23rd Annual Denison Spring Programming Contest
Granville, Ohio
25 February, 2012

Rules:

1. There are **six** questions to be completed in **four hours**.
2. All questions require you to read the test data from standard input and write results to standard output. You cannot use files for input or output. Additional input and output specifications can be found in the General Information Sheet.
3. The allowed programming languages are C, C++ and Java.
4. All programs will be re-compiled prior to testing with the judges' data.
5. Non-standard libraries cannot be used in your solutions. The Standard Template Library (STL) and C++ string libraries are allowed. The standard Java API is available, except for those packages that are deemed dangerous by contestant officials (e.g., that might generate a security violation).
6. The input to all problems will consist of multiple test cases unless otherwise noted.
7. Programming style is not considered in this contest. You are free to code in whatever style you prefer. Documentation is not required.
8. All communication with the judges will be handled by the PC^2 environment.
9. Judges' decisions are to be considered final. No cheating will be tolerated.

Problem A: Lyndon Words

Consider the word w = `abacd`. If we look at all the words obtained by rotating w we get the set {`bacda`, `acdab`, `cdaba`, `dabac`, `abacd`}. (We count w itself as one of the rotations.) Note that w is lexicographically first in this set. We call w a *Lyndon word* — more precisely, a 4-Lyndon word of length 5, since the alphabet {`a`, `b`, `c`, `d`} has four letters. There is another proviso: w cannot be the same as a non-trivial rotation of itself. So, neither `aabaab` nor `aaaaaa` would be considered Lyndon words of length 6.

A *binary Lyndon word* of length n is a word with n letters, each letter being either a 0 or 1 that is minimal in lexicographic order in the set of all its rotations (provided the word is not the same as a non-trivial rotation of itself). For example, `00101` and `010111` are binary Lyndon words of length 5. Indeed, there are 6 binary Lyndon words of length 5. You are to find how many binary Lyndon words there are of various lengths.

Input

Input for each test case will consist of one line containing a single positive integer n ($n < 20$). A line with 0 will follow the final test case.

Output

Each test case should generate one line of output, in the format given below, giving the number of possible binary Lyndon words of length n.

Sample Input

```
5
2
4
10
0
```

Sample Output

```
Case 1: 6
Case 2: 1
Case 3: 3
Case 4: 99
```

Problem B: Got No Title For This One

There are a few algorithms on $n \times n$ matrices that first do some manipulation on the entries of the matrix, then check to see if there is a choice of n entries so that each of the n entries is 0 and each row and column has one of these chosen entries. (Once this condition is detected, the algorithm usually moves to its next phase.)

For example the matrix $\begin{bmatrix} 0 & 1 & 0 & 4 & 1 \\ 1 & 0 & 2 & 1 & 0 \\ 0 & 2 & 3 & 4 & 0 \\ 1 & 3 & 2 & 0 & 1 \\ 0 & 1 & 1 & 0 & 3 \end{bmatrix}$ has this property while $\begin{bmatrix} 0 & 1 & 0 & 4 & 1 \\ 1 & 2 & 0 & 1 & 1 \\ 1 & 2 & 3 & 0 & 0 \\ 0 & 3 & 0 & 1 & 1 \\ 1 & 0 & 1 & 0 & 3 \end{bmatrix}$ does not.

You can see that the first matrix has the property since we can pick entries $(1, 3)$, $(2, 2)$, $(3, 5)$, $(4, 4)$, and $(5, 1)$; that is, 5 entries that are zero, in 5 different rows and 5 different columns. Try as you might, it can't be done for the second matrix. Of course, some matrices might have the property in multiple ways – we are only interested if it can be done at all, not how many times.

Input

Input for each test case will be on multiple lines. The first line of input will contain a positive integer $n \leq 20$. Following this will be n rows, each containing n integers, indicating the n rows of the matrix. A line with 0 follows the input for the last test case.

Output

Each test case should produce one line of output in the format below, indicating if the matrix has the property (`YES`) or not (`NO`).

Sample Input

```
5
0 1 0 4 1
1 0 2 1 0
0 2 3 4 0
1 3 2 0 1
0 1 1 0 3
5
0 1 0 4 1
1 2 0 1 1
1 2 3 0 0
0 3 0 1 1
1 0 1 0 3
0
```

Sample Output

```
Case 1: YES
Case 2: NO
```

Problem C: The Driveway Game

Emma lives in a large home with one of those big semicircular drives in the front. Emma likes to throw parties. When the guests arrive, they pull into the drive from the right. Unfortunately the drive is only one car wide. So, if a guest needs to leave early and is blocked, others must move their cars. The guests are all good-humored about it and in fact play a game where those that pull out of the drive, return and repark in the order they left the drive using the same end of the drive they exited.

For example, suppose the drive initially had the cars ABCDEFGH and D leaves. Once A, B, and C pull out and return the drive looks like CBAEFGH. If F leaves, G and H would have to back out on the right, since F is closer to the right end of the line of remaining cars. If a car finds itself in the middle, the cars on the left always pull out. (See the first sample test case.)

You are to follow this game and show the drive after each exchange. Assume Emma's drive can hold 26 cars.

Input

Input for each test case is contained on one line. The line starts with a positive integer n no greater than 26, indicating the number of cars initially in the drive. These are represented by the first n upper case letters. Following n is m ($1 \leq m < n$) indicating the number of cars that leave. Following this are m letters (separated by blank spaces) indicating the cars that leave in the order that they leave. You may assume that the cars leaving are indeed still in the driveway. A line with 0 follows the last test case.

Output

Output for each test starts with a line containing only an asterisk (`*`). Then there should be m lines of output showing the order of the cars after a car leaves and the other cars repark.

Sample Input

```
5 3 C A B
8 5 D F E B A
0
```

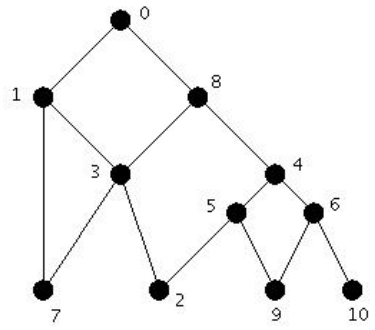
Sample Output

```
*
BADE
BDE
DE
*
CBAEFGH
CBAEHG
CBAGH
CAGH
CGH
```

Problem D: Draino

As a summer internship, you find yourself on an archeological dig of the ancient city of Binarium (tucked away in the mountains north of Coloradodium) had an intricate system of water reservoirs. Water from the snow melt would come together at one point and then there was constructed a series of canals. Canals would meander around and occasionally split into two parts with half the water flowing one direction and half the other. These locations we'll call *nodes*. (We'll call the reservoirs nodes also, although no water flows out of a reservoir.) More than one canal might meet at a node, but the water then would divide and continue on downhill – again split 50-50. Since water only flows downhill, the canals (and water) flow in just one direction. The ultimate destination of the water is a series of reservoirs. More than one canal might dump into a reservoir and no canal ever leaves a reservoir. As you excavate Binarium, you're wondering how the water is distributed to these reservoirs. That is, what portion of the water ends in each.

For example, consider the system below:



Note that water starts at node 0 and ends at reservoirs labeled 7, 2, 9, and 10. If we follow the water down the left-most canal path, we see that 1/2 the water flows between nodes 0 and 1, then 1/4 flows between node 1 and reservoir 7. Following all the paths down we see reservoirs 7, 2, 9, and 10 get 1/2, 5/16, 1/8 and 1/16 of the water, respectively. (We notice this sums to 1, of course.)

There are other cities up in these mountains from the same civilization, each with their own system of canals. Your job, as the only programmer on the crew, is to write a program to compute the portion of water flowing to each reservoir, given the canal configuration.

Input

Input for each test case will start with a line of the form $n\ m\ r_1\ \ldots\ r_m$ indicating there are $n+1$ nodes (labeled 0 thru n with node 0 being the topmost node), and the m reservoir nodes are labeled $r_1, \ldots, r_m$. The next $n+1-m$ lines will be of the form $k\ a\ b$ indicating node k splits with canals running down to nodes a and b. (The values for k will be all $0 \le k \le n$ that are not reservoirs.) You may assume $2 \le n \le 10$. A 0 will follow the last test case.

Output

Output for each test case should be in the form given below, giving the portion of water flowing into each reservoir. List the reservoirs in the order given in the the input (which may not be sorted nor consecutive). The values should be real numbers, formatted with no trailing zeros. You may assume water flows into all reservoirs, all the water flows into the reservoirs, and water flows in all canals.

(over)

Sample Input

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

Sample Output

```
7:0.5 2:0.3125 9:0.125 10:0.0625
2:0.5 1:0.5
```

Problem E: Semicircle the Wagons

Given some points on a circle, do they all lie on the same semicircle? Points will be labeled by their degree clockwise from north. So, the northern-most point is at location 0, eastern-most point is 90, and so on. Two points diametrically opposite each other are considered on the same semicircle.

Input

Input for each test case is on one line of the form $n\ p_1\ p_2 \ldots p_n$ where $1 \leq n \leq 100$ and $0 \leq p_i < 360$, all integers. The p_i indicate the positions of the n not necessarily distinct points on the circle. A 0 will follow the last test case.

Output

Each test case should produce one line of output, in the format below, saying either `YES` or `NO`, accordingly.

Sample Input

```
5 10 20 15 190 56
6 10 20 30 40 50 60
6 330 340 350 0 10 150
4 0 90 180 270
0
```

Sample Output

```
Case 1: YES
Case 2: YES
Case 3: YES
Case 4: NO
```

Problem F: I Could Go For A Cup Of Joe

Your boss, the regional manager of Starbuzz Coffee Houses, is going to open shops along the interstate. Research has done their work and estimated the weekly profits for potential stores at each intersection. They also know Starbuzz can't simply put a store at each intersection, else those potential profits quickly decrease. In fact, they've specified the minimum distance between any two Starbuzz stores.

For example, suppose there are 5 intersections on an east-west highway with profits (going east to west) of 3, 5, 6, 2, and 3 thousand dollars a week. The distance (east to west) between consecutive intersections is 2, 1, 1, and 4 miles. Also, let's assume that no two stores can be closer than 4 miles. Then the maximum profit would be 9 thousand dollars which is realized if building stores at the 3rd and 5th intersection. Of course, in some examples (but not the one just given) there may be more than one way to realize the maximum profit.

Your job is to figure out where to put the Starbuzz stores, or rather just what the maximum profit could be. So you fill your cup with the free office coffee (a perk for working at Starbuzz) and set to work.

Input

Input for each test case is on one line and is of the form $n\ m\ p_1\ d_1\ p_2\ d_2\ \cdots\ p_{n-1}\ d_{n-1}\ p_n$, indicating there are n intersections with profit p_i at intersection i and distance between intersection i and $i+1$ is d_i. Furthermore, the minimum distance between Starbuzz is m. Assume that $1 < n \leq 100$, $1 \leq p_i, d_i \leq 100$, and $1 \leq m \leq 1000$. A line containing 0 follows the last test case.

Output

For each test case out, in the format below, the maximum profit Starbuzz can realize.

Sample Input

```
5 4 3 2 5 1 6 1 2 4 3
5 4 3 2 5 1 6 1 4 4 3
0
```

Sample Output

```
Case 1: 9
Case 2: 10
```