# HKUST

# Programming Contest

## Oct 26th, 2002

6 problems, 3 hours
Running time limit: 10 seconds

## Problem Index

(This is a blank page)

# HKUST Programming Contest 2002

# Problem 1

## Expressions

The precedence and associativity of operators are very important in evaluating the values of expressions as they determines the "correct" order of different operations. For example, 1+2*3=7 and we know we will do multiplication first and then addition. However, we can change the order by adding some parentheses, say (1+2)*3=9. In this problem, you are given an expression and a target value *x*. You're asked to add some parentheses so that the value of the expression will be equal to *x*.

## Input

There are more that one test case and each test case will have 2 lines. First line consists of 2 integers *n* and *x* where *n* is the number of operators of the expression and *x* is the target value. Note that $1 \le n \le 7$ and $|x| \le 10000$. The second line of the input consists of an expression with n operators. '+', '-' and '*' are possible operators for the expressions and any operator must appear between 2 positive integers. There is no space between operators and operands. If *n* is equal to -1, it means your program should terminate.

## Output

Your program should add parentheses to the expressions so that the result of the expression will be equal to *x*. As it can have more than one solution for each test case. As long as your output is a valid mathematical expression, it will be accepted. But please keep in mind that you should not have newlines (`'\n'`) within an expression. If there is no method to parenthesize the expression to get *x*, print “not found”.

## Sample input

2 9
1+2*3

3 0
1+1-1*2
5 5
1+1+1+1+1+1
-1

## Sample output

(1+2)*3
(1+1)-(1*2)
not found

# HKUST Programming Contest 2002

# Problem 2

## ON and OFF

Lord Alfons is known for his crazy ideas how to surprise visitors to his old castle. Recently, he has built complicated locks for the entrance doors. A lock consists of $n$ switches whose position ON or OFF is visible from the outside.

The switches can only be operated according to the following rules:

1. Switch 1 can always be operated.
2. Switch 2 can only be operated if Switch 1 is ON.
3. Any other switch can only be operated if the switch with the next smaller number is ON and all other switches with smaller numbers are OFF.
4. Switch $n$ can also be operated if switches 1 to $n$-1 are OFF.

Lord Alfons has several such locks with different number of switches. Unfortunately, it often happens that visitors leave a lock in an untidy state because they fail to open the lock. Lord Alfons considers a lock to be tidy if all its switches are OFF. Therefore, every evening his butler must go through the castle and tidy up all locks. You should help him.

## Problem

Write a program that for any given $n$ and any given switch positions of a lock outputs the sequence of switches that must be operated to tidy up the lock (i.e., turn all switches OFF).

## Input

The input consists of a number of cases. Each line contains one case. A line will begin with an integer $n$ ($1 \leq n \leq 10$), the number of switches, followed by $n$ integer(s) from the set {0,1}, where 0 means “OFF” and 1 means “ON”. A value of $n$ = 0 indicates the end of the input.

## Output

For each test case you have to print the steps showing how to turn off all the switches. There can be more than one solution to a test case. Your solution does not need to be the shortest one, but the number of steps should not exceed 10000.

## Sample Input

```
4 1 1 1 1
10 0 0 1 1 0 1 0 1 0 1
0
```

## Sample Output

1 1 1 1
0 1 1 1
0 1 0 1
1 1 0 1
1 0 0 1
0 0 0 1
0 0 0 0
0 0 1 1 0 1 0 1 0 1
(etc…)
0 0 0 0 0 0 0 0 0 0

# HKUST Programming Contest 2002

# Problem 3

## Stuttering Subsequence

Recall that given two strings $A$ of length $m$ and $B$ of length $n$, where $m \geq n \geq 1$, we say that $B$ is a subsequence of $A$ if $A = a_1 \ldots a_m$, $B = b_1 \ldots b_n$, and there is an ascending sequence $i_1 < \ldots < i_n$ of positions in $A$ such that, for all $j$, $1 \leq j \leq n$, the symbol $a_{i_j}$ equals the symbol $b_j$.

Now, define $B^i$, for $i \geq 1$, to be $b_1^i \ \ldots \ b_n^i$; that is, $B^i$ is $i$ copies of $b_1$ followed by $i$ copies of $b_2$ ... followed by $i$ copies of $b_n$. We say that $B^i$ is the $i$-stutter of $B$. For example, if $B$ = wood, then $B^3$ = wwwoooooooddd. To complete the definition, we define $B^0 = \lambda$, where $\lambda$ is a string of length 0. Notice that $\lambda$ is a subsequence of any strings, including itself.

Problem: Given two such strings $A$ and $B$, find the maximal value of $i$ such that $B^i$ is a subsequence of $A$.

### Input

The first line contains integer T ($1 \leq T \leq 100$), the number of test cases. It is followed by 2T lines, where $2i$ -1and $2i$ contain strings $A$ and $B$, respectively, for the $i$-th test case. Strings will contain alphabets (both upper case and lower case) and spaces. You are assured that $m,n \leq 10000$.

### Output

For each test case, output $i$, on a line.

### Sample Input

```
2
stuttering subsequence
se
```

```
hkust programming contest
rmt
```

## Sample Output

```
3
2
```

# HKUST Programming Contest 2002

# Problem 4

## Coin Game

This is a 2-person game in which there is a collection of coins. The coins are separated into *N* piles, with pile *i* having $p_i$ coins. The players alternate moves; during each move, the current player chooses one pile and then takes as many coins as he/she wants from the pile (at least 1 coin must be taken per move). The person who takes the last coin *loses*.

Example: Suppose *N*=3. We will denote the current POSITION by the triple $(p_1, p_2, p_3)$. Suppose $(p_1, p_2, p_3)$= (3,4,5). Here is a possible game:

```
Position:  Next Move
-----------------------------------------------
(3,4,5) :  Player 1  takes 2 coins from pile 3
(3,4,3) :  Player 2  takes 4 coins from pile 2
(3,0,3) :  Player 1  takes 1 coin  from pile 1
(2,0,3) :  Player 2  takes 1 coin  from pile 3
(2,0,2) :  Player 1  takes 2 coins from pile 1
(0,0,2) :  Player 2  takes 1 coin  from pile 3
(0,0,1) :  Player 1  takes 1 coin  from pile 3
(0,0,0)
```

Player 1 LOSES because he took the last coin.

The LENGTH of a game is the number of moves made by all of the players. The length of the game above is 7. A position $(p_1, p_2, \ldots, p_n)$ is a WINNING POSITION if the current player has a move that can always ultimately force a win no matter what the next player responds. Example: (2,0,3) is a WINNING POSITION since the current player can always take 1 coin from pile 3 transforming into position (2,0,2). No matter what the second player now plays the first player will always win. If the second player takes 1 coin from pile 1 the first player takes 2 coins from pile 2. If the second player takes 2 coins from pile 1 the first player takes 1 coin from pile 2 (the other cases are symmetric).

Now, Roger and Martin are going to play such game, and they are so PRO that they will NEVER make mistakes in this game! Also, Roger will be the one who takes the coins first. Then, given a position ( $p_1, p_2, \ldots, p_n$ ), you are required to write a program to predict who will win.

## Input

There are several test cases and each test case will have a line of input. For each line, the first number is *N* and it is followed by *N* numbers $p_1, p_2, \ldots, p_n$ which are the number of coins for pile 1 to pile *N*. Note that $1 \le N \le 5$, $0 \le p_i \le 10$ and there is at least 1 coin in the game. if you read a line with *N*=0, it means the end and your program should terminate.

## Output

Print the name of the winner for each game.

## Sample Input

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

## Sample Output

```
Roger
Roger
Martin
```

# HKUST Programming Contest 2002

# Problem 5

## Interval Graphs

Given a set $S$ of intervals on the real line, we say that $S$ is proper if no interval is contained by another interval or the union of other intervals.

Given an undirected graph $G$=($V$,$E$), we say that $G$ is a proper interval graph if it models the overlapping among a proper set of intervals. That is, each vertex in $V$ corresponds to an interval in $S$ and two vertices in $G$ are connected by an edge in $E$ if and only if the corresponding two intervals overlap. For example, suppose $S=\{a,b,c,d\}$ are intervals and $a$=[0,10], $b$=[2,12], $c$=[4,14], $d$=[13,20]. Then the corresponding proper interval graph will be $G$=($V$,$E$), where $V=\{a,b,c,d\}$ and $E=\{(a,b), (b,c),(c,a),(c,d)\}$.

Your task is to write a program that determines if an input graph $G$ is a proper interval graph or not.

## Input

The input will consist of an arbitrary number of graphs. Each graph will begin with an integer $N$ ($1 \leq N \leq 100$) which stands for the size of $V=\{1,2,\ldots,N\}$. It is immediately followed by another integer $E$ ($1 \leq E \leq 1000$), the number of edges in the graph. Then will follow $E$ lines, each line consists of two integers $i,j$ ( $i \neq j$ and $i,j \leq N$), showing that vertices $i$ and $j$ are connected. Read the input until $N=E=0$, which is not a real test case and should not be processed. You can assume that every input graph has only 1 connected component.

## Output

For each graph, print “YES” if the graph is a proper interval graph, and “NO” otherwise.

## Sample Input

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

## Sample Output

```
YES
NO
```

# HKUST Programming Contest 2002

# Problem 6

## Pick's Theorem

A lattice point on the plane is a point having integer coordinates. The area, $A$, of a convex polygon whose vertices are lattice points is, by Pick's Theorem,

$$A = I + (B/2) - 1,$$

where $I$ is the number of lattice points inside the polygon and $B$ is the number of lattice points on the boundary of the polygon. The theorem is also true for non-convex polygon, but we will only consider convex polygon in this problem.

## Problem

Given $N$ lattice points that form a convex polygon on the plane, you are asked to find $I$ (the number of lattice points inside the polygon), $B$ (the number of lattice points on the boundary of the polygon), and $A$ (the area of the polygon).

## Input

Each test case will begin with an integer $N$, the number of lattice points that form the convex polygon ($3 \leq N \leq 100$). It is followed by $N$ pairs of integers, each pair on a line representing a lattice point $(x,y)$ on the polygon ( $0 \leq x,y \leq 10000$). Input is terminated by EOF.

## Output

For each case, print the values of $B$, $I$, $A$ on a line, separated by spaces.

## Sample Input

```
3
0 0
0 10
10 0
4
0 0
10 0
0 10
10 10
```

## Sample Output

```
30 36 50
40 81 100
```

- End of Problem Set -