2004 National Collegiate Programming Contest

- Problems: There are 9 problems (14 pages in all, not counting this cover page) in this packet.
- Problem Input: Input to the problems are through the input files. Input filenames are given in the table below. Each input file may contain one or more test cases. Test cases may be separated by any delimiter as specified in the problem statements.
- Problem Output: All output should be directed to standard output (screen output).
- Time Limit: The judges will run each submitted program with certain time limit (given in the table below).

Table 1: Problem Information Sheet

	Problem Name	Input File	Time Limit
Problem A	Baseball Team Assembly	pa.in	10 secs.
Problem B	Mining Strategy	pb.in	10 secs.
Problem C	Quadratic Curve Fit	pc.in	10 secs.
Problem D	Affine Decipher	pd.in	10 secs.
Problem E	Killing Cycles	pe.in	10 secs.
Problem F	Lego Decomposition	pf.in	10 secs.
Problem G	Printer Queue	pg.in	10 secs.
Problem H	Job Assignments	ph.in	10 secs.
Problem I	Maximize Revenue	pi.in	10 secs.

Problem A
Baseball Team Assembly
Input File: *pa.in*

It's time to assemble a national baseball team again. We have a list of pitchers and a list of opponents. After years of scouting, we know the winning probability for any pitcher to pitch against any opponent. According to the contract, each pitcher can pitch at most one game, and the starting pitcher will always finish the game by himself . Your job is to select a set of pitchers so that the probability of the team beating all the opponents is maximized.

Technical Specification

1. There are n $(1 \leq n \leq 300)$ pitchers.
2. There are m $(1 \leq m \leq n)$ teams.
3. The value of the winning probability for a pitcher against a team is $0, \frac{1}{5}, \frac{2}{5}, \frac{3}{5} \cdot \frac{4}{5}$, or 1.

Input File Format

The first line of the input file contains an integer indicating the number of test cases to follow. Test cases are separated by a single blank line. For each test case starts with a single line containing two positive integers n and m, where n is the number of pitchers in the candidate list and m is the number of opponents. For each of next m lines, one for each opponent, there are n integers, $p_1, ..., p_n$ where, $p_i \in \{0, 1, 2, 3, 4, 5\}$. If the i^{th} number in the j^{th} line is p, it means that the i^{th} pitcher against opponent j has winning probability $p/5$.

Output Format

For each test case, output the maximum winning probability achievable according to its form. Output integers a,b,c, in that order in a single line, if the winning probability is $\frac{2^a 3^b}{5^c}$; output 0 if the winning probability is 0.

Sample Input

```
2
2 1
2 3

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

Output for the Sample Input

```
0 1 1
2 0 1
```

Problem B
Mining Strategy
Input File: *pb.in*

Consider a popular surface mining operation in which blocks of earth are dug from the surface to extract the ore contained in them. During the mining process, the surface of the land is excavated, forming a deeper and deeper pit until the mining operation terminates. The final shape of this open pit is determined before the mining operation begins. To design the optimal pit–one that maximizes profit–the entire mining area is divided into three-dimensional blocks. Using geological information from drill cores, the value of the ore in each block is estimated. Additionally, the cost of mining each particular block is determined, thus we can assign a profit value to each block in the mine. The optimal pit design problem is then to maximize the total profit of the mine, while satisfying constraints on the slope of the pit walls and constraints that allow underlying blocks to be mined only after blocks on top of them. That is, it is impossible to excavate a block without also excavating those above it. For convenience, we give each block an identification number. You are asked to write a program to calculate the maximum profit that can be achieved and list the evacuated blocks in increasing order.

Input File Format
The input file contains several lines. Each line shows the information of a block with the format: block identification number (positive integer ≤ 1000), its profit value (an integer in the interval $[-10000, 10000]$), and followed by the identification numbers of blocks that are above it, where adjacent numbers are separated by one or more spaces. If there is no block above it, there is no number behind the profit value, such as block 4 in the sample input. The input is terminated by a zero on a line by itself. Your program should be able to handle up to 1000 blocks.

Output Format
The output contains two lines. The first line shows the maximum profit that can be achieved and the second line contains the evacuated block identifications in increasing order.

Sample Input
```
1 2 5
2 -5 1
3 2 2
4 -2
5 3 4
0
```

Output for the Sample Input
```
Profit: 3
Evacuated blocks: 1, 4, 5
```

Problem C
Quadratic Curve Fit
Input File: *pc.in*

Given a set of N two-dimensional points $S = \{(x_i, y_i) \mid 1 \leq i \leq N\}$, the quadratic curve fit problem is to find a quadratic function $y = f(x) = ax^2 + bx + c$ such that the squared error $Error = \sum_{i=1}^{N}(y_i - f(x_i))^2$ is minimized. This problem asks you to report three parameters a, b, c by *round-off* with the precision to 4 decimals for a given data set $S = \{(x_i, y_i) \mid 1 \leq i \leq N\}$.

Input File Format
The first line of the input file always contains one integer indicating the number of test cases to come. Test cases are always separated by a single blank line. Each test data set consists of $N + 1$ lines, where the first line indicates the number of points N ($5 \leq N \leq 7$) which is followed by N pairs of floating-point numbers to 2 decimals.

Output Format
For each data case, report the estimated *parameters* a, b, c by round-off up to 4 decimals in a single line.

Sample Input

```
2
7
-3.01 5.99
-1.99 3.01
-1.01 2.01
0.01 2.99
0.99 5.99
2.01 10.99
-1.20 2.04

5
-2.00 16.00
-1.00 7.00
0.00 2.00
1.00 1.00
2.00 4.00
```

Output for the Sample Input

```
0.9903 1.9867 3.0089
2.0000 -3.0000 2.0000
```

Problem D
Affine Decipher
Input File: *pd.in*

Let Γ={A, B, C, ···, X, Y, Z } be the set of 26 English letters, and let the letters A, B, ···, Y, Z can be represented as numbers 0, 1, ···, 24, 25, respectively. Denote $Z_{26} = \{0,\ 1,\ \cdots, 24, 25\}$. An integer affine transformation $(mod\ N)$ can be defined as $f:\ Z_N \rightarrow Z_N$ with $f(x) = \alpha x + \beta\ (mod\ N)$, where $\alpha, \beta \in Z_N$. The inverse integer affine transformation exists only if $gcd(\alpha, N) = 1$, that is, α and N are relatively prime. This problem assumes that $N = 26$.

An affine encipher takes a message, a character string consisting of English letters from Γ={A, B, C, ···, X, Y, Z }, as input and outputs an enciphered message of the same length based on an integer affine transformation, for example, 'HAPPY' could be enciphered as 'WBUUV' based on $f(x) = 3x + 1\ (mod\ 26)$. An affine decipher $g(y) = 9y + 17\ (mod\ 26)$ is the inverse affine transformation of f which deciphers 'WBUUV' as 'HAPPY'. This problem asks you to design an *affine decipher* based on a given affine encipher to decipher an enciphered message.

Input File Format
The first line of the input file always contains one integer indicating the number of test cases to come. Test cases are always separated by a single blank line. Each test data set consists of two lines, where the first line gives three integers α, β, γ corresponding to the affine encipher $f(x) = \alpha x + \beta$, $x \in Z_{26}$ and the number of characters for the enciphered message given in the next line.

Output Format
Two integers σ, τ corresponding to the affine decipher $g(x) = \sigma x + \tau$, $x \in Z_{26}$ is shown in the first line with the deciphered message appearing in the next line.

Sample Input

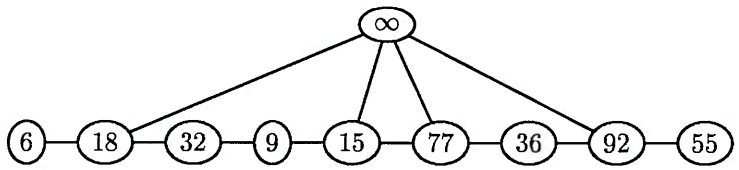
```
2
3 1 5
WBUUV

1 1 25
UBJXBOJTBCFBVUJGVMJTMBOEA
```

Output for the Sample Input

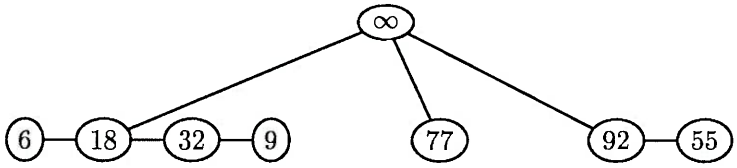
```
9 17
HAPPY
1 25
TAIWANISABEAUTIFULISLANDZ
```

Problem E
Killing Cycles
Input File: *pe.in*

Consider a graph consisting of a path together with a single node v that is adjacent to some of the nodes on the path. Each node in the graph has a cost. The cost of v is infinite.



The goal is to delete a subset of nodes so that no cycle remains in the graph. (By deleting a node we mean removing the node and its incidental edges.) For example, you could delete just the node v; the remaining graph has no cycle. However, you want to delete a subset having minimum cost. Since the cost of v is infinite, you definitely don't want to delete v. For the above graph one possibility is deleting the nodes with costs 15 and 36. The resulting graph is as follows.



Write a program to find a minimum-cost collection of vertices whose deletion results in a graph with no cycles.

Technical Specification

There are n nodes on the path, where $1 \leq n \leq 30$. For each $i = 1, 2, \ldots, n$,

- let $c[i]$ be the cost of the i-th node of the path; and
- Let $a[i]$ be a single bit such that $a[i] = 1$ if and only if the i-th node on the path is adjacent to v.

You may assume that each $c[i]$ is a positive integer no more than 100.

Input File Format

The first line of the input file contains an integer indicating the number of test cases to follow. Test cases are separated by a single blank line. For each test case, the first line of input contains a single integer n. Then, n lines of input follow: For $i = 1, 2, \ldots, n$, the i-th line contains $c[i]$ and $a[i]$.

Output Format

For each test case, output the cost of a minimum cost collection of nodes whose deletion kills all cycles.

Sample Input

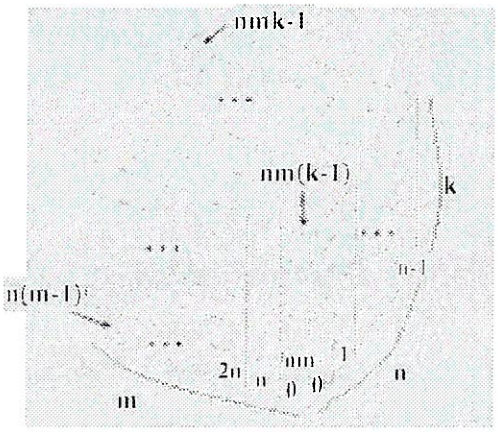
```
2
5
1 1
1 1
1 1
1 1
1 1

9
6 0
18 1
32 0
9 0
15 1
77 1
36 0
92 1
55 0
```

Output for the Sample Input

```
2
51
```

Problem F
Lego Decomposition
Input File: *pf.in*

The Lego Company has come up with a new Lego construction kit. The new kit contains a set of irregular shaped Lego blocks, which are made of connected $1 \times 1 \times 1$ square Lego cubes. This set of Lego blocks, when put together correctly, forms an $n \times m \times k$ rectangular cube. A block removing game is played by removing Lego blocks one-by-one from the fully constructed $n \times m \times k$ rectangular cube. A point is scored if when a Lego block is removed from the rectangular cube, the remaining blocks become or remain disconnected. Two blocks are disconnected if they do not share any common face. The Lego blocks are removed one by one until the entire rectangular cube disappeared. Given a Lego block removing sequence, compute the total points scored. You may assume that Lego blocks do not collaps onto each other during the block removal process.



Technical Restrictions

1. $1 \leq n, m, k \leq 30$
2. The following convention as illustrated in the figure above should be used to reference each $1 \times 1 \times 1$ Lego cube within the rectangular cube.

Input File Format

The first line contains an integer indicating the number of test cases to follow. Test cases are separated by a single blank line. For each test case, the first line contains 4 integers, n, m, k and l, where n, m, k give the dimensions of the fully composed rectangular Lego cube ($n \times m \times k$) and l is the number of Lego blocks that make up this rectangular cube. Each of the next l lines takes the form $p_i \;\; d_1 \;\; d_2 \;\; \ldots \;\; d_{p_i}$, where p_i (≤ 20) is the number of connected 1×1 Lego cubes, namely $d_1 \;\; d_2 \;\; \ldots \;\; d_{p_i}$, in Lego block i. Those l lines fully define all the blocks and the order in which each block is removed.

Output Format

For each test case, print on one line, the number of points scored if the given sequence of block removal is carried out.

Sample Input

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

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

2 3 3 6
2 0 1
2 8 9
2 16 17
4 6 7 12 13
6 2 3 4 5 10 11
2 14 15
```

Output for the Sample Input

```
0
0
2
```

Problem G
Printer Queue
Input File: *pg.in*

The HQ software company is designing a network printing solution for IBN Corporation. The IBN Corporation has n network printers with c computers sharing those network printers. The n printers may print at different speed. Once a computer sends a print job, the printer-queue controller assigns the print job to a printer that can accomplish the print job in the least amount of time (including waiting time) and to notify the user that the print job has completed. Once a job is assigned to a printer, it cannot be removed from the queue and be reassigned. You are hired by HQ to write this printer-queue controller.

Technical Specification

1. A print job must be assigned to the printer that can finish printing at the earlist time possible. If a job can be assigned to two or more printers that would finish the job at the same time, then the job is always assigned to the printer with the fastest printing speed.

2. There are n ($1 \leq 10$) network printers, all print at differnt speeds. However, those printers can start printing new job only at the beginning of a minute. So after a print job is finished, that printer must wait till beginning of the next minute before start printing again (if there is a job waiting to be printed). If a job finished printing at the end of a minute, (which is also beginning of the next minute), then the next print job may start right away.

3. There are c ($1 \leq 20$) computers that would request print jobs.

4. There are at most j ($1 \leq 100$) print jobs requested in all.

5. The printer is maintenance free. That is the printers never run out of ink or paper.

Input File Format

The first line of the input file contains an integer indicating the number of test cases to follow. Test cases are separated by a single blank line. For each test case, the first line of input contains three integers, n, c and j, respectively. The second line contains n integers, indicating the speed of the n printers, respectively. The printer speed is measured in pages per minute. On the next j lines, each contains three integers: T_i, C_i, P_i, meaning that at time T_i, computer C_i requests a printer job to print P_i pages. All the print jobs are listed in non-decreasing order of job submit time T_i.

Output Format

Write a program to simulate a printer queue controller according to the rules specified above. After finding the optimal print job completion time for

each print job requested, output the following information. For each test case, first output *Case j*, where j is the case number, on a line. On the next n lines, output the following for each printer from printer *1* to printer n: N_i, C_i, P_i, FS_i, indicating that the last job printed on printer N_i is from computer C_i, which contains P_i pages and started printing at time FS_i. If a printer did not have any print job, then output *no print job asssigned.* Leave a blank line between two consecutive cases.

Sample Input

```
2
3 1 3
1 2 3
0 1 100
1 1 90
2 1 10

3 2 4
1 2 5
0 1 100
2 2 5
4 1 100
4 2 4
```

Output for the Sample Input

```
Case 1
1 1 10 2
2 1 90 1
3 1 100 0

Case 2
1 no print job assigned
2 2 4 5
3 1 100 20
```

Problem H
Job Assignments
Input File: *ph.in*

Computers $1, 2, \cdots, n$ ($n \leq 100$) are connected in the following manner: For $i = 1, 2, \cdots, n-1$, there is a directed link between computer i and computer $i+1$. The direction of the link is either from i to $i+1$ or from $i+1$ to i. If there is a directed link from i to $i+1$, then computer $i+1$ can get information from computer i. Similarly, if there is a directed link from $i+1$ to i, then computer i can get information from computer $i+1$. Figure 1 is an example of such a connection.

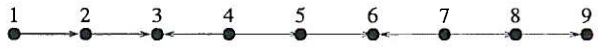


Figure 1: Directed links between computers

There is a group of m ($m \leq 300$) jobs that need to be assigned to the computers. Some of these jobs are related. We put a directed link from job j to job j' to mean that j and j' must be assigned to distinct computers, and the computer assigned to work on job j' needs to get information from the computer which is assigned to work on job j. So if there is a directed link from job j to job j', and job j is assigned to computer i, job j' is assigned to computers i', then we must have $i \neq i'$ and there is a directed link from computer i to computer i'. For example, Figure 2 is a group of jobs with directed links between them.

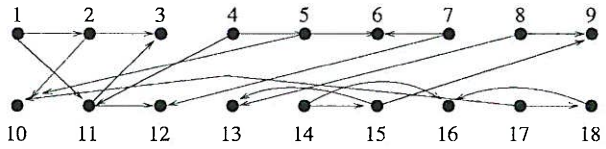


Figure 2: Directed links between jobs

We can assign the computers in Figure 1 to do the jobs in Figure 2 as in Figure 3:

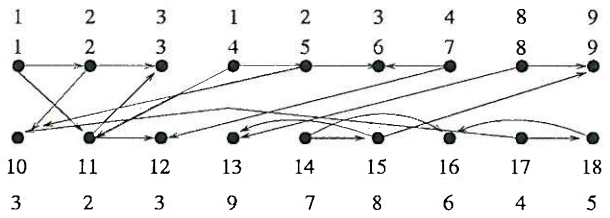


Figure 3: Assigning the jobs to the computers

Given a set of computers and the directed links between them, a set of jobs and the directed links between them, your task is to determine if there is a valid assignment. I.e., each job i is assigned to a computer $\phi(i)$, and if there is a directed link from job i to job j, then there must be a directed link from computer $\phi(i)$ to computer $\phi(j)$. Note that it is not necessary that every

computer has a job, but every job must be assigned to a computer.

Input File Format

The input consists of a number of test cases.

The 1st line of a test case contains a $+,-$ sequence of length $n-1$, which represent the connection between n computers. $+$ means a forward link, $-$ means a backward link. For example $++-++-++$ means that there are 9 computers (and hence 8 links). The directed links are as shown in Figure 1, i.e., $1 \to 2$, $2 \to 3$, $4 \to 3$, $4 \to 5$, $5 \to 6$, $7 \to 6$, $7 \to 8$ and $8 \to 9$. The second line contains one positive integer m, which is the number of jobs. The next m lines contain information of the relation between the m jobs. Each of the m lines contains either a sequence of positive integers, or a single number 0. If the ith line contains the single number 0, then there is no directed links from job i to other jobs. Otherwise, if the ith line contains integer j, then it means that there is a directed link from job i to job j. For example, if the ith lines is

2 4 8

it means that there are directed links from job i to jobs $2, 4, 8$.

The next test case starts after the last line of the previous case. A 0 signals the end of the input.

Output File Format

The output contains one line for each test case. Each line contains a single letter Y or N. Y means that the required assignment exists, and N means that the required assignment does not exist. The second sample input is the example given in Figures 1 and 2, and Figure 3 shows that the output is Y.

Sample Input

```
++
4
2
3
4
0
++-++-++
18
2 11
3 10
0
5 11
6 10
0
6 12
9 13
0
0
12 3
```

```
0
0
15 16
9 13
0
10 18
16
0
```

Output for the Sample Input

```
N
Y
```

Problem I
Maximize Revenue
Input File: *pi.in*

Alphanti drives a truck transporting goods between n ($n \leq 100$) cities. If he drives from city i to city j, he earns c_{ij} ($-1000 \leq c_{ij} \leq 1000$) dollars. Find a route that maximizes the money he can make in k ($k \leq 300$) trips (he starts from city s, and needs not be back to the city s when he finishes his kth trip).

Input File Format

The input consists of a number of test cases.

The 1st line of a test case contains three positive integers n, s, k, separated by single spaces, where n is the number of cities, and s is the city where Alphanti starts, and k is the number of trips he will make. This is followed by n lines, each line contains n integers. The jth number of the ith line is c_{ij}: the number of dollars earned by one trip from city i to city j.

The next test case starts after the last line of the previous case. A 0 signals the end of the input.

Output Format

The output contains one line for each test case. Each line contains an integer, which is the maximum revenue can be gained by Alphanti in k trips.

Sample Input

```
3 2 10
0 2 5
3 0 2
-1 2 0
5 1 11
0 10 12 13 2
2 0 5 3 12
23 21 0 3 15
-2 3 -4 0 6
-12 14 5 -10 0
0
```

Output for the Sample Input

```
33
188
```