2015 ACM ICPC Asia Taipei Regional

December 6, 2015

- Problems: There are 12 problems (34 pages in all) in this packet.
- Program Input: Input to the program are through standard input. Program input may contain one or more test cases. Test cases may be separated by any delimiter as specified in the problem statements.
- Program Output: All output should be directed to the standard output (screen output).
- Time Limit: Judges will run each submitted program with certain time limit (given in the table below).

Problem Information

	Problem Name	Time Limit
Problem A	Parentheses	3 sec.
Problem B	Linear Ecosystem	1 sec.
Problem C	Least Crucial Node	1 sec.
Problem D	Discrete Logarithm Problem	1 sec.
Problem E	Locating Facilities	3 sec.
Problem F	Decode Alien Message	5 sec.
Problem G	Maximum Cut Order	3 sec.
Problem H	Separating Pebbles	12 sec.
Problem I	Social Network	3 sec.
Problem J	Task Review	5 sec.
Problem K	Robots	8 sec.
Problem L	Reward the Troop	2 sec.

Problem A
Parentheses
Time Limit: *3 Seconds*

A bracket is a punctuation mark, which is used in matched pairs, usually used within articles or programs. Brackets include round brackets, square brackets, curly brackets, angle brackets, and various other pairs of symbols. Let's focus on the round brackets, also called *parentheses*.

A sequence of parentheses is said to be *well-formed* if the parentheses are properly nested. For example, $A = a_1a_2...a_{18}$ = "(()())()()()(())()" is well-formed, but $B = b_1b_2...b_{18}$ = "(()()))((((((()((" is not. (See Figure 1.) More formally, a sequence of parentheses $P = p_1p_2...p_n$ is well-formed if (1) when scanning it from p_1 to p_n, the number of right parentheses does not exceed the number of left parentheses at any state, and (2) the numbers of left and right parentheses are equal.

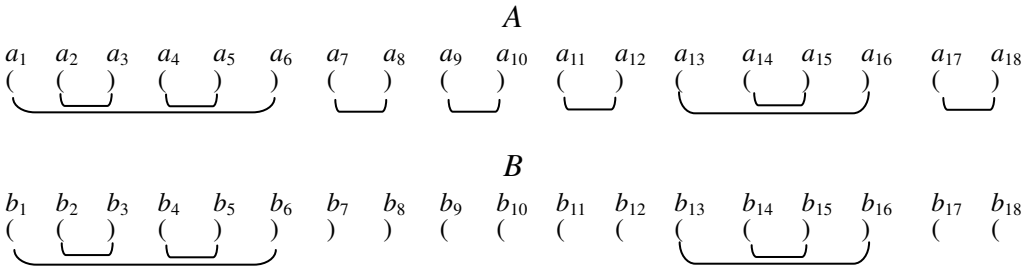


Figure 1. Two sequences of parentheses.

AutoText is a company, which is developing a text editor for programmers. The new editor will provide many powerful functions to automatically correct typing errors. On a keyboard, the left and right parentheses are adjacent. Thus, it is often that ")" is mistyped as "(" or vice versa. And therefore, one of the functions AutoText wants to provide is to automatically convert a sequence of parentheses P (that may not be well-formed) into a well-formed sequence P'. In the conversion, the only allowed operation is to *reverse* a parenthesis (i.e., either to replace a "(" with a ")" or to replace a ")" with a "("). For example, in Figure 1, we can convert B into the well-formed sequence A by performing 4 reverse operations on $b_7, b_{10}, b_{12}, b_{18}$. Of course, there may be several ways to convert a sequence into a well-formed sequence. A conversion is optimal if it uses the minimum number of reverse operations.

Please write a program to compute the minimum number of reverse operations that make a given sequence of parentheses $P = p_1p_2...p_n$ well-formed.

Input Format
The first line contains an integer $T \leq 10$ indicating the number of test cases. The first line of each test case contains an even integer n, $2 \leq n \leq 100$, indicating the length of P. Next, the second line gives the sequence P.

Output Format
For each test case, output the minimum number of reverse operations that make P well-formed.

Sample Input

```
3
18
(()()))((((((()((
2
()
8
(()))()(
```

Output for the Sample Input

```
4
0
2
```

Problem B
Linear Ecosystem
Time Limit: *1 Second*

An ecosystem is a community of living organisms in conjunction with the nonliving components of their environment (things like air, water and mineral soil), interacting as a system. A sustainable ecosystem is vital for human to immigrate to other planets. To simplify the matter, the concept of linear ecosystem is proposed. In a linear ecosystem living organisms and nonliving components are all called *comorg* and the number of comorgs in the system is a fixed constant, k. A unified measurement, *mass equivalence*, is developed to quantify the presence of each comorg. Therefore, the state of the ecosystem can be described as k-tuple with the values of the mass equivalence of the k comorgs. The whole ecosystem changes state at discrete time steps, that is the value of the mass equivalence of every comorg changes at the same time. And the mass equivalence of any comorg at next time step is determined by a linear combination of the mass equivalence of all comorgs at the current step. Therefore a linear ecosystem can be described as (C, E), where $C = \{c_1, c_2, ..., c_k\}$ is the set of comorgs, and $E = \{e_1, e_2, ..., e_k\}$ is the set of linear transition equations for these k comorgs, where $e_i = (\alpha_{i,1}, \alpha_{i,2}, ..., \alpha_{i,k})$. For example, the value of the mass equivalence of i^{th} comorg at time step $t + 1$, denoted by $c_i(t + 1)$, is determined by the following equation: $c_i(t + 1) = \sum_{j=1}^{k} \alpha_{i,j} c_j(t)$. A state of an ecosystem is the k-tuple $(c_1, c_2, ..., c_k)$. Note that we do not require c_i to be positive. We say that a state, s, is stable if the state after the transition is still s. Apparently the empty state, that is the state that all comorgs have zero mass equivalence, is always stable but meaningless. We say an ecosystem is potentially stable if there exist a non-empty stable state. Please write a program to decide if an linear ecosystem is potentially stable.

Technical Specification

1. The number of comorgs is at most 10.
2. $-100 < \alpha < 100$ and has at most two digits after the decimal point.
3. There are at most 20 test cases.

Input File Format
The first line of the input file contains an integer, n, which is the number of ecosystems. For each case, the first line contains the integer k which is the number of comorgs. Followed by k lines, where the i^{th} line contains,$\alpha_{i,1}, \alpha_{i,2}, ..., \alpha_{i,k}$, the coefficients of the transition equation for c_i.

Output Format
For each test case, output 1 if the ecosystem is potentially stable, otherwise output 0. Output only 5 answers per line. There should be a blank space between any two output answers.

Sample Input

```
6
2
4 -2
-6 5
2
2 2
0 0
3
0.3 0.2 0.5
0.4 0.4 0.2
0 0.8 0.2
3
0.3 0.2 0.5
0 0 0
0 0.8 0.2
2
4 2.0
-6 5
2
1 0
0 1
```

Output for the Sample Input

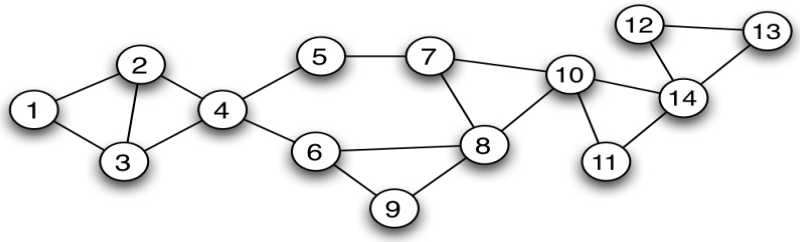
```
1 0 1 0 0
1
```

Problem C
Least Crucial Node
Time Limit: *1 Seconds*

A wireless sensor network is a self-organizing network without specific infrastructure. It is a multi-hop network in which sensor nodes can be randomly deployed and the data transmission between nodes usually involves other intermediate nodes. Sensor nodes are often deployed in outdoor or hazardous environments. Power outage of sensors can lead to node failure and cause many serious problem. For example, node failure can disconnect the whole network, causing data from one part of the network to fail to transmit to another part of the network. It is difficult to replace failed sensor nodes in hazardous conditions or far-reaching areas such as rainforests or high mountains.

Wireless sensor networks usually connect to the outside world through the so called sinks (may be regarded as gateways). All data collected by sensor nodes are sent to the sink, and then the sink relays such data to remote users or servers through the Internet, satellite, or any viable medium.

A wireless sensor network can be modeled by a graph with each vertex (edge) representing a sensor (a two-way communication link). In the following figure, nodes 4, 10, and 14 are more crucial to the connectivity of the network since a power shortage in any one of them can lead to a disconnected network.



Suppose that node 1 is the sink node. The failure of node 4 disconnects nodes in the set $\{5, 6, 7, 8, 9, 10, 11, 12, 13, 14\}$ from the sink. The failure of node 10 disconnects the nodes in the set $\{11, 12, 13, 14\}$ from the sink. Lastly, the failure of node 14 disconnects the nodes in $\{12, 13\}$ from the sink. This means node 4 is more crucial than nodes 10 and 14 in network connectivity; thus we call node 4 a *crucial node* of the network. Notice that crucial nodes do not include the sink. Specifically, the failure of a crucial node in a given sensor network disconnects the largest number of nodes from the sink. Moreover, the *least crucial node* is a crucial node with the least number label among all the crucial nodes. Please write a program to find the least crucial node for a given sensor network with a specific sink.

Input Format

There are several input lines to a test case. The first line of each test case contains an integer n ($2 \leq n \leq 100$), where n is the number of nodes (vertex set of G) labeled with $\{1, 2, \cdots, n\}$ in the network. The second line contains an integer, which is the label of the unique sink of the network. The third line contains an integer m, indicating the number of communication links in the network. The next m lines each contains a pair of integers, say

i and j (separated by a space), denoting there is a communication link (edge in E) between node i and node j. There are at most 10 test cases and $n = 0$ denotes end of the test cases.

Output Format

The output for each instance should contain an integer denoting the least crucial node in the given network.

Sample Input

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

Output for the Sample Input

```
3
2
```

Problem D
Discrete Logarithm Problem
Time Limit: *1 Second*

Finite groups are used in almost all modern cryptosystems. Let p be a prime. The finite multiplicative group constructed by integers modulo p is denoted by $\mathbf{Z}_p^*$. The elements of the group $\mathbf{Z}_p^*$ are $1, 2, \ldots, p-1$. Let a and b be two elements in $\mathbf{Z}_p^*$. The value of $a \times b$ is define as $a \cdot b \bmod p$. For example, let $p = 13$, $a = 5$, and $b = 7$. Then $5 \times 7 = 5 \cdot 7 \bmod 13 = 35 \bmod 13 = 9$.

You are going to write a program to solve the *discrete logarithm problem* in $\mathbf{Z}_p^*$. That is, given p, a, and b, compute x satisfying

$$a^x = \underbrace{a \times a \times \cdots \times a}_{x} = b.$$

For very large p, solving discrete logarithm problem in $\mathbf{Z}_p^*$ may not be esay. However, in this problem the value of p will not be very large.

Input File Format
The first line of the input file contains a prime p, $1 < p < 2^{13}$. This prime will be used in the following test cases. Each test case contains two integers a and b in a line. The last test case is followed by a line containing 0.

Output Format
For each test case, print out the solution x to the equation $a^x = b$ in the finite group $\mathbf{Z}_p^*$. If there are no solutions, print 0.

Sample Input

```
31
24 3
3 15
0
```

Output for the Sample Input

```
7
21
```

Problem E
Locating Facilities
Time Limit: *3 Seconds*

There is a network system which can be represented as a clique tree. A *clique tree* is an undirected graph in which every 2-connected component is a clique. A 2-*connected component* is a maximal subgraph satisfying the condition that the resulting subgraph is still connected after removing any vertex from the maximal subgraph. A *clique* is a subgraph in which there is an edge between any two vertices. For example, in Figure 1, there are nine 2-connected components which are the induced subgraphs of the vertices in sets $\{v_1, v_2, v_3, v_4, v_5\}$, $\{v_5, v_7\}$, $\{v_6, v_7\}$, $\{v_7, v_9\}$, $\{v_8, v_9, v_{12}, v_{13}\}$, $\{v_9, v_{10}\}$, $\{v_{10}, v_{11}\}$, $\{v_{10}, v_{14}\}$, and $\{v_{10}, v_{15}\}$, where an *induced subgraph* is a subset of the vertices of a graph together with any edges whose endpoints are both in this subgraph. Let $G = (V, E, w)$ be a clique tree in which every vertex u is associated with a positive weight $w(u)$ and V and E are the vertex and edge, respectively, sets of G.

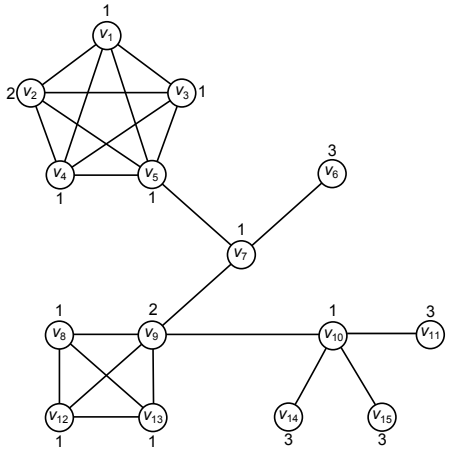


Figure 1: A clique tree $G = (V, E, w)$.

Note that each vertex in a clique tree contains a number of processors and there is an edge between two vertices if they can communicate directly with each other. The weight associated with each vertex is the number of processors in the vertex. For example, in Figure 1, $w(v_1) = 1$, $w(v_2) = 2$, $w(v_3) = 1$, etc. We say that p vertices are connected if the induced subgraph of those p vertices are connected. Let $d(u, v)$ denote the length of a shortest path between vertices u and v, i.e., the number of edges in the path, and $d_{\mathrm{w}}(u, v) = w(u)d(u, v)$.

We want to locate p facilities on p-connected vertices so that the communication cost of the network system is as small as possible, where each vertex can be located at most one facility. Let Q be a set of p-connected vertices with p facilities. The communication cost of a network system with respect to Q is defined as $\sum_{u \in V \setminus Q} d_{\mathrm{w}}(u, Q)$, where $d_{\mathrm{w}}(u, Q) = \min\{d_{\mathrm{w}}(u, v) | v \in Q\}$. Thus your task is to find a subset Q of V such that the communication cost of the network is minimized. For example, in Figure 1, if $p = 3$, then the communication cost of $Q = \{v_7, v_9, v_{10}\}$ is 26 which is minimum among all possible subsets of 3-connected vertices in V.

Technical Specification

- The number of vertices in a clique tree is at most 200.
- $1 \leqslant p \leqslant 10$.
- $1 \leqslant w(u) \leqslant 10$ for all $u \in V(G)$.

Input File Format

The first line of the input file contains an integer denoting the number of test cases. There are at most 10 test cases. In each test case, the first line contains two integers n and p, where n is the number of vertices in a clique tree and p is the number of facilities. Then the following n lines contain the weight (the first number) and the neighbors (the other numbers) of vertex v_i for $1 \leqslant i \leqslant n$.

Output Format

For each test case, output the minimum communication cost in a line.

Sample Input

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

```
1 9 11 14 15
3 10
1 8 9 13
1 8 9 12
3 10
3 10
```

Output for the Sample Input

```
46
26
```

Problem F
Decode Alien Message
Time Limit: *5 Seconds*

SETI (Search for Extra Terrestrial Intelligence) finally received messages from planet Kepler 452b ! Humans are not alone in the universe! The messages from Kepler 452b are several streams of codes. Through years of decoding efforts, scientists found that the purpose of streams is to explain the secrete of universe and instruct humans to create wormholes for inter-terrestrial travel.

The message said that the universe is a n-dimension timed event system. A stream of codes represents a one-dimension timed event flow. An example of one-dimension timed event flow is in Fig. 1.

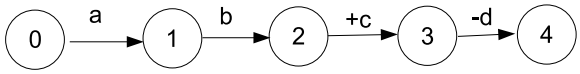


Figure 1: An example of timed event flow.

States are indexed by integers (i.e., circles in the figure), which are the states of space-time continuum. One symbol from $\{a, .., z, +a, .., +z, -a, .., -z\}$ represents an event that causes the timed event flow to change to next state. The scientist found that an event symbol can be polarized as positive and negative. The functions of polarized event symbols will be explained later.

A one-dimension timed event flow M_i is a tuple $(Q_i, s_i, \Sigma_i, \delta_i, e_i)$, where

- Q_i is a set of states, which is indexed by $0 - 999$,
- s_i is the initial state. $index(s) = 0$.
- Σ_i is a set of events, which is a symbol from $\{a, ..z, +a, .., +z, -a, .., -z\}$
- $\delta_i \subseteq Q_i \times Z_i \times Q_i$ is a set of state transitions in which $(q, w, q') \in \delta_i$ indicates that a state $q \in Q_i$ can change into $q' \in Q_i$ with an event symbol $w \in \Sigma_i$.
- The last state of a one-dimension timed event flow is called a final state e.
- Each state $q \in Q_i - \{e\}$, q has exactly only one transition to next state q' and $index(q) + 1 = index(q')$

The universe is $n-$dimension. Scientists found that a universe model must be composed from n one-dimension timed event flows. The composition is called *quantum-superposition*. That is, the Kepler aliens decomposed a universe model into n one-dimension timed event flows and sent these timed event flows to earth.

A brilliant scientist eventually solved the puzzle and discovered an algorithm to construct a universe model from n one-dimension timed event flows. A universe model U_n can be composed from n one-dimension timed event flow $M_i, i = 1..n$ is a tuple $(Q_1 \times Q_2 \times .. \times Q_n, (s_1, s_2, .., s_n), \Sigma - \{+a, .., +z, -a, .. - z\} \cup \{(a), ..(z)\}, \triangle, (e_0, e_1, .., e_n))$. $\triangle$ is the state transition function of U_n which can be computed by the following algorithm:

1. let $(q_1, q_2, .., q_n)$ be $(s_1, s_2, .., s_n)$
2. repeat until no more states can be generated.

2.1 for each (q_i, w, q_i') in M_i and $w \in \{a, .., z\}$, add $((q_1, ., q_i, ., q_n), w, (q_1, .., q_i', ..q_n))$ to $\triangle$.

2.2 for each $(q_i, +\alpha, q_i')$ in M_i and $(q_j, -\alpha, q_j')$ in M_j, $i \neq j$ and $\alpha \in \{a, .., z\}$, add $((q_1, ., q_i, .., q_j, .., q_n), \text{“}(\alpha)\text{”}, (q_1, .., q_i', .., q_j', .., q_n))$ to $\triangle$.

According to the algorithm, polarized event symbols,$\{+a, .., +z, -a, .., -z\}$, can no longer appear in the universe model U_n. A positive event symbol and a negative event symbol must match in pair (as in step 2.2) to appear as an (x), $x \in \{a, .., z\}$ in the universe model.

Fig. 2 is an example of 3 timed event flows. According to the algorithm, a 3-dimension universe model can be constructed as in Fig. 3.

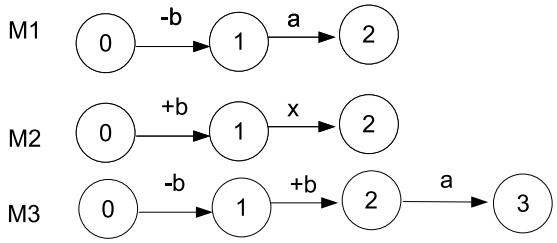


Figure 2: Three timed event flows.

If states are reachable, polarized event symbols, such as `+b` and `-b`, must match in pair and merge into a “`(b)`” symbol in the universe model.

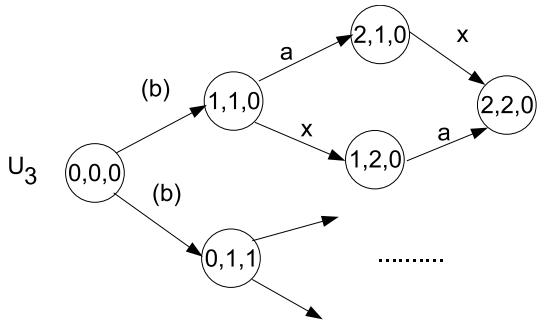


Figure 3: The universe model.

After the universe model was constructed, the brilliant scientist found that there are *“trapped”* universe states, such as $(2, 2, 0)$ in Fig. 3. It is not the final state ($(2, 2, 3)$ is the final state in this example) but no more states can be generated from this state. In physics meaning, Kelper aliens tried to teach humans to manipulate space-time to create black holes in a universe model. These black holes actually serve as wormholes so that future interstellar travel is possible.

Given n timed event flows, please compose the universe model and find the wormholes. There could be more than one wormhole in a universe model.

Input File Format

The input file begins with an integer T which is the number of test cases. Each test case begins with an integer $n, 2 \leq n \leq 20$ to indicate the number of timed event flows in the test case. Each timed event flow begins with an integer $k, 1 \leq k \leq 999$ which is the length of the timed event flow. The length of a timed event flow is the number of transitions. Following the number k is k event symbols from $\{a, .., z, +a, .., +z, -a, .., -z\}$ separated by space. While constructing the universe model, the initial state index starts from 0.

Output Format

For each wormhole state, please output its state index ($s_0\ s_1\ ..s_n$). These state index is separated by exactly one space. If there are more than one, please output the wormholes *in one line* in the order that $(a, b, c) \leq (x, y, z)$ if $((a \leq x) or (a = x, b \leq y) or (a = x, b = y, c \leq z)$ and each wormhole is separated by exactly one space. For example, if there are three wormholes, please output "(1 0 1) (2 3 5) (7 9 8)", in which a space is inserted between each wormhole. If no wormhole are found, please output "no wormholes".

Sample Input

```
2
3
2 -b a
2 +b x
3 -b +b  c
4
2 +b +a
2 -a -b
2 +a x
2 -a y
```

Output for the Sample Input

```
(2 2 0)
(0 0 2 2)
```

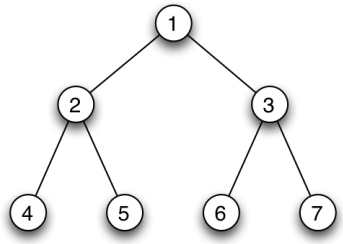
Problem G
Maximum Cut Order
Time Limit: *3 Seconds*

Max uses a tree $G(V, E)$ with undirected edges to model a network topology. Suppose that the vertex set $V = \{1, \ldots, n\}$ and each undirected edge $\{i, j\} \in E$ has an integral capacity $c(\{i, j\}) \geq 0$. For this problem, Max likes a specific tree, where the edge set is $E = \{\{i, \lfloor i/2 \rfloor\} : i = 2, \ldots, n\}$, that is for each vertex $i \in V$ with $i > 1$ there is an edge $\{i, \lfloor i/2 \rfloor\}$. For edge $e = \{i, j\} \in E$, the capacity of e is $c(e) = |i - j| \ \% \ m$, for some given positive integer m as the modulus, i.e., $c(e)$ is the remainder after $|i - j|$ is divided by m.

To maximize the communication throughput, Max wants to order the vertices in the sequence $v_1, \ldots, v_n$, which is a permutation of V, such that for all $i \in \{2, \ldots, n\}$ the *cut capacity* between v_i and $\{v_1, \ldots, v_{i-1}\}$ is maximized over all possible $v_j \in V - \{v_1, \ldots, v_{i-1}\}$. Max calls such order a *Maximum Cut Order*. The cut capacity between two disjoint sets $A, B \subseteq V$ is the sum of capacities of edges with one end vertex in A and the other in B. For convenience, define the edge subset between A and B as $E(A, B) = \{e : e = \{i, j\} \in E, i \in A, j \in B\}$. A *Maximum Cut Order* $v_1, v_2, \ldots, v_n$ satisfies that for all $i \in \{2, \ldots, n\}$:

$$\sum_{e \in E(\{v_1, \ldots, v_{i-1}\}, \{v_i\})} c(e) = \max_{j \in \{i, \ldots, n\}} \sum_{e \in E(\{v_1, \ldots, v_{i-1}\}, \{v_j\})} c(e).$$

If there is a tie when determining v_i, we choose the vertex with smaller index, i.e., following the lexicographical order.



For example, as in the above graph with $n = 7$, we have $V = \{1, 2, 3, 4, 5, 6, 7\}$ and $E = \{\{1, 2\}, \{1, 3\}, \{2, 4\}, \{2, 5\}, \{3, 6\}, \{3, 7\}\}$. Note that $c(\{i, j\}) = |i - j| \ \% \ 2$. Let the starting vertex v_1 be 1. Then the Maximum Cut Order is: 1, 2, 5, 3, 6, 4, 7. If $v_1 = 7$, then the order is: 7, 3, 6, 1, 2, 5, 4.

Your task is to write a program to find the Maximum Cut Order of $G(V, E)$ from a given starting vertex.

Technical Specification

1. k: the number of test case, $k \leq 10$.
2. n: the number of vertices, $2 \leq n \leq 500000$.
3. s: the starting vertex, $s \leq n$.
4. m: the capacity modulus, $2 \leq m \leq 10000$.

Input File Format

The first line of the input file contains an integer $k(\leq 10)$ indicating the number of test cases. Each test case consists of three integers n, s and m, separated with space(s), in a line, where $n(\leq 500000)$ indicates the number of vertices, $s(\leq n)$ indicates the starting vertex, and $m(\leq 10000)$ is the modulus for capacity. The capacity for an edge $e = \{i, j\}$ is simply $|i - j| \ \% \ m$.

Output Format

For each test case, output the Maximum Cut Order in a line, where the first element is the starting vertex. Separate adjacent vertices with a space.

Sample Input

```
3
2 2 5
7 1 2
7 7 2
```

Output for the Sample Input

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

Problem H
Separating Pebbles
Time Limit: *12 Seconds*

Dr. Y has travelled to an ancent ruin in which a large number of circular-shaped ('o') and cross-shaped ('+') pebbles are scatterd on an open field. Being a puzzle enthusiast, Dr. Y tried to figure out if she can draw one straight line so that circular-shaped pebbles and cross-shaped pebbles are on the opposite side of the line and no pebbles on the line. For example, in Fig. 1 (left), Dr. Y can draw a stright line to separate the two types of pebbles, whereas in Fig. 1 (right), it would not be possible. Please write a program to determine if a given group of pebbles can be separated by a single straght line.

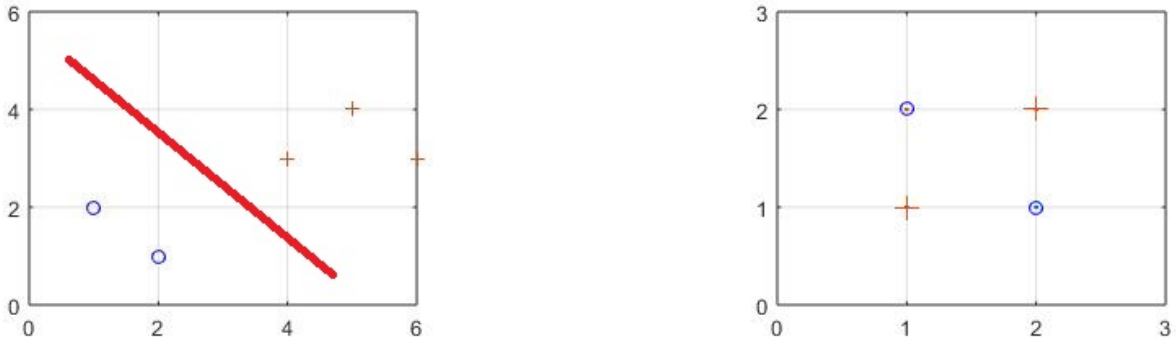


Figure 1: (left), pebbles can be separated by a straight line. (right), pebbles can not be separated by a straight line.

Input Format
The first line of the input contains an integer K ($K \leq 20$) indicating the number of test cases. The first line of each test case consists of an integer N ($N \leq 250$) indicating the number of pebbles. The next N lines each contains a triplet (x, y, c), where x and y represent the x and y coordinates (all integers, $0 \leq x, y \leq 150$) of a pebble point, and c represents the type of pebble: 'o' denoted by 0 and '+' denoted by 1.

Output Format
For each test case, output 1 if Dr. Y can separate the pebbles with a single straight line; if not, output 0.

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

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

Output for the Sample Input

```
1
0
```

Problem I
Social Network
Time Limit: *3 Seconds*

Raymond is a big data analyst. He constructs a social network G from n persons numbered $1, 2, \ldots, n$. For two groups of persons G_1 and G_2, he utilizes the following two rules to construct the network:

Rule 1: If two persons $p_1 \in G_1$ and $p_2 \in G_2$ have a common interest, then G_1 and G_2 can be merged into one group by connecting every person in G_1 to all the persons in G_2. This construction is called the *join* operation, denoted by $G_1 \bigotimes G_2$. For each person $p_1 \in G_1$ and $p_2 \in G_2$, we say that p_1 and p_2 have a connecting relation after applying Rule 1.

Rule 2: If any two persons $p_1 \in G_1$ and $p_2 \in G_2$ have no common interest, then no connecting relation is created between G_1 and G_2. This construction is called the *union* operation, denoted by $G_1 \bigcup G_2$.

According to the above two rules, Raymond constructs a social network using the following recursive manner:

(1) A social network can contain only one person;

(2) If G_1 and G_2 are social networks, then $G = G_1 \bigotimes G_2$ is a social network;

(3) If G_1 and G_2 are social networks, then $G = G_1 \bigcup G_2$ is a social network.

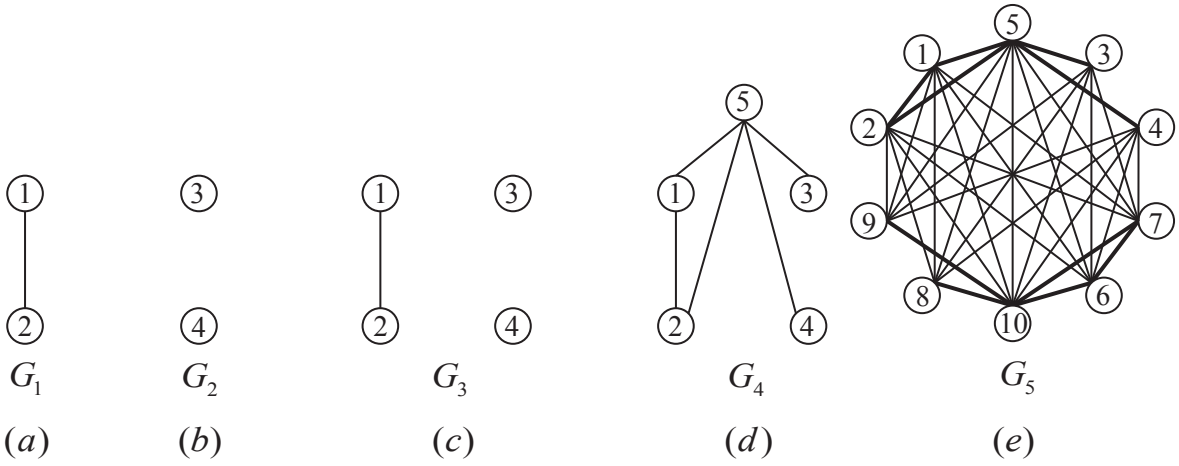


Figure 1: The construction of a social network

For ease of input, we use "J" to represent the join operation $\bigotimes$, and use "U" to represent the union operation $\bigcup$. A social network can be represented by a construction expression. For example, consider the following construction expression

$$((((1)\text{J}(2))\text{U}((3)\text{U}(4)))\text{J}(5))\text{J}((((6)\text{J}(7))\text{U}((8)\text{U}(9)))\text{J}(10)).$$

The corresponding social network can be constructed as follows. In Step 1, $G_1 = (1)\text{J}(2)$ and $G_2 = (3)\text{U}(4)$ are built (Figure 1(a) and Figure 1(b)). In Step 2, $G_3 = ((1)\text{J}(2))\text{U}((3)\text{U}(4))$

is built by applying the union operation to G_1 and G_2 (Figure 1(c)). In Step 3, G_4 = (((1)J(2))U((3)U(4)))J(5) is built by applying the join operation to G_3 and Person 5 (Figure 1(d)). In Step 4, G'_4 = (((6)J(7))U((8)U(9)))J(10) is built, where G'_4 has the same structure with that of G_4.

In Step 5, G_5 = (((((1)J(2))U((3)U(4)))J(5))J(((((6)J(7))U((8)U(9)))J(10)) is built by applying the join operation to G_4 and G'_4 (Figure 1(e)).

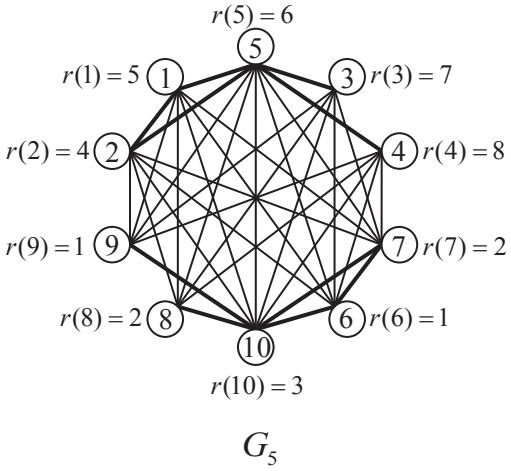


Figure 2: An optimal assignment of a social network

After constructing the network, Raymond would like to assign all the persons in the network to nature numbers (this weighting is used for some special survey). For each person i, we use $r(i)$ to denote the number assigned to i. The assignment must satisfy the following property that for **every** path between two persons, i and j, having the same number (i.e., $r(i) = r(j)$) in the network, there exists at least one person k (on that path) whose number is higher than $r(i)$ (=$r(j)$), that is $r(k) > r(i) = r(j)$. An assignment is *optimal* if the largest used number is the smallest among all assignments. Moreover, the largest used number in an optimal assignment is called the *assignment number*. Figure 2 illustrates an optimal assignment of the social network shown in Figure 1(e). The assignment number equals 8.

Given a construction expression corresponding to a social network G, your task is to write a computer program to compute the assignment number of G.

Technical Specification

1. $2 \leq n \leq 10000$.
2. The number of operations is exactly $n - 1$, which constructs one social network from n persons.

Input File Format

The first line of the input file contains an integer indicating the number of test cases to follow. Each test case consists of two lines: the first line contains an integer n and the second line contains a construction expression, where the construction expression consists of

digits, alphabets J and U, and parentheses indicating the order of construction. Formally, a construction expression, denoted by *Exp*, can be recursively defined as follows: *Exp* can contain only one positive integer i, where $1 \leq i \leq n$. For each *Exp* containing more than one positive integer, it has the form: *Exp*= (*Exp*)J(*Exp*) or *Exp*= (*Exp*)U(*Exp*). Note that the most outer part of construction expression will not enclosed by redundant parentheses. For each test case, each number between 1 and n appears exactly once in the construction expression.

Output Format

The output contains one line for each test case. Each line contains the assignment number of the corresponding social network.

Sample Input

```
2
10
((((1)J(2))U((3)U(4)))J(5))J((((6)J(7))U((8)U(9)))J(10))
10
((((1)J(2))J((3)J(4)))J(5))J((((6)J(7))J((8)J(9)))J(10))
```

Output for the Sample Input

```
8
10
```

Problem J
Task Review
Time Limit: *5 Seconds*

To design the programming tasks for 2055 ACM ICPC Taipei site, there is a committee of N members. Each member has designed a task, and thus there are N tasks in total. For the quality of the game, the tasks must be reviewed before they can be used. Now, the question is how to assign the review work. Greg, as the chair of the committee, decides that an assignment is *feasible* if and only if each task is reviewed by one member and each member reviews exactly one task. Of course, any one cannot review his/her own task. To find a good review assignment, Greg models it as the following optimization problem.

The members are labelled from 0 to $N-1$. The task designed by member i is called task i. Greg creates a directed graph $G = (V, E, w)$, where $V = \{0, 1, \ldots, N-1\}$ denotes the members, as well as the tasks. If member i can review task j, then there is an edge (arc) from i to j with weight $w(i, j)$, where the weight is the difficulty of this review. For a directed graph $G = (V, E, w)$, an edge subset F is a *cycle cover* if these edges form some disjoint cycles and cover all the vertices, i.e., each vertex has in-degree and out-degree one in the subgraph induced by F. Thus, each feasible assignment corresponds to one cycle cover.

For example, Figure 3(a) shows all possible review edges: member 0 can review task 1 with difficulty 4; member 0 can review task 2 with difficulty 8; member 1 can review task 0 with difficulty 5; and so on. The assignments in Figure 3(b) is not feasible since it is not a cycle cover, and both assignments in (c) and (d) are feasible.

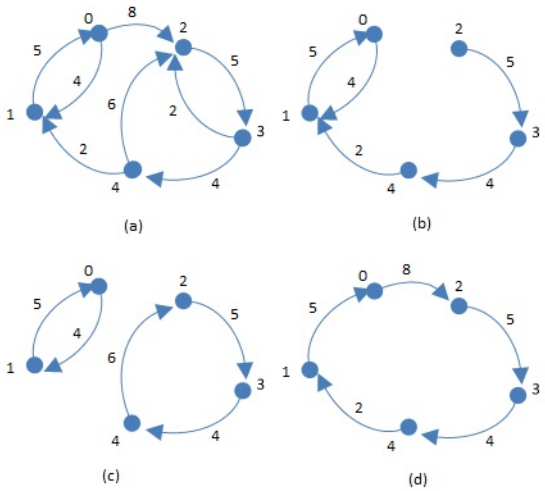


Figure 3: Examples of review assignments.

However, there may be more than one feasible assignment. If the difficulties are too small, the members feel boring. But if the difficulties are too large, the loads are heavy. For a set of edges F, let $\max(F) = \max_{e \in F} w(e)$ and $\min(F) = \min_{e \in F} w(e)$. To find an assignment to make the committee happy, Greg defines the following measures. For a cycle cover C, the load is defined by

$$L(C) = \sum_{e \in E, w(e) \leq \max(C)} w(e);$$

the boringness is defined by

$$B(C) = \sum_{e \in E, w(e) \geq \min(C)} w(e);$$

and the unhappyness is given by

$$S(C) = \max\{L(C), B(C)\}.$$

The goal of this problem is to find a cycle cover with minimum unhappyness. Let C and D be the cycle covers in Figure 3(c) and (d), respectively. We have $\max(C) = 6$, $\min(C) = 4$, $\max(D) = 8$, and $\min(D) = 2$. Thus, $L(C) = 2 + 2 + 4 + 4 + 5 + 5 + 6 = 28$, $B(C) = 4+4+5+5+6+8 = 32$, and therefore $S(C) = 32$. For D, we have $L(D) = B(D) = S(D) = 36$. So C has a smaller unhappyness and thus is better than D.

Technical Specification

1. The number of vertices (committee member) N is between 2 and 500.
2. The difficulty for each task review, i.e. the edge weight $w(i, j)$, is between 1 and 10000.

Input File Format

The first line of the input file contains an integer T, $1 \leq T \leq 8$, which indicates the number of test cases. The first line of each test case contains two integer N and M, where M is the number of edges. The following M lines contain the data of edges, one line for one edge. The data of each edge consists of three integer i, j, and $w(i, j)$. Any two consecutive numbers in the same line are separated by a space.

Output Format

For each test case, output the minimum unhappyness in one line. If there is no feasible assignment, then output -1.

Sample Input

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

Output for the Sample Input

```
32
-1
```

Problem K
Robots
Time Limit: *8 Seconds*

Write a program to collect data from robots. We are given two sets of robots $X = \{X_1, \ldots, X_m\}$, $Y = \{Y_1, \ldots, Y_n\}$, and a base B. Each robot has a data and we would like to compute the sum of data from all robots and deliver it to the base. In order to do so a robot can send its data to another robot or the base with the following constraints.

- A robot can only send its data to one destination (a robot or the base) at a time.
- A robot (or the base) can receive data from one robot at a time.
- The base can not send data to anyone.
- A robot in X can complete sending its data in x seconds. A robot in Y can complete sending its data in y seconds.

The robots and the base can perform addition, so we can collect the final sum at the base. That is, we assume that after receiving a data, a robot or the base can perform an addition with zero time. Now let us illustrate this concept by an example. Let us consider a system with one robot X_1 in X and two robots Y_1 and Y_2 in Y. We also assume that x is 1 and y is 10. At the beginning Y_1 can send its data to Y_2 and X_1 can send its data to the base. After 1 second the base will know the data of X_1. However, only after 10 seconds Y_2 will have the data of Y_1, add its own data, and send the sum to the base. After 20 seconds the base receives the sum of data from Y_1 and Y_2, adds the data from X_1, and has the final sum. The entire summation will take 20 seconds.

Now let us try a different schedule. At beginning Y_1 sends data to the base, and Y_2 sends data to X_1, and both can complete after 10 seconds. Finally X_1 starts sending the sum of data from Y_2 and itself to the base after 10 seconds, and the entire summation can finish in 11 seconds.

Now given m, n (the numbers of robots in X and Y), x, and y, please determine the minimum number of seconds to finish the summation.

Constraints

- $1 \leq x < y \leq 1000$.
- $0 \leq m < 1200$.
- $0 \leq n < 500$.

Input Format

The input consists of multiple test cases. First line contains a single integer t indicating the number of test cases to follow. Each of the next t lines contains four integers – x, y, m, n.

Output Format

For each test case, output on a single line the minimum number of seconds to sum up all numbers from all robots.

Sample Input

```
1
1 10 1 2
```

Output for the Sample Input

```
11
```

Problem L
Reward the Troop
Time Limit: *2 Seconds*

Long long time ago, there was an ancient country with a strong army. The soldiers and all the high ranking officers formed one of the most powerful troops. Besides having a high sense of honor, they organized the army in a very regular pattern. There were m military ranks in total. Each higher ranking officer led n soldiers of a lower rank. Figure 1 shows the three layers of ranks. Each higher ranking officer is leading two lower ranking soldiers.

After each victorious battle, the king would award all the soldiers certain titles to inspire them. The soldiers were also looking forward to the titles. To be honored by unique and individual titles in the large units seemed to be more important, to all the soldiers, than what the titles really were. They wish to have each soldier titled differently from other soldiers who are direct subordinates to the upper two layers and the lower two layers. Also, everyone's title should differ from the one in his same rank from the same direct commander. In order to satisfy the needs, the king and his advisers discussed a variety of titles, such as "most courageous", "kill the most", "fastest dispatch" and so on. Everyone was conferred on an individual title with different amounts of golds.

Due to the great amount of commanders and soldiers, the king wished to title each commander and his subordinate soldier diversely in the state of saving the expense of the exchequer. Take the 3-layer ranking in Figure 1 for example, the general gets title 4 since he cannot have the same title as any soldier in the lower two ranks, while his left subordinate one gets title 3 and his right subordinate one gets title 2 in Figure 2 because they should not have the same title. Meanwhile, the left commander's subordinate officers were conferred on titles 1 and 2 while the right commander's subordinate officers were titled 1 and 3 which are distinct from their direct commander. Therefore, the total amount of the gold to these 7 soldiers in Figure 2 is $4 + 3 + 2 + 2 + 1 + 1 + 3 = 16$. The gold of the exchequer was the most efficiently distributed in this way. Please help the king find the least amount of gold needed to reward the troop.

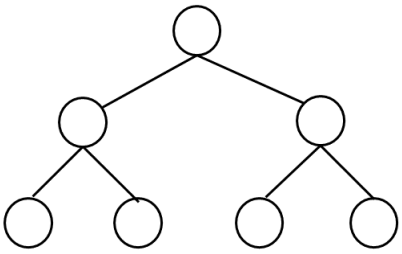


Figure 1: Troop with 3 ranks.

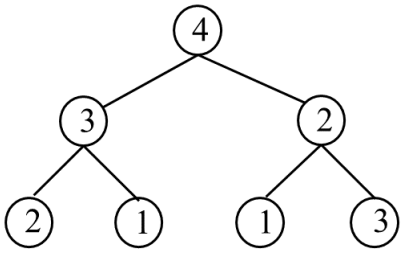


Figure 2: Reward for the troop.

Technical Specification

- The number of military ranks is bounded by 1000, i.e., $1 \leq m \leq 1000$.
- Each commander led at most 100 sergeants of the next rank, i.e., $1 \leq n \leq 100$.

- There are total of $n^0 + n^1 + \ldots + n^{m-1}$ military in the troop.

Input Format

There are at most 10 test cases. The first line is the number of test cases. Each test case contains two integers m and n separated by a space which are the numbers of military ranks and the number of soldiers led by each general respectively.

Output Format

For each test case, output the minimum amount of gold (module by 10000000) needed to reward the troop.

Sample Input

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

Output for the Sample Input

```
16
66
1
```