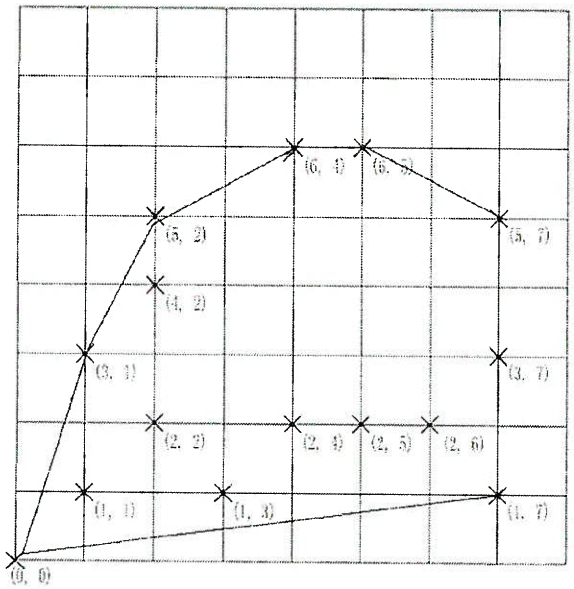
Problem A
Minimal Fence Length

Input file: pa.txt

Problem Statement

A landlord partitions his land into a 100 by 100 grid. He randomly plants trees on the intersection points of the grid within his land. Lately he wants to build a fence to enclose all the trees. Given those trees, your job is to calculate the minimal length of the fence to encompass them. A sample figure is as follows. The minimal fence is then formed by the tree in positions (0,0), (3,1), (5,2), (6,4), (6,5),(5,7), (3,7), and (1,7), with length 21.9412.



Input File Format

The first line of input file consists of a single number denoting the number of test cases in the file. There is a single line containing a '/' character separating two consecutive test cases. The end of the file is marked with a line containing a '.' character. Each test case starts with a positive integer n ($n \leq 30$) representing the number of trees, followed by n ordered pairs of row and column numbers (separated by a blank space) to represent the positions of the trees. The order of trees that appears in the input data set is random.

Output Format

For each test case, print out all of the following on a single line separated by blanks: The test case number i, the string "The minimal length of the fence is ", and your computed minimal length of the fence to 4 decimal places. Accuracy within 0.001 of the correct answer will be accepted as correct.

Sample Input

```
1
15
0 0
1 3
1 1
2 2
3 1
4 2
5 2
6 4
6 5
5 7
2 5
2 4
2 6
3 7
1 7
.
```

Sample Output

```
Case 1: The minimal length of the fence is 21.9412
```

Problem B
Graded Reasoning of Ordinal Conditional Formulas

Input file: pb.txt

Problem Statement

Ordinal conditional functions (OCF) are employed as a kind of knowledge representation formalism in the so-called κ calculus. In the calculus, knowledge is represented as graded implications of the form $(p \rightarrow q, v)$ where p and q are called events and $v \geqq 0$ is a non-negative integer. The intuitive meaning of the implication $(p \rightarrow q, v)$ is that $p \rightarrow q$ (i.e., p implies q) is believed with certainty 2^{-v}. Given a set of graded implications, other graded implications not explicitly appearing in the set can be derived by a chaining rule. Assume that n events $p_1, p_2, \ldots, p_n$ and a set of implications $S=\{(p_i \rightarrow p_j, v_{ij}) \mid i \neq j\}$ are given, then an implication $(p \rightarrow q, v)$ is derivable from S, denoted by $S\alpha\ (p \rightarrow q, v)$, if one of the following conditions holds:

(i) $p=q$ (i.e, p and q are the same event) and $v=0$;

(ii) $(p \rightarrow q, v) \in S$;

(iii) for some $v' \leqq v$: $S\alpha\ (p \rightarrow q, v')$;

(iv) for some event r and non-negative integers v_1 and v_2:
$S\alpha\ (p \rightarrow r, v_1)$, $S\alpha\ (r \rightarrow q, v_2)$, and $v_1+v_2=v$.

The implication degree, $D_S(p, q)$, between two events p and q with respect to the set of graded implications S is defined as the smallest integer v such that $(p \rightarrow q, v)$ is derivable from S. Mathematically, this is defined as

$$D_S(p, q) =_{dcf} \min \{v \mid S\alpha\ (p \rightarrow q, v)\},$$

where if $\{v \mid S\alpha\ (p \rightarrow q, v)\}$ is an empty set, then $D_S(p, q)$ is defined as ∞.

Therefore, our problem is to compute $D_S(p_l, p_m)$ from the number of events n, the set of graded implications S, and two events p_l and p_m .

Input File Format

The first line of input file consists of a single number denoting the number of test cases in the file. There is a single line containing a '/' character separating two consecutive test cases. The end of the file is marked with a line containing a '.' character. For each of the test cases, the first line contains a positive integer n ($n \leq 20$) denoting the number of events and the second line contains a pair of positive integers l and m, separated by a blank space. After the second line, there are several lines that form the set of graded implications S. Each line of S contains a triplet i, j, and v, each separated by a blank space, corresponding to $(p_i \rightarrow p_j, v)$, where $1 \leqq i \neq j \leqq n$ and v is a non-negative integer.

Output Format

For each test case of the input, compute the value $D_S(p_l, p_m)$ and output it in a line. If $D_S(p_l, p_m) = \infty$, then output a special character * .

Sample input:

```
3
3
1 2
1 2 5
1 2 4
1 3 2
3 2 4
/
3
2 3
1 2 5
1 2 4
1 3 2
3 2 4
/
4
1 2
1 3 1
1 2 8
1 4 5
2 3 10
3 2 7
4 2 1
3 4 0
.
```

Sample output:

```
4
*
2
```

$S\alpha$ $(p_1 \rightarrow p_2, 5)$ since $(p_1 \rightarrow p_2, 5) \in S$; **Analysis of Case 1**

$S\alpha$ $(p_1 \rightarrow p_2, 4)$ since $(p_1 \rightarrow p_2, 4) \in S$;

$S\alpha$ $(p_1 \rightarrow p_2, 6)$ since $S\alpha$ $(p_1 \rightarrow p_3, 2)$ and $S\alpha$ $(p_3 \rightarrow p_2, 4)$;

So the minimal value is 4

Problem C
Largest All-One Submatrix

Input file: pc.txt

Notation and Definitions

For any set S of integers, let $|S|$ denote the number of elements in S. For example, if $S=\{1,3,5\}$, then $|S|=3$. Let A be an $m\times n$ matrix. For any indices i and j with $i\in\{1, 2, \ldots, m\}$ and $j\in\{1, 2, \ldots, n\}$, let $A(i, j)$ denote the element of A in row i and column j. For any index sets I and J with $I \subseteq \{1, 2, \ldots, m\}$ and $J \subseteq \{1, 2, \ldots, n\}$, let $A(I, J)$ denote the submatrix of A consisting of $A(i, j)$ for each pair i and j of indices with $i\in I$ and $j\in J$. We say that $A(I, J)$ is an <u>*all-one*</u> submatrix of A if $A(i, j) = 1$ holds for each pair i and j of indices with $i\in I$ and $j\in J$.

Problem Statement

Suppose A is an input $m\times n$ <u>*binary*</u> matrix. That is, for each $i\in\{1, 2, \ldots, m\}$ and $j\in\{1, 2, \ldots, n\}$, we have $A(i, j)\in\{0, 1\}$. You are asked to output an all-one submatrix of A with maximum size.

Example

$$A = \begin{bmatrix} 0 & 1 & 0 & 0 & 1 & 0 & 1 \\ 0 & 0 & 1 & 0 & 0 & 1 & 1 \\ 0 & 1 & 0 & 0 & 1 & 1 & 1 \\ 0 & 1 & 1 & 0 & 0 & 1 & 0 \\ 1 & 1 & 0 & 1 & 1 & 0 & 1 \end{bmatrix}$$ is a 5×7 binary matrix. Clearly, $A(\{1,2,3\},\{5,6,7\})$ is *not* an all-one submatrix of A, since $A(2,5) = A(1,6) = 0$. It is also clear that $A(\{2,4\},\{3,6\})$ is an all-one submatrix of A of size $2\times2 = 4$; $A(\{5\},\{1,2,4,5,7\})$ is an all-one submatrix of A of size $1\times5 = 5$; and $A(\{1,3,5\},\{2,5,7\})$ is an all-one submatrix of A with size $3\times3 = 9$. It is not difficult to verify that $A(\{1,3,5\},\{2,5,7\})$ is the only all-one submatrix of A with size at least 9.

Constraints

To simplify the problem, you may assume that $1\le m,n\le 80$ and that the number of non-zero entries in each row of A is at most 15.

Input format

The first line of input file consists of a single number denoting the number of test cases in the file. There is a single line containing a '/' character separating two consecutive test cases. The end of the file is marked with a line containing a '.'

character. For each test case, the first line has an integer specify number of rows m in this test case. For the next m lines, each line contains no more than 15 integers, each separated by a blank space character. Each number j in the i-th row specifies that $A(i, j) = 1$. You may also assume that the numbers in each line are monotonically increasing.

Output Format

Please output two sets I and J of indices such that $A(I, J)$ is an all-one submatrix of A with maximum size. Your program should output a single line of numbers, separated by space characters in the following order: (a) the numbers in I in increasing order, (b) a zero, and (c) the numbers in J in increasing order. If there is more than one all-one submatrix of A with maximum size, then output the solution with the least alphabetical order.

Sample Input

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

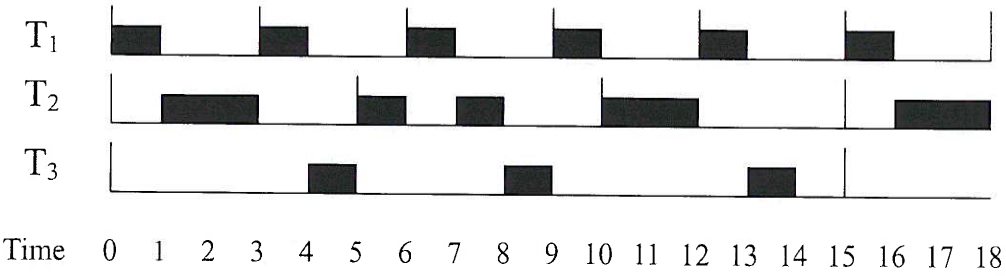
Sample Output

```
1 3 5 0 2 5 7
```

Problem D
A Preemptive EDF Scheduler
Input file: pd.txt

Problem Statement

Priority-driven scheduling approach is commonly used in modern computer operating systems to execute a set of tasks where the systems always execute the task with the highest priority. Earliest Deadline First (EDF) scheduling approach assigns a higher priority for a task with earlier deadline where the tasks are periodic and preemptive. We are interested in the earliest time when all tasks have finished at least once. Note that it would be easier and faster to simulate the problem using two priority queues, one for the ready and preempted tasks, and the other for the rest.



For example, in the above figure, tasks T_1, T_2, T_3 with execution time 1, 2, 3 and periods 3, 5, 15 respectively are feasible using the preemptive EDF scheduler. T_1, T_2, and T_3 first finish execution at time 1, 3, and 14 respectively. The whole schedule repeats at 15. T_2 is preempted by T_1 at time 6 and resume at time 7, T_3 is preempted by T_2 at time 5 and resume at time 8, preempted again by task T_1 at time 9 and resume at time 13.

Definitions and Constraints

1. A task T_2 is preempted by a task T_1 means T_1, with a higher priority, replaces T_2, with a lower priority, to run the CPU, and T_2 will resume later.
2. A ready task is a task ready to run as long as it gets the right to use CPU.
3. A periodic task is executed exactly once in every constant interval, called period.
4. For simplicity, a periodic task is ready at the beginning of each period and its deadline is at the end of each period.
5. A set of periodic tasks is feasible if every task finishes execution before its deadline.
6. All the input numbers are positive integers and are less than or equal to 500000.
7. The number of tasks in each task set is less than or equal to 10.
8. Each execution time is less than or equal to its period.

9. The periods are not sorted and are all different in a task set.
10. All the test task sets are feasible.

Input File Format

The first line of input file consists of a single number denoting the number of test cases in the file. There is a single line containing a '/' character separating two consecutive test cases. The end of the file is marked with a line containing a '.' character. For each test case, the first line gives the number of tasks, *n*, followed by *n* lines of blank space separated task execution time and period pairs.

Output Format

For each test cases, print on a single line the earliest time when all tasks will have finished at least once.

Sample Input

```
3
3
1 3
2 5
3 15
/
2
1 2
1 3
/
3
1 3
2 15
2 5
.
```

Sample Output

```
14
2
9
```

Problem E
Min-Max List Partitioning
Input file: pe.txt

Problem Statement

Let $L=<a_1, a_2, \ldots, a_n>$ be an *ordered* list of n integers $a_1, a_2, \ldots, a_n$. An ordered list $L'=< b_1, b_2, \ldots, b_m>$ is a *sub-list* of L if $1 \le m \le n$ holds and there exists an integer i $(1 \le i \le n - m + 1)$ such that for all j=1, 2,..., m, we have $b_j=a_{i+j-1}$. Let p and q be two integers with $1 \le p \le n, 1 \le q \le n$, and $p \times q \ge n$. A *(p,q)-way partitioning* of L is to divide L into p *non-overlapping* sub-lists

$$<a_1, a_2, \ldots, a_{m_1}>,$$

$$<a_{m_1+1}, a_{m_1+2}, \ldots, a_{m_1+m_2}>,$$

.

.

.

$$<a_{m_1+m_2+\ldots+m_{p-1}+1}, a_{m_1+m_2+\ldots+m_{p-1}+2}, \ldots, a_{m_1+m_2+\ldots+m_{p-1}+m_p}>$$

such that $m_1+m_2+\ldots+m_p=n$ and each sub-list has at most q numbers (i.e., for all j=1, 2,.., p, $m_j \le q$). The *value* of a sub-list is the summation of the numbers in the sub-list. The *cost* of a (p,q)-way partitioning is the *maximum* of the values over the p sub-lists. A (p,q)-way partitioning is *optimal* if its cost is *minimum* among all possible (p,q)-way partitionings. The min-max list partitioning problem is to find the cost of an optimal (p,q)-way partitioning.

Example: Suppose L=<1, 2, 4>, p=2, q=2. L has six possible sub-lists <1>, <2>, <4>, <1, 2>, <2, 4>, <1, 2, 4>. There are two (2,2)-way partitionings S_1={<1>, <2,4>} and S_2={<1,2>, <4>}. The cost of S_1 is 6 and the cost of S_2 is 4. S_2 is an optimal partitioning, but S_1 is not.

Input File Format

The first line of an input file consists of a single number denoting the number of test cases in the file. There is a single line containing a '/' character separating two consecutive test cases. The end of the file is marked with a line containing a '.' character. For each test case, the first three lines specify n, p, and q, respectively, with $n \le 4000$. The next n lines give an ordered list L of n integers, one on each line.

Output Format

For each test case, print on a single line the cost of an optimal (p, q)-way partitioning.

Sample Input

```
2
3
2
2
1
2
4
/
4
2
3
1
3
5
7
.
```

Sample Output

```
4
9
```

Problem F
Optimal Selection Problem

Input file: pf.txt

Problem Statement

Consider a set of items, where each item has an integer value, which may be positive, negative or zero. We want to select some of the items such that the sum of values is maximized. There is no restriction on the number of items to be selected. However, for each item we associate with a group of items, such that when this item is selected, each of the group members must be selected, too. Note that groups may be empty. Given a set of items, their values and the associated group items that must be selected together, you must write a program to calculate the maximum value that can be achieved and list the selected items in lexicographic order. Your program should be able to handle up to 250 items.

Input File Format

The first line of input file consists of a single number denoting the number of test cases in the file. There is a single line containing a '/' character separating two consecutive test cases. The end of the file is marked with a line containing a '.' character. For each test case, the first line has an integer *n*, denoting number of items in this test case. On the next *n* lines, each line shows the information of an item with the format: item name, item value, and followed by the group members that must be selected together with, all are separated by blank characters.

Output File Format

For each test case, output a single line showing maximum value that can be achieved and the selected items in lexicographic order.

Sample Input

```
1
5
A 3 B C
B -4 C
C 1 D
D 2 E
E -2
.
```

Sample Output

```
Maximum value:  1.  The selected items: C D E
```

Problem H
Dr. Mad Scientist
Input file: ph.txt

Problem Statement

Dr. Mad Scientist invented a device that can predict the performance distribution of an athlete when the detailed data about the athlete is given. This device can solve the inter-galaxy traveling problem the Universe Olympic Committee is facing by using the device to predict the performance of each athlete in different galaxy. You are asked to write a program to report the top three athletes whose chances of winning the championship are the greatest. Given an athlete's vital information, the devise will output a sequence of score and probability pairs. For example, A 0.10 30 0.25 40 0.10 50 0.55 80 means that the athlete A will score 30 with probability 0.10, 40 with probability 0.25, 50 with probability 0.10 and 80 with probability 0.55. Assume that the athletes' performances are independent from each other. Given a sequence of each individual athlete's performance prediction, we can compute the probability of winning for each individual athlete by considering all the possible contest outcomes. For example, given two athletes with the following scores and probabilities:

```
A 0.1 37 0.9 60
B 0.3 45 0.7 60
```

There are 4 possible scenarios to be considered:

A scores 37 while B scores 45 → B wins with probability of 0.1*0.3 = 0.03
A scores 37 while B scores 60 → B wins with probability of 0.1*0.7 = 0.07
A scores 60 while B scores 45 → A wins with probability of 0.9*0.3 = 0.27
A scores 60 while B scores 60 → A wins with probability of 0.9*0.7 = 0.63

Therefore, A will win with probability 0.27+0.63=0.9 and B will win with probability 0.03+0.07=0.1. Note that in the 4th scenario above, although the scores for A and B are both 60, as a rule of thumb, since A appears first in the input list, A is given the nod as the winner. In general, given m athletes, with s_1, s_2, …, s_m possible scores for each athletes, respectively, there are $s_1 \times s_2 \times \ldots \times s_m$ outcome scenarios to consider.

Input File Format

The first line of input file consists of a single number denoting the number of test cases in the file. There is a single line containing a '/' character separating two consecutive test cases. The end of the file is marked with a line containing a '.' character. For each test case, the first line contains an integer m ($m \leq 10$) denoting the number of athletes. In the following m lines, an athletes ID and at least one

probability-score pair is given. The ID, probabilities, and scores are all separated by blank spaces. Please note that ID each ID is a character string with at most four characters, each score is an integer between 0 and 100, each probability is a floating point number between 0 and 1 with at most two digits after the decimal point.

Output Format

For each tournament, output the IDs of the 3 athletes who have the highest winning probabilities, in order of decreasing probability. If there is a tie, the athlete that appears first in the input list should output first.

Sample Input

```
3
4
Abe 0.4 91 0.04 84 0.31 35 0.25 13
Bob 0.05 99 0.22 96 0.54 68 0.04 58 0.15 24
C 0.05 81 0.08 80 0.15 72 0.72 30
Dave 0.18 88 0.06 80 0.44 78 0.09 69 0.23 19
/
4
Able 0.06 65 0.22 59 0.21 40 0.13 27 0.38 22
Blob 0.54 74 0.02 56 0.38 30 0.06 20
Cat 0.21 88 0.47 77 0.12 62 0.2 28
Dog 0.13 80 0.45 77 0.4 21 0.02 16
/
4
Air 0.03 93 0.66 64 0.25 29 0.06 6
Boy 0.1 98 0.07 95 0.34 42 0.49 24
Coke 0.22 85 0.27 73 0.1 71 0.2 62 0.21 5
Zoo 0.15 91 0.15 82 0.16 51 0.19 30 0.35 19
.
```

Sample output

```
Bob Abe Dave
Cat Dog Blob
Coke Zoo Air
```