1. There are 12 problems, 25 pages, in this programming contest.
2. The input data is stored in a file, and it may contain many test cases. Your program must process all the test cases in the file.
3. For reading and writting files, use *standard input* and *standard output* in your program such as `stdin` and `stdout` in **C**.

Problem A
Dating Game
Time limit: 2 seconds

The prettiest girl (Belle) at the NCPC University has many suiters. Belle majors in Computer Science, so she devised the folloing puzzle for a dating competition. Whoever can solve the puzzle with the best result wins a date with her. She wrote a program that would generate n lines of random positive integers. Each line has a different number of integers on it. In fact, there are exactly i integers on the i^{th} line. The suiters must connect the numbers from the first line to the n^{th} line, one number per line, to yield the largest sum. Note that the j^{th} number on line i can only be connected to the j^{th} or $(j+1)^{th}$ number on line *i+1*. In the example below, the largest sum is achieved by connecting the numbers in circle.

②
5 ④
3 4 ⑦
1 6 ⑨ 6

Input Format

An instance of the problem consists of several lines of input. The first line contains one integeter, n, $1 \leq n \leq 100$ denoting the number of lines of integers to follow. For the next n lines, each line contains exactly *1, 2, ..., n* positive integers ($\leq 10,000$), respectively. Note that the test data may contain more than one instances. The last instance is followed by a line containing a single 0.

Output Format

For each instance, output on a single line the maximum sum of the connected numbers from line 1 to line n.

Sample Input

```
3
1
2 3
4 5 6
4
2
5 4
3 4 7
1 6 9 6
```

```
0
```

Sample Output for the Sample Input

```
10
22
```

Problem B
Creating Lists
Time limit: 2 seconds

Garbage Collection is an important issue for memory management in operating system. Suppose we maintain the available free memory with a list of n nodes, where each node consists of a fixed size of continuous memory. To allocate the memory into a specific length more efficiently, we want to break the list into a few smaller lists such that it would be possible to create a list of any length with 1 to n nodes by reconnecting some of the smaller lists. We split the list by removing several single nodes one by one. When we remove a node from the list, the node turns into a list with only one node and the other part(s) become smaller list(s).

For example, consider a list of length 10 and the nodes are indexed from left to right with numbers from 1 to 10. By removing the 4th and 10th nodes, we have 4 shorter lists of length 3, 1, 5 and 1, respectively. Then it is clear that we can construct lists of any length from 1 to 10 with these 4 smaller lists. The following illustrates a list with a pair of brackets.

```
Before:  [1, 2, 3, 4, 5, 6, 7, 8, 9, 10]
After :  [1, 2, 3],  [4],  [5, 6, 7, 8, 9],  [10]
```

What is the minimum number of single node that must be detached from the list to achieve the goal? Your task is to write a program to find the minimum number of node that should be removed from the list.

Input Format

The first line of the input gives the number of test cases, T (≤ 50). T test cases follow. Each test case consists of one line that has an integer n ($2 \leq n < 2^{32}$), where n indicates the number of nodes in the list.

Output Format

For each test case, output your answer.

Sample Input

```
2
10
5
```

Sample Output for the Sample Input

```
2
1
```

Problem C
Binary Search Trees
Time limit: 3 seconds

Suppose that we have numbers between 1 and 100000 in a binary search tree, and we want to search for the number x. You are asked to determine whether a sequence of numbers is a possible examined sequence or not.

Input Format

In each problem instance, there is a line containing the numbers in a sequence. The last number in a sequence is the number that we want to search. There are at most 100 numbers in a sequence. Note that there is a blank between any two numbers in each line.

Note that the test data file may contain more than one instances. The last instance is followed by a line containing a single 0.

Output Format

For each test case, output 1 in a line if it is a possible examined sequence; otherwise, output 0 in a line.

Sample Input

```
924 220 911 244 898 258 240 247
2 252 401 398 330 344 397 363
0
```

Sample Output for the Sample Input

```
0
1
```

Problem D
Design of the Size of Packages
Time limit: 5 seconds

Each year lots of students take College entrance exam, which is for high school students to test whether they have the basic knowledge to enter universities in Taiwan. After the exam, students may select a certain number of their favorite departments from various universities to apply for admission. Each department offers different available admission spots and according to the second phase oral exam result hold by each department, has a unique ranking list of the students who applied that department for admission. Of course each student can enrol to only one department and each department cannot accept students exceed its available admission spots. However, each student may apply for a number of departments in different schools and each department would like to admit as good students as possible according to its own ranking list priority. You have to write a program to determine the match result that can be announced as soon as possible. Please output the total number of students who admitted by a department and the total number of spots which are still vacant after the matching procedure.

Here is an example with three students s_1, s_2, and s_3 and three departments d_1, d_2,and d_3. The available admission spots of d_1, d_2, d_3 are 1, 1, 2 respectively. The ranking lists of the students and department from high to low are as follow:

$$\begin{array}{ll} s_1 : d_1, d_2 & d_1 : s_3, s_2, s_1 \\ s_2 : d_1, d_2 & d_2 : s_3, s_1, s_2 \\ s_3 : d_1, d_2, d_3 & d_3 : s_3 \end{array}$$

Then the match plan $\{(s_1, d_2), (s_2, d_1), (s_3, d_3)\}$ is not acceptable since d_1 prefers s_3 than s_1, and s_3 also prefers d_1 than d_3. Hence, $\{(s_1, d_2), (s_3, d_1)\}$ is the desired match plan in this example which has 2 students be admitted and total of 2 vacant spots left.

Technical Specification

- There are S students taking an oral exam. The students are numbered from 1 to S, $1 \leq S \leq 100000$.
- There are a total of D departments in all schools, which are labelled from 1 to D, $1 \leq D \leq 1000$.
- Each student will submit a preference list of departments up to 10 in advance. If one is admitted in the department of higher rank in his/ her preference list, he/she would not consider the department of lower rank in his/her preference list, since each student can only enrol to one department.
- Each department has different quota for students taking oral exam. The quota can not be transferred between departments.

Input Format

The first line is an integer indicating the total number of test data. The first line of each test data consists two integers S and D with a space between them, which represent the number of students and departments respectively. Then there are S+D lines, and each line contains several integers separated by a space. The first S lines are the preference list for each student (end by 0); the next D lines are the quota offered (which is the first number) and the admission list for each department (end by 0). Each list is from higher priority to lower priority.

Output Format

For each test case, output two integers separated by a space in one line. The first integer indicates the total number of students who has been admitted by a department and the second integer indicates the total number of spots that are still vacant.

Sample Input

```
2
3 3
1 2 0
1 2 0
1 2 3 0
1 3 2 1 0
1 3 1 2 0
2 3 0
2 2
1 2 0
2 0
2 1 2 0
1 2 0
```

Sample Output for the Sample Input

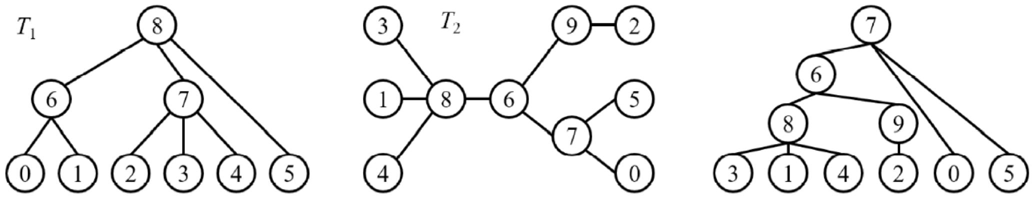
```
2 2
2 1
```

Problem E
Subtrees Matching
Time limit: 3 seconds

Phylogenies or evolutionary trees represent the evolutionary relationships among species. Since the actual evolutionary relationships among species are usually unknown, most phylogenies are constructed based on biological analyses. However, different data or construction methods may generate different phylogenies for the same set of species. To further analyze the relationships among species and improve the phylogeny construction methods, it is important to measure the differences among phylogenies of the same set of species. One such measure involves comparing the subtrees of two different phylogenies.

In this problem, you are given a positive integer k, a rooted tree T_1, and a free tree T_2. Both trees have l leaves that are uniquely labeled from 0 to $l-1$. Let n_1 and n_2 be, respectively, the numbers of internal nodes (non-leaf nodes) of T_1 and T_2. The internal nodes of T_1 are labeled from l to $l+n_1-1$ and node $r_1 = l+n_1-1$ is its root. The internal nodes of T_2 are labeled from l to $l+n_2-1$ and no node is specified as its root. Your job is to select an internal node as the root of T_2 such that T_2 has the largest *similarity* with T_1.

The similarity of two rooted trees is defined as follows. Assume that T_2 is rooted at a selected node r_2. (Note that r_2 is an internal node.) For any non-root internal node v of T_1 (and T_2, resp.), define $T_1(v)$ (and $T_2(v)$, resp.) to be the subtree rooted at v. For a pair of subtrees $T_1(u)$ and $T_2(v)$, the leaves that appear in both subtrees are their *shared leaves* and the number of shared leaves is denoted by $\mathit{Shared}(T_1(u), T_2(v))$. Recall that k is a given integer. The *similarity* of T_1 and T_2 is the number of subtree pairs $(T_1(u), T_2(v))$ with $\mathit{Shared}(T_1(u), T_2(v)) \geq k$, where $l \leq u \neq r1 \leq l+n_1-1$ and $l \leq v \neq r_2 \leq l+n_2-1$. Note that in this problem, we only consider *proper* subtrees. More specifically, the tree itself and the leaf nodes are not considered as subtrees for the computation of similarities.



(a) rooted tree T_1 (b) free tree T_2 (c) root T_2 at node 7

Figure 1: Figure 1

For example, consider the two trees in Figure 1 (a) and (b), in which $l = 6$, $n_1 = 3$, and $n_2 = 4$. Assume that $k = 2$ and node 7 is selected as the root of T_2. (See Figure 1 (c).) The tree T_1 has 2 proper subtrees $T_1(6)$, $T_1(7)$, and the tree T_2 (rooted at node 7) has 3 proper subtrees $T_2(6)$, $T_2(8)$ and $T_2(9)$. The total number of subtree pairs is $2 \times 3 = 6$. The subtree pair $(T_1(7), T_2(6))$ has 3 shared leaves. The subtree pair $(T_1(7), T_2(8))$ has 2 shared leaves. The subtree pairs $(T_1(6), T_2(6))$, $(T_1(6), T_2(8))$, and $(T_1(7), T_2(9))$ have 1 shared leaf. The subtree pair $(T_1(6), T_2(9))$ has 0 shared leaf. Consequently, if $k = 2$ and node 7 is selected as the root of T_2, the similarity of T_1 and T_2 is 2.

Input Format

There are at most 10 cases. The first line of each test case gives the 4 integers l, n_1, n_2, and k, where $2 \leq l \leq 3000$ and $1 \leq n_1, n_2, k \leq 3000$. The first line is followed by the edge lists of T_1 and T_2. The edge list of T_1 is $l + n_1 - 1$ lines with two integers i and j representing edge (i, j). The edge list of T_2 is $l + n_2 - 1$ lines with two integers i and j representing edge (i, j). The input is terminated by a line containing four zeros, which should not be processed.

Output Format

For each case, output the largest similarity and the selected root. In case several nodes can be selected as the root, output the node with the largest label.

Sample Input

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

```
5 7
0 7
0 0 0 0
```

Sample Output for the Sample Input

```
1 4
3 9
```

Problem F
Message delivery
Time limit: 5 seconds

Input File: *f.in*
Time Limit: *2 Seconds*

National Computer Programming Company (NCPC) is a well-established corporation that attracts many young talents every year. The NCPC has a map, in which there are N checkpoints, denoted by $0, 1, \ldots, N-1$, including the source checkpoint S and terminal checkpoint T, and K edges such that $0 \leq K \leq 20000$. For any two checkpoints u and v, they can be connected by at least one edge, or do not be connected by any edge. Each edge (u, v), where $0 \leq u, v \leq N-1$, has exactly one direction from the starting checkpoint u to the ending checkpoint v, which is associated with a rational $e_{uv} = \frac{q}{p}$ representing the probability of successfully passing this edge, where $0 < e_{uv} = \frac{q}{p} \leq 1$ and $0 < q \leq p \leq 33$. We assume that probabilities associated with the edges are independent.

The NCPC wants to deliver a message from S to T by finding a path from S to T with the maximum successful probability $\frac{q}{p}$ in an irreducible form, and outputs the numerator q and the denominator p of the probability; otherwise, if such a path does not exist, then outputs -1.

For an example shown in Figure 2, there are seven checkpoints and nine edges in the given map. The probabilities of all the edges are $e_{01} = \frac{3}{25}$, $e_{02} = \frac{3}{20}$, $e_{03} = \frac{1}{2}$, $e_{14} = \frac{1}{5}$, $e_{25} = \frac{1}{5}$, $e_{34} = \frac{1}{25}$, $e_{35} = \frac{3}{5}$, $e_{46} = \frac{1}{10}$, and $e_{56} = \frac{2}{5}$. The path $\langle 0, 3, 5, 6 \rangle$ has the maximum successful probability $\frac{1}{2} \times \frac{3}{5} \times \frac{2}{5} = \frac{3}{25}$. The output is 3 and 25.

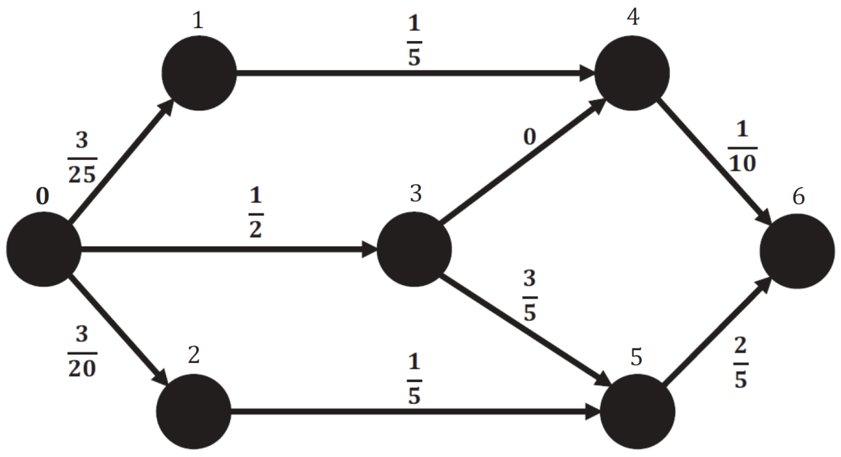


Figure 2: Map 1

For another example shown in Figure 3, there are five checkpoints and five edges in the given map. The probabilities of all the edges are $e_{01} = \frac{1}{10}$, $e_{02} = \frac{1}{5}$, $e_{13} = \frac{1}{5}$, $e_{23} = \frac{1}{20}$, and $e_{43} = \frac{1}{2}$. Because there is no path from S to T, the output will be -1.

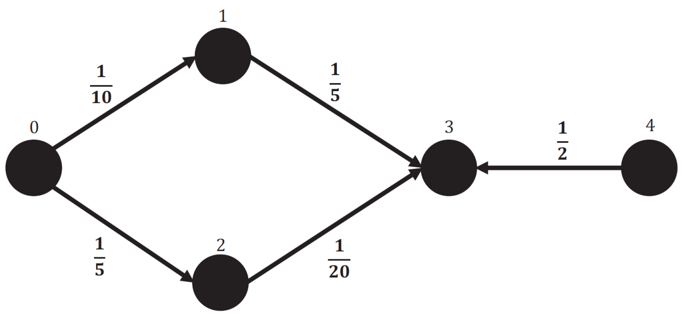


Figure 3: Map 2

Technical Specification

1. $2 \leq N \leq 1000$ for each test case.
2. The probability of each edge is $0 < e_{uv} = \frac{q}{p} \leq 1$, where $0 < q \leq p \leq 33$.
3. The number of edges K is a positive integer at most 20000.
4. For each checkpoint v, if v is reachable from S, the maximum successful probability from S to v can be represented by $\frac{q}{p}$, where $0 < q \leq p \leq 2^{32}$.

Input File Format

The first line of the input file contains an integer L ($L \leq 11$) that indicates the number of test cases as follows. For each test case, the first line contains one integer representing the number of checkpoints; the second line contains two integers S and T, separated by a space, representing the source checkpoint and the terminal checkpoint, respectively; and the third line contains the number of edges K. Then it is immediately followed by K lines, in which each line contains four integers that represent the starting checkpoint, the ending checkpoint, the numerator, and the denominator of the probability of successfully passing the corresponding edge (any two consecutive integers are separated by a space).

Output Format

The output contains one line for each test case. Each line contains two positive integers q and p, separated by a space, where q and p are the numerator and denominator of the maximum successful probability. Note that $\frac{q}{p}$ is an irreducible fraction.

Sample Input

```
2
7
0 6
9
0 1 3 25
0 2 3 20
0 3 1 2
1 4 1 5
2 5 1 5
3 4 1 25
3 5 3 5
4 6 1 10
5 6 2 5
5
0 4
5
0 1 1 10
0 2 1 5
1 3 1 5
2 3 1 20
4 3 1 2
```

Output for the Sample Input

```
3 25
-1
```

Problem G
Alien Cells from Mar
Time limit: 5 seconds

Input File: *pg.in*
Time Limit: *1 Second*

In year 2500, some alien cells are discovered from Mar and sent back to earth for observation. In the lab, the bio scientists want to trigger the replication process of these alien cells. In a petri dish, bio scientists plant an alien cell. They find that in most of the time, an alien cell can replicate itself from one cell into two identical cells. Sometimes, the replication can go wrong in cell division and a cell can become two different cells. More interestingly, an alien cell sometimes can replicate into 3 or more cells (identical or not) at a time.

By collecting the data and studying the process for a period of time, scientists begin to know some types of alien cells stop replication forever.

Let the alien cell names be 'A'-'Z','a'-'z', where 'A' is the original alien cell. If a cell is named by 'a'-'z', it stops replication and maintains its form. An example of cell replication rules can be summarized as Table 1.

rule#	original	after	energy consumed
1	A	BA	10
2	A	bcd	5
3	B	c	4

Table 1: An example of cell replication table.

Note that it is possible to have an "after" string mixed of 'A'-'Z','a'-'z'. Scientists observe that when a rule is applied, an amount of energy is consumed (see the rightmost column in the table). So, according to the table in Table 1, if a scientist plants a cell 'A' (the original alien cell), it can eventually replicate itself into 'bcd', or 'cbcd', or 'ccbcd'...

Given a string of cell names (mixed of 'A'-'Z','a'-'z') , please compute the minimum total energy consumed by the replication of alien cells.

Input File Format

The test data begins with an integer number N, which is the number of test cases. Each test case begins with an integer number $M(1 \leq M \leq 20)$, which is the number of rules. Each rule contains two strings and one integer, which are separated by a space. The first string is the "original" string which only contains a letter from 'A'-'Z'. The second string is "after" string mixed of 'A'-'Z','a'-'z'. The integer is the energy consumed of the rule.

Following the rule table is a string of cell names (mixed of 'A'-'Z','a'-'z'). The length of cell string is less than 50.

Output Format

For each test case, please output the minimum total energy consumed by the multiplication. If a final string is unreachable by the rules, please output *NIL*.

Sample Input

```
3
3
A BA 10
A bcd 5
B c 4
ccbcd
3
A AAAAAAA 20
A AA 15
A a  5
aaaaaaaa
3
A AAAAAAA 20
A AAA 15
A a  5
aaaa
```

Output for the Sample Input

```
33
75
NIL
```

Problem H
Slither is Back
Time limit: 5 seconds

Time Limit: *1 Second*

The famous game Slither is back. In Slither, each player controls a snake. When a snake's head runs into another snake's body, it dies and its corpses become the food for other snakes. When a snake eats the corpses, its length increased.

Fig. 4 is an example slither field of 6×6, where $(0, 0)$ is the bottom left corner cell. A snake is represented by a capital English letter from A to Z. For example, in figure 4, a snake is represented by "A1, A2. . . .A5", where A1 is the head and its body is A2-A5. Another snake is "B1, B2,. . . .B5". If a cell in field is marked with a '@', it represents the corpse of a dead snake.

	@	@	@	@	
					A1
					A2
	B1				A3
	B2				A4
(0,0)	B3	B4	B5		A5

Figure 4: An 6x6 example of Slither field.

The following is the game rules:

Movement

- The field size is always $(100, 100)$, from $(0, 0)$ to $(99, 99)$.
- All snakes move at the same speed – one cell per second.
- A snake can only move in three directions - straight ahead, its left, and its right. A snake is not allowed to move backward directly.
- A snake can run over his own body. For example, it can circle left three times to run over its own body.
- in each second, all the snakes are moved by alphabetical order. That is, snake 'A' moves first, 'B' next, and then "C,D,....".
- A snake's body moves as follows: Let a snake be $X_1, X_2, .., X_n$, if X_i is at (x, y) at time t, then X_{i+1} is at (x, y) at time $t + 1$ for all $i, 1 \leq i \leq n$.

Collision

- Before detecting any collision, always move a snake completely .
- If a snake runs into a cell outside of $(0, 0)$ to $(99, 99)$, it dies. Its head is gone but its body remains as '@'.
- If a snake's head (say X1) runs into other living snake (head or body), snake X dies. Its head disappears and its body remains as '@'
- When a snake dies, its head disappears and its body immediately becomes '@' in the field. Since a snake can run over its own body, it is possible to have multiple body sections die in a cell. In that case, only one '@' is left to the cell. A cell can only store one '@' at any time.

Eating

- Collision is always processed before eating.
- If a snake head moves to a cell and the cell has '@', it eats the corpse and increases its length by 1. That is, Let a snake be $X_1, X_2, .., X_n$ and X_n is at (x, y) at time t. At time $t + 1$, X_1 reaches a cell and eats a '@', the snake becomes $X_1, X_2, .., X_n, X_{n+1}$ and $X_{n+1}$is at (x, y) at time $t + 1$.
- When a '@' is eaten by a snake, it is removed from the cell.

Given an initial configuration of snakes and the moves of snakes in next n seconds, please output the name of the longest snake and its length.

Input File Format

The test data begins with an integer N which is the number of test cases. Each test case begins with a number S ($1 \leq S \leq 26$) which is the number of snakes. Each snake's data begins with a capital letter from 'A'-'Z' and then followed by the cell coordinates of snake body sections from head to tail. The cell coordinates are cartesian coordinate system, where the bottom left corner cell is (0,0)

Following the snake data is an integer T ($1 \leq T \leq 100$), which is the number of seconds of a game to be played. The moves of each snake begin with the snake name and followed by T letters indicating T moves, which are separated by a space. A move is either 'L','R', or '0', where 'L' is moving to left, 'R' is moving to right, and '0' is moving straight.

Output Format

For each test case, please output the longest snake's name and its length after T seconds. The name and the length is separated by a space. If there are more than one longest snakes, please output the snake name with least alphabetical order and its length. If there are no snakes alive, please output “null”.

Sample Input

```
1
2
A 1 6 2 6 3 6 4 6
B 4 5 3 5 2 5
4
A R R R L
B L L R L
```

Output for the Sample Input

```
B 4
```

Problem I
Seating Arrangement
Time limit: 5 seconds

Susan is going to have her wedding party. The wedding venue has Q tables, and each has a different number of seats. Guests are from P families. Susan sets the following rule to increase the guests' social interaction: no two members of the same family are arranged at the same table.

Given the ith family has p_i members and the jth table has q_j seats, Susan would like to find a seating arrangement that meets her rule.

Input Format

An instance of the problem consists of three lines. The first line contains two integers, P ($1 \leq P \leq 50$) and Q ($1 \leq Q \leq 200$), denoting the number of families and tables. The second line has P integers p_i ($1 \leq i \leq P, 1 \leq p_i \leq 50$), representing the number of members in each family. Similarly, the third line has Q integers q_j ($1 \leq j \leq Q, 1 \leq q_j \leq 12$), representing the number of seats in each table.

Note that the test data file may contain more than one instances. The zero value of P and Q indicate the end of test cases and should not be processed.

Output Format

The maximum number of guests can be seated

Sample Input

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

Sample Output for the Sample Input

```
8
11
```

Problem J
Health or Fever?
Time limit: 3 seconds

There is a village where all villagers have only two health conditions: healthy or having a fever. Only the village doctor can do the diagnosis (i.e., the villagers cannot determine the health condition by themselves). The doctor makes a diagnosis by asking patients how they feel. The villagers response only three possible symptoms: normal, cold, or dizzy.

The doctor believes that the health condition of his patient operates as a state machine. There are two states, "health" and "fever", but the doctor cannot observe them directly. However, the transition probabilities are known, as shown in Table 2. For example, if a patient is healthy today, the probability that the person stays healthy tomorrow is 0.7; however, there is 30 percent chance that the person becomes feverish tomorrow. The emission probabilities are also known, as shown in Table 3. For example, suppose a patient is healthy, the probability that the person feels normal is 0.5. Similarly, the probabilities that the person feels cold or dizzy in the health condition are 0.4, and 0.1, respectively.

Now, a patient tells the doctor how he feels (normal, cold, or dizzy) for N consecutive days. We assume that the patient is always healthy in the first day. What is the most probable health condition sequence (i.e., a binary sequence of "health" or "fever") of this patient for these N days? Please compute the number of "health" in the health condition sequence.

Table 2: Transition probabilities of the two health conditions

	`health`	`fever`
`health`	0.7	0.3
`fever`	0.4	0.6

Table 3: Emission probabilities of a health condition to the symptoms

	`normal`	`cold`	`dizzy`
`health`	0.5	0.4	0.1
`fever`	0.1	0.3	0.6

Input Format

An instance of the problem consists of two lines. The first line contains an integer N ($1 \leq N \leq 1000$), indicating the number of consecutive days the patient sees the doctor. The second line contains an integer sequence of length N, and each integer indicates how he feels: 1 (normal), 2 (cold), 3 (dizzy).

Note that the test data file may contain more than one instances. The last instance is followed by a line containing a single 0.

Output Format

The number of “health” in the most probable sequence

Sample Input

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

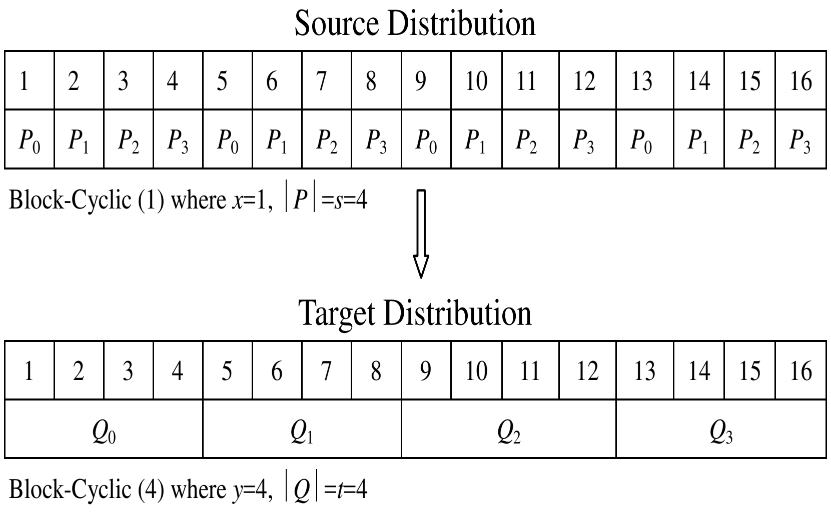
Sample Output for the Sample Input

```
3
(the most probable conditions are: h h h)
1
(the most probable conditions are: h f f)
2
(the most probable conditions are: h h f f f f f)
```

Problem K
Data Redistribution
Time limit: 1 seconds

Parallel computing systems have been extensively adopted to resolve complex scientific problems efficiently. When processing various phases of applications, parallel systems normally exploit data distribution schemes to balance the system load and yield a better performance. Many parallel programming languages thus support run-time primitives for rearranging a programs array distribution. A Block-Cyclic(x) to Block-Cyclic(y) data redistribution from s processors to t processors can be denoted by BC(x, y, P, Q) where $P(Q)$ denotes the source processor set $\{P_0, P_1, \ldots, P_{s-1}\}$ (destination processor set $\{Q_0, Q_1, \ldots, Q_{t-1}\}$) where $x(y)$ denotes its distribution pattern.

For example, the following figure depicts the a data redistribution pattern BC(1, 4, 4, 4) for array $A[1 : 16]$. In BC(1, 4, 4, 4), array $A[1], A[2], A[3], A[4], A[5], \ldots, A[16]$ are assigned and stored cyclically among $P = s = 4$ processors (with block size $x = 1$); that is, array $A[1], A[2], A[3], A[4], A[5], \ldots, A[16]$ are stored originally in $P_0, P_1, P_2, P_3, P_0, P_1, P_2, P_3, P_0, P_1, P_2, P_3, P_0, P$
respectively in the source processor set. On the other hand, array $A[1], A[2], A[3], A[4], A[5], \ldots, A[16]$ are to be redistributed cyclically among $Q = t = 4$ processors (with block size $y = 4$); that is, array $A[1], A[2], A[3], A[4], A[5], \ldots, A[16]$ are to be stored to $Q_0, Q_0, Q_0, Q_0, Q_1, Q_1, Q_1, Q_1, Q_2, Q_2, Q_2, Q_2, Q$
respectively in the destination processor set.


Source Distribution

1	2	3	4	5	6	7	8	9	10	11	12	13	14	15	16
P_0	P_1	P_2	P_3	P_0	P_1	P_2	P_3	P_0	P_1	P_2	P_3	P_0	P_1	P_2	P_3

Block-Cyclic (1) where x=1, $|P|$=s=4

Target Distribution

1	2	3	4	5	6	7	8	9	10	11	12	13	14	15	16
Q_0				Q_1				Q_2				Q_3			

Block-Cyclic (4) where y=4, $|Q|$=t=4

Assume that all processors send message synchronously, and each processor can send only an array element to a desired target processor successfully within a single step due to limited network bandwidth. In BC(1, 4, 4, 4), processor P_0 needs to send A[1] to Q_0, processor P_1 also needs to send $A[2]$ to Q_0. Evidently, when the above two messages are sent at the same step, a conflict occurs at Q_0. When processor P_0 sends $A[1]$ to Q_0 , processor P_0 cannot send $A[5]$ to Q_1 at the same step, because each processor can only send a message (an array element) to a

single processor in each step. However, the following message communications can be achieved in the same step without a conflict:
(1) P_0 sends $A[1]$ to Q_0
(2) P_1 sends $A[10]$ to Q_2
(3) P_2 sends $A[7]$ to Q_1
(4) P_3 sends $A[16]$ to Q_3

Suppose that P_0 tries to sends two array elements (e.g., $A[1]$ and $A[2]$) to Q_0. Note that these two array elements should be scheduled in two different steps here. P_0 can send $A[1]$ to Q_0 in the first step, and then P_0 sends $A[2]$ to Q_0 in the second step. After careful arrangements, the above data redistribution (i.e., BC(1, 4, 4, 4) for array $A[1 : 16]$) can be scheduled within four steps without a conflict.

Please write a program to compute the minimum number of steps without any communication conflict for a given data redistribution.

Input Format

There are at most 10 test cases. The first line of each instance consists of an integer $n (10 \leq n \leq 100)$, where n is the size of the given array for distribution. The second line of each instance contains four integers: x, y, P, Q to represent the desired Block-Cyclic(x) to Block-Cyclic(y) data redistribution BC(x, y, P, Q), where $1 \leq x, y, P, Q \leq 50$. The last test case will be followed by a line containing $Z = 0$.

Output Format

The output for each instance should contain an integer denoting the minimum number of steps without any conflict for a given data redistribution.

Sample Input

```
16
1 4 4 4
17
1 4 4 4
0
```

Sample Output for the Sample Input

```
4
5
```

Problem L
Medicine design
Time limit: 2 seconds

A company is now working on a medicine design project, and a key step of the project is to align a set of molecules to a base molecule. In this task, a molecule can be represented as a binary string, which can also be thought of as an array of '1' and '0'. For integers $x \leq y$, let $[x, y]$ denote the set of all integers between x and y. Let $|S|$ denote the length of a string S. For $1 \leq i \leq |S|$, the i-th element of S is denoted by $S[i]$, and the substring from the i-th to the j-th elements is denoted by $S[i, j]$. To align a string S to a base string B means to find an increasing function $g : [1, |S|] \to [1, |B|]$ such that $S[i] = B[g(i)]$ for all $i \in [1, |S|]$. In other words, S is a subsequence of $B[g(1), g(|S|)]$. For this alignment, the range $[g(1), g(|S|)]$ is called the aligned range. To help the project, you are asked to write a program to align some of the given strings to a base string.

You will be given two sets of binary strings $\mathcal{S}$ and $\mathcal{T}$, where $\mathcal{S} = \{S_i \mid 1 \leq i \leq n\}$ and $\mathcal{T} = \{T_i \mid 1 \leq i \leq n\}$. The base string is a repeated string $B = Q^k$, where k is a positive integer and Q^k is the concatenation of k times of Q. For example, if $Q = 110$ and $k = 3$, then $B = Q^3 = 110110110$. For $x \in \mathcal{S} \cup \mathcal{T}$, there is a profit $p(x)$. A pair $(\mathcal{X}, \mathcal{Y})$, where $\mathcal{X} \subseteq \mathcal{S}$ and $\mathcal{Y} \subseteq \mathcal{T}$, is feasible if all strings in $\mathcal{X} \cup \mathcal{Y}$ can be aligned to B such that the following conditions are satisfied.

- All the aligned ranges are mutually disjoint.
- The strings in $\mathcal{X}$ must be aligned left to the strings in $\mathcal{Y}$. That is, if $[L, R]$ is the aligned range for $x \in \mathcal{X}$ and $[L', R']$ is the aligned range for $y \in \mathcal{Y}$, then $R < L'$.

The total profit of a feasible solution $(\mathcal{X}, \mathcal{Y})$ is the sum of all the profits of strings in $\mathcal{X} \cup \mathcal{Y}$. Your task is to find the maximum profit among all feasible solutions.

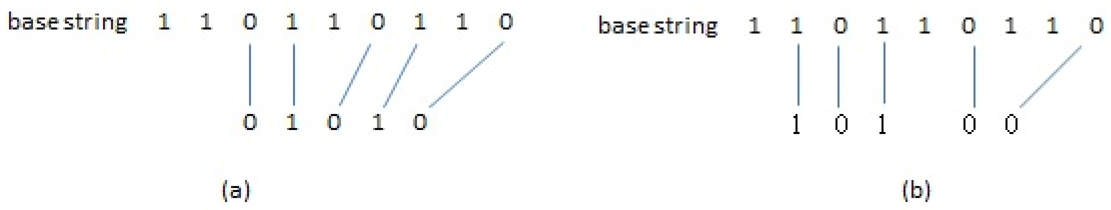


Figure 5: The base string is $B = Q^3 = 110110110$, where $Q = 110$. (a) Aligning a string 01010 to the base string. (b) Two strings 101 and 00 are disjointly aligned to the base string, i.e., the two aligned ranges are disjoint.

Input Format

The first line of the input contains an integers C indicating the number of test cases. For each test case, the first line contains the binary string Q and the integer k. As described above, the base string $B = Q^k$. The second line is the integer n, which is the number of strings in $\mathcal{S}$ and $\mathcal{T}$. In the next $2n$ lines, each line contains a binary string and its profit, separated by a space. The first n strings are in $\mathcal{S}$, and remaining n strings are in $\mathcal{T}$. You may assume the following specifications: $C \in [1, 16]$, $|Q| \in [1, 64]$, $k \in [1, 100]$, $n \in [1, 17]$, and $|\lambda| \in [1, 400]$ for all $\lambda \in \mathcal{S} \cup \mathcal{T}$. The profit of any string is a positive integer at most 50000. The strings are not necessarily distinct.

Output Format

For each test case, output the maximum profit in one line.

Sample Input

```
2
110 3
2
101 5
01010 12
00 3
11 2
1101 2
3
101 2
101 5
000 7
1 3
111 4
100 2
```

Sample Output for the Sample Input

```
12
9
```