respective direction.

Sample input
```
1
10 1
```

Sample output
Case 1:

```
 _ _           _ _
  |             |
  |             |
   _____ _____
  |     |     |
  |     |     |
 _ _    |    _ _
        |
        |
        |
        |
        |
        |
        |
```

Jewel Smuggling Dolls

Catwoman has created a new type of Russian doll that she can use to smuggle stolen jewels. It can hold a tremendous amount of weight in a small form. For her creation, she designed the dolls in a unique way. Specifically, for all dolls within the nesting, the ratio of the weights of a given doll and the next one down is the smallest prime factor of the weight of the larger. That continues down to the innermost doll, whose weight will always be one.

For example, if the outermost doll weighs 21 units, the next one down weighs 7 units (21 / 3), and the innermost doll weighs 1 unit (7/7).

Catwoman needs to know the weight of the entire nested doll (the outer doll as well as all dolls nested within it) to make sure she will be able to carry the weight. Given the weight of the outer doll, you are to determine the weight of the set of nested dolls.

Details of the input

The first line of input contains a positive integer, *c*, specifying the number of cases to be processed. Each of the remaining *c* lines contain a single positive integer *w* ($1 < w < 1{,}000{,}000$) representing the weight of the outermost doll for that case.

Details of the output

For each case, the output begins with the label `Case i:` followed by the total weight of the nested dolls.

Sample input

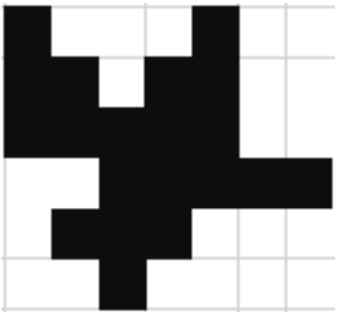
```
2
21
101
```

Sample output

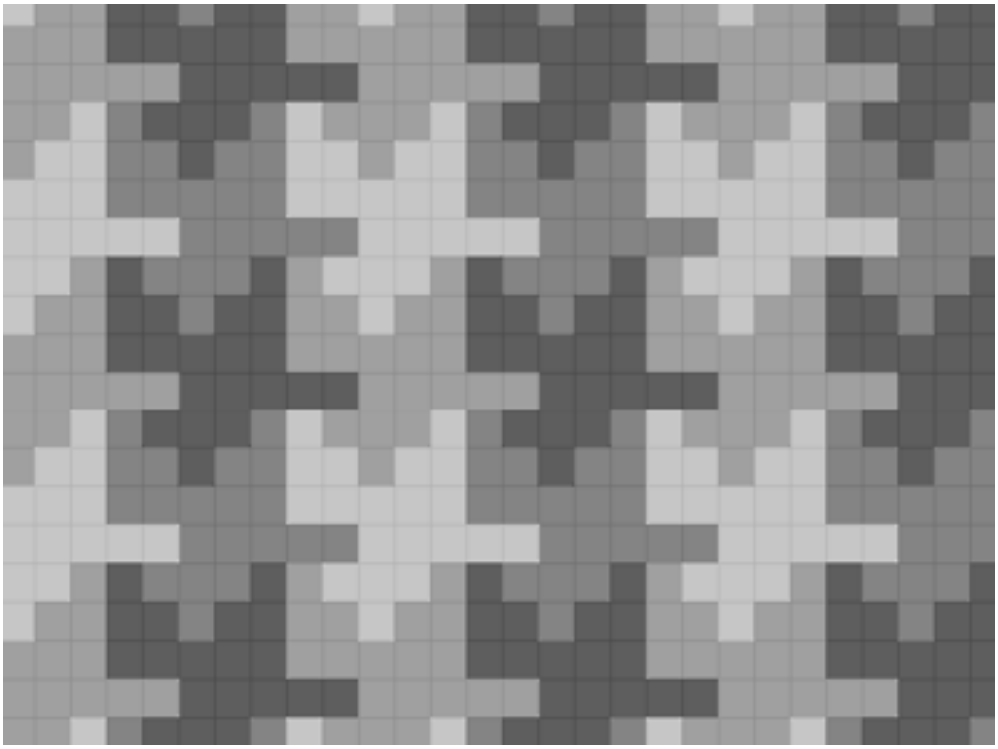
```
Case 1: 29
Case 2: 102
```

Small… No, Nano Puzzles

Lucius Fox has been working on a new nano technology at Wayne Enterprises. Each of the nano bots is shaped like a puzzle piece inside of a rectangular grid (top picture below). Before Lucius can release the new nano bots, he must figure out how to carefully assemble them because if the nano bots cannot fit together perfectly (that is, leaving no blank space anywhere as they interlock as depicted in the bottom picture), they will fall apart and not be able to function. The nano bots cannot be rotated or reflected (flipped) and they must be able to align both vertically and horizontally (but not diagonally) with adjacent bots. Lucius needs to finalize the design for the new nano bots before he can tell Bruce Wayne that the job is complete. Lucius Fox has come to you with several different possible designs, and you must figure out if the designs will work or not.



.



Details of the input

The first line of the input will be a positive integer *c*, indicating the number of test cases. Then for each case, the first line will contain two integers *R* and *C* ($2 \le R \le 20$, $2 \le C \le 20$) where *R* is the height and *C* is the width of the rectangle containing an individual piece. The rest of the input for each case will be *R* lines of *C* integers that are either 0 or 1. 0 represents a blank square and 1 represents a solid square of the piece. The first sample input below describes the example pictured above.

Details of the output

For each case, one line of output is generated. The line begins with a label of the form `Case i:` (*i*=1…*c*) and then prints `yes` if the pieces can fit together or `no` otherwise.

Sample input

```
2
7 6
1 0 0 0 1 0 0
1 1 0 1 1 0 0
1 1 1 1 1 0 0
0 0 1 1 1 1 1
0 1 1 1 0 0 0
0 0 1 0 0 0 0
3 2
0 1 0
1 1 1
```

Sample output

```
Case 1: yes
Case 2: no
```