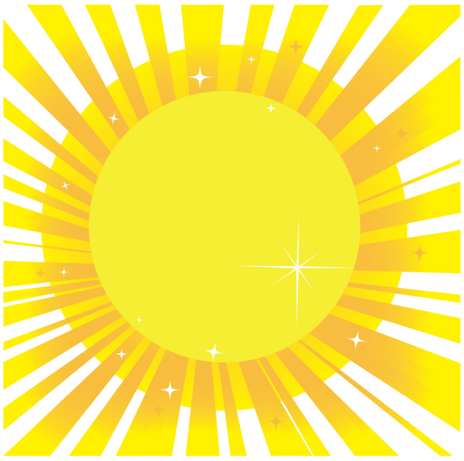
COMPUTER AND INFORMATION SCIENCE

UNIVERSITY OF OREGON

Twentieth Annual University of Oregon Eugene Luks Programming Competition

2016 April 30, Saturday
am10:00 – pm2:00



Problem Contributors
James Allen, Eugene Luks, Chris Wilson, and the ACM

Technical Assistance and Organization
Paul Bloch, Adriane Bolliger, Lauradel Collins

Magic Multiples

The Elvish races of Middle Earth believed that certain numbers were more significant than others. When using a particular quantity *N* of metal to forge a particular sword, they believed that sword would be most powerful if the thickness *K* were chosen according to the following rule:

Given a nonnegative integer *N*, what is the smallest K such that the decimal representations of the integers in the sequence:

N, 2N, 3N, …, KN

contain all ten digits (0 through 9) at least once?

Lord Elrond of Rivendell has commissioned you with the task to develop an algorithm to find the optimal thickness *(K)* for any given quantity of metal *(N)*.

Input will consist of a single integer *N* per line. The end of input will be signaled by end of file. The input integer will be between 1 and 200,000,000, inclusive.

The output will consist of a single integer per line, indicating the value of *K* needed such that every digit from 0 through 9 is seen at least once.

Sample Input
```
1
10
123456789
3141592
```

Sample Output
```
10
9
3
5
```

***note**: problem courtesy of the 2012 ACM Pac NW Regional Competition*

In a **Ranked Pairs voting** system, each voter ranks the candidates in order according to the voter's preference, so for example, a voter might fill out the ballot as on the right. Here, the voter indicated they would most like to see Kasich elected and least like to see Trump elected. The ballots are then run through the following processes to select the winner:

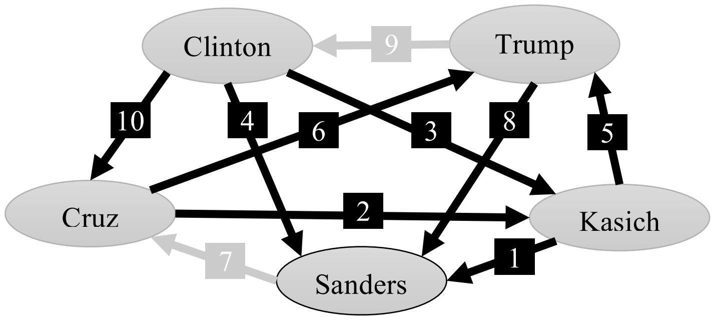
Ballot	
Clinton	3
Cruz	4
Kasich	1
Sanders	2
Trump	5

1. Compute the outcome of every possible race between pairs of candidates
2. Sort pairwise races in descending order of winning margin. For example, if the vote is 43% to 57%, then the winning margin is 14%,
3. Create a digraph whose vertices are the candidate and the directed edges are from the loser to the winner of each pairwise match. A pairwise tie does not create an edge. Only add the edges that do not create a cycle, when considering the edges in descending order.
4. The overall winner is the sink of the digraph (see back).

For example, in the sample ballot, Clinton beats Cruz; Kasich beats Cruz; Sanders beats Cruz; Cruz beats Trump; Kasich beats Clinton; Sanders beats Clinton; Clinton beats Trump; Kasich beats Sanders; Kasich beats Trump; and Sanders beats Trump. With millions of ballots, these would be expressed as percent of voters. For example, suppose all the ballots combined lead to the grid of percents on the right.

	Cruz	Clinton	Kasich	Sanders	Trump
Cruz	—	57%	4%	61%	24%
Clinton	43%	—	5%	12%	58%
Kasich	96%	95%	—	2%	23%
Sanders	39%	88%	98%	—	60%
Trump	76%	42%	77%	40%	—

In this grid, we would read that 57% prefer Cruz to Clinton. Notice that we only have to give the upper triangular portion, because we can infer that 43% prefer Clinton to Cruz. This would generate the digraph on the right, where the edges are numbered in the order in which they were considered. The black edges are part of the digraph, but the grey edges were not added to the digraph because they would have created cycles.



The first line of input will consist of a single number between 1 and 1000 indicating the number of elections that follow. Each election will consist of two lines. The first line consists of the number of candidates between 1 and 1000 followed by a space and a space-delimited list of the candidates sirnames. The sirnames in each election consist of upper and lowercase letters and are guaranteed to be unique in that election. The second line is the integer percents to fill into the upper triangular matrix of pair-wise races where the names are listed along the top and side in the order given in the first line. Your output should consist of one line for each election that lists the winner.

Sample Input	Sample Output
3 5 Cruz Clinton Kasich Sanders Trump 57 4 61 24 5 12 58 2 23 60 4 Larry Curly Moe Shemp 94 32 18 76 5 87 3 Patty Laverne Maxine 40 87 14	Sanders Moe Patty

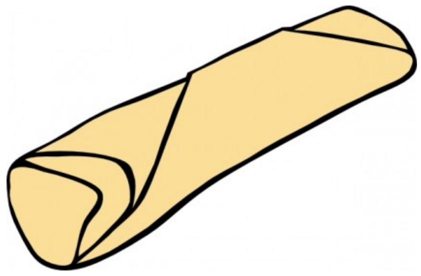
Ranked Pair Voting Almost Always Produces a Winner

To see that a finite DAG has at least one sink, pick a random vertex. If it is not a sink, then follow an edge out to a new vertex. Continue following edges as long as you can. The last vertex you visit is a sink. Since it is a DAG, you cannot revisit vertices you have already visited. Since there are only finitely many vertices, the process must halt.

Since the ranked pair method creates a finite DAG, it has at least one sink. Suppose there are more that one sink. Select sinks A and B. Without loss of generality, assume A beats B. Since B is a sink, there cannot be an edge from B to A. Thus, at the time the edge from B to A was considered, it was disqualified for creating a cycle. This means there is a path from A to B, but this means there must be an edge leading out of A. This contradicts the choice of A as a sink.

A pairwise tie may cause us not to consider the edge from B to A in the argument above, resulting in two sinks and therefore, no winner. Also, ranked pair voting does not produce a winner if two pairwise race results are equal and adding both edges would create a cycle, but adding either edge alone won't, and the choice of edge to delete generates a different outcome. Either of these is less likely than a tie in a traditional voting system. The test input to this problem contains no such situations.

CHIPOTLE [UDATED DESCRIPTION]



You are a successful executive of a tech startup, and you want to bring your coding team out for a nice lunch at the nearby Chipotle restaurant. Because you are a great lover of efficiency, you want to spend all of your money and to do so in a way that maximizes the calorie count for your hungry programmers.

For example, suppose you have $100.00 to spend, and the menu looks like the following.

cost	*cal*
$1.39	150
$4.39	600
$8.99	550
$6.27	800

Then you would be able to obtain a maximum of 12800 calories and spend *exactly* $100.00. Note that you could obtain more calories (13550) by spending less money ($99.85), but that is another question. Here we **must** spend exactly $100.00.

The input will start with a number C, C ≤10, of test cases. Each test case will start with a float T, T≤ 9000.00, the target amount to spend. The next number M, M ≤ 50, is the number of menu items. The next M lines contain a pair V W, where V ≤ 20.00 is a float giving the cost of a menu item and W≤ 1000 is the calorie count of one order of that menu item. (All floats represent dollars and cents, so there are only two digits to the right of the decimal.)

The output should consist one integer for each test case, that being the maximum number of calories obtainable by spending exactly V. If it is not possible to spend exactly V, then that line should be 0.

Sample Input

```
3
100.00
4
1.39 150
4.39 600
8.99 550
6.27 800
1.00
1
1.39 150
1.00
1
0.89 100
```

Sample Output

```
12800
0
0
```

FIBONACCI SMOOTHIES

You are surely familiar with the Fibonacci sequence: 0, 1, 1, 2, 3, 5, 8, 13, 21, 34, 55, 89, 144, ...

which is defined by

$$\mathcal{F}(0)=0,\ \mathcal{F}(1)=1, \text{ and } \mathcal{F}(n) = \mathcal{F}(n\text{-}1) + \mathcal{F}(n\text{-}2) \text{ for } n\geq 2.$$

An alternate, non-recursive characterization,

$$\begin{pmatrix} 0 & 1 \\ 1 & 1 \end{pmatrix}^n = \begin{pmatrix} \mathcal{F}(n-1) & \mathcal{F}(n) \\ \mathcal{F}(n) & \mathcal{F}(n+1) \end{pmatrix} \quad \text{for } n \geq 1.$$

may help in appreciating the exponential growth of $\mathcal{F}(n)$ with n. In fact, it is not hard to show that $\mathcal{F}(n) \approx \tau^n/\sqrt{5}$ where $\tau = (\sqrt{5}+1)/2$, the *Golden Section*. So, for example, $\mathcal{F}(10^{15})$ has over 209 trillion digits.

Fortunately, the output for this problem will not involve the full value of $\mathcal{F}(n)$.

A concept from computational number theory: An integer is called *smooth* if it factors completely into "small" primes. Here, we will consider only the single-digit primes

2, 3, 5, and 7

to be small. Any integer n has a maximum smooth divisor $\mathcal{S}(n)$, thus the integer $n/\mathcal{S}(n)$ is not divisible by 2, 3, 5, or 7. Examples:

$$\mathcal{S}(5) = 5,\quad \mathcal{S}(13) = 1,\quad \mathcal{S}(34) = 2,\quad \mathcal{S}(144) = 144,\quad \mathcal{S}(377) = 1,\quad \mathcal{S}(610) = 10.$$

In this problem, you will be given values for n and you are to determine $\mathcal{S}(\mathcal{F}(n))$ in each case. Even though $\mathcal{F}(n)$ grows exponentially, you may take comfort from the fact that $\mathcal{S}(\mathcal{F}(n)) < 100n$ for all n.

The first line of the input will declare the number N of problem instances to be considered. Each of the next N lines will contain a positive integer $n \leq 10^{15}$.

Sample Input
```
6
5
7
9
12
14
15
```

Sample Output
```
5
1
2
144
1
10
```

DIAMONDS

A diamond's overall worth is determined by its mass in carats as well as its overall clarity. A large diamond with many imperfections is not worth as much as a smaller, flawless diamond. The overall clarity of a diamond can be described on a scale from 0.0–10.0 adopted by the American Gem Society, where 0.0 represents a flawless diamond and 10.0 represents an imperfect diamond.

Given a sequence of *N* diamonds, each with weight, w_i, in carats and clarity, c_i, on the scale described above, find the longest subsequence of diamonds for which the weight and clarity are both becoming strictly more favorable to a buyer.

In the following sequence of diamonds,

w_i	c_i
1.5	9.0
2.0	2.0
2.5	6.0
3.0	5.0
4.0	2.0
10.0	5.5

the longest desirable subsequence is

1.5	9.0
2.5	6.0
3.0	5.0
4.0	2.0

Input begins with a line with a single integer *T*, $1 \leq T \leq 100$, indicating the number of test cases. Each test case begins with a line with a single integer *N*, $1 \leq N \leq 200$, indicating the number of diamonds. Next follow *N* lines with 2 real numbers w_i and c_i, $0.0 \leq w_i, c_i \leq 10.0$, indicating the weight in carats and the clarity of diamond *i*, respectively.

Output for each test case a single line with the length of the longest desirable subsequence of diamonds.

(see over for sample data)

Sample Input

```
3
2
1.0 1.0
1.5 0.0
3
1.0 1.0
1.0 1.0
1.0 1.0
6
1.5 9.0
2.0 2.0
2.5 6.0
3.0 5.0
4.0 2.0
10.0 5.5
```

Sample Output

```
2
1
4
```

***note**: problem courtesy of the 2014 ACM Pac NW Regional Competition*