Jimmy's Perfect Vacation

Filename: vacation

Jimmy and his family are planning a vacation. They all love amusement parks, and they always try to go on every ride that the park has. Unfortunately, they only have a certain amount of time in the day. Not only does timing make things complicated, some of the paths between two rides have been blocked off for parades, construction, and mothers tending to crying babies. To make matters even worse, Jimmy is very sensitive to rides and throws up every time he gets off, so he never wants to go near a ride he has already ridden. They could write out by hand to figure out the fastest way to visit each ride, but Jimmy has a good friend (you)! Help Jimmy and his family find the fastest way to visit all of the rides. Of course, they will be going on a lot more vacations, so your program should be able to handle multiple vacations.

The Problem:

Given each ride the amusement park has, as well as a list of blocked paths between two rides, find the fastest time it will take for Jimmy and his family to visit each one. Assume that he and his family walk at 1 meter per second, and they can always travel to their destination in a straight line assuming that the path is not blocked. Rides always take a constant time of 120 seconds per ride (they always visit the parks during the off-season so the waiting time is negligible).

The Input:

Input will begin with a single integer, n, which is the number of different parks Jimmy and his family have planned to visit. For each park, there will be two non-negative integers, r and b, which are the number of rides and blocked paths respectively ($r < 11$, $b < r^2$). On the following r lines are two integers, x and y, denoting the Cartesian coordinates of the ride (in meters). The rides are numbered from 1 to r for simplicity, and no two rides will have the same coordinates. After these r lines are b lines, each giving two integers i and j stating that the path between ride i and ride j is blocked, and therefore, the path from j to i is blocked as well. All coordinates will be non-negative integers less than 1,000, and Jimmy and his family always start their visit at the entrance, which is at the origin, (0,0). There will never be a blocked path that includes the entrance, and you may never revisit the entrance (they consider that exiting the park and make you buy another ticket!). Assume there is a path between every pair of rides unless it was blocked.

The Output:

For each vacation, first output a header, on a line by itself, stating "`Vacation #`k`:`" where k is the number of the vacation, starting with 1. If Jimmy can and his family can ride every ride, output a new line "`Jimmy can finish all of the rides in `s` seconds.`" where s is the number of seconds rounded to the nearest thousandth of a second it takes to ride every ride (as an example of rounding, a time of 100.5455 would be rounded up to 100.546, whereas a time of 100.5454 would be rounded down to 100.545). If Jimmy and his family cannot ride every ride, output one line saying "`Jimmy should plan this vacation a different day. `" instead. Output a single blank line after the output for each vacation.

Sample Input:

```
3
4 0
0 2
2 2
2 4
4 4
5 4
10 10
12 35
64 60
3 7
100 857
1 2
1 3
1 4
1 5
5 2
0 5
0 10
0 20
0 50
0 25
3 4
1 2
```

Sample Output:

```
Vacation #1:
Jimmy can finish all of the rides in 488.000 seconds.

Vacation #2:
Jimmy should plan this vacation a different day.

Vacation #3:
Jimmy can finish all of the rides in 670.000 seconds.
```

Taco Bell Combos

Filename: tacobell

The Problem

Taco Bell offers many delicious items! You've been elected to buy Taco Bell for your group of friends. To maximize everyone's satisfaction, you've decided to get all unique items. Given the number of items you want to buy and a list of all the items currently available at Taco Bell, make a list of each combination of items of the proper size that you can buy.

The Input

The first line of the input file will contain a single positive integer, *T (T < 100)*, representing the number of Taco Bell runs to evaluate. The first line of each test case will have to space-separated positive integers, *N (N ≤ 10)*, and *K (K ≤ N)*, representing the number of distinct items on Taco Bell's menu currently and the number of items you are supposed to buy, respectively. The following *N* lines will each contain one string representing an item for that Taco Bell run. Each string will consist only of lowercase letters and be in between 1 and 20 characters long.

The Output

For each input case, output a single line for each possible combination of items. Each combination should be listed in alphabetical order. The order of the combinations should be in lexicographical order. When comparing two different combinations, find the first corresponding pair of items that don't match. Whichever item comes first between the two corresponds to the combination that should be listed first. In printing each line, print a space after each item in the combination, including the last one. Follow the output of each Taco Bell run with a blank line.

Sample Input

```
2
3 2
taco
burrito
nachos
4 4
chalupa
softshelltaco
gordita
pizza
```

Sample Output

```
burrito nachos
burrito taco
nachos taco

chalupa gordita pizza softshelltaco
```

Programming Contractor

Filename: contractor

The Problem

After graduating from UCF with your Computer Science degree, you've decided that you'd like to work for yourself, instead of some big corporation. In starting your business, you find that various companies want to outsource jobs that you can do. For each job, you've become very good at determining the number of days it will take you to finish it. Naturally, each of these jobs comes with a fixed amount of compensation, regardless of how long the work takes. Due to your superior education, you may receive more offers for jobs than you can take. Given the number of days you are willing to work in the year, write a program to determine the maximal amount of money you can make if you accept the appropriate set of jobs.

The Input

The first line of the input file will contain a single positive integer, c ($c \leq 1000$), representing the number of input cases to analyze. The first line of each input case will have two space separated positive integers, n ($n \leq 20$), and d ($d \leq 365$), representing the number of job offers you've received for the year and the number of days you are willing to work during the year, respectively. The next n lines will each contain two space separated integers, d_i ($d_i \leq 365$) and p_i ($p_i \leq 1000000$), representing the number of days and amount of payment, respectively, you would receive if you accepted the i^{th} job.

The Output

For each input case, output a single integer representing the maximal amount of money in dollars you can make by taking a set of the possible jobs that you can finish within the number of days given (or fewer days).

Sample Input

```
2
2 5
3 10000
4 8000
3 100
20 20000
40 50000
40 30000
```

Sample Output

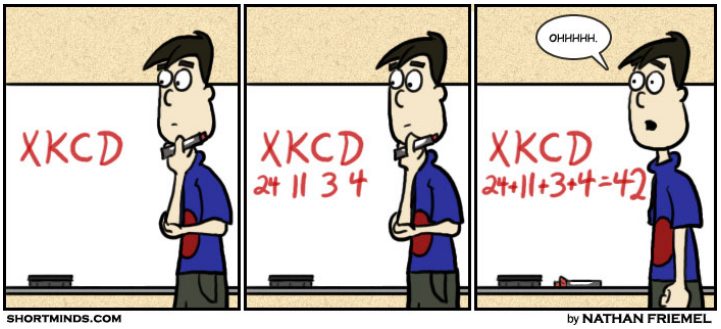
```
10000
100000
```

XKCD

Filename: `xkcd`

Matt is a fan of the web comic XKCD. Being such a fan he would like to come up with an online name similar to xkcd (he wants to keep his lower-case). Your job will be to help Matt come up with a name similar to xkcd.

Looking at this name it can be seen that taking the sum of the ordinal's of each letter in the word totals to 42 or, as we know, the answer to life, the universe and everything. Ordinals are defined as the 1-based index for each letter in the alphabet (a=1, b=2, c=3, …, z=26).



Matt also remembered watching an interview with Randall Munroe (the creator of xkcd) and learning one of the requirements of the online name he chose for himself: "xkcd" (xkcd the comic was later given this same name). Randall required that his name be non-pronounceable. Therefore, Matt would also like his name to be non-pronounceable as well. To ensure this, Matt would like to choose a name that does not have any vowels in it. To be extra careful he would like to include the letter "y" as a vowel.

Lastly, Matt noticed that the values of each of the ordinals of the letters in "xkcd" are strictly decreasing if you swap the last two letters (i.e., xkcd → xkdc where x=24 > k=11 > d=4 > c=3). To make his name even more similar to xkcd, Matt would like his name to have a strictly decreasing order when the last letters are swapped.

Given these rules, Matt would like to see what names he could possibly choose from a list of names that meet these requirements. Even though he is a huge fan of XKCD, he still wants to be unique, however. He has decided to have one major difference in his chosen name. He would like to be able to look at names of different lengths that meet his requirements. Your job is to write a program that takes in a length desired and prints out all the names of that length in alphabetical order that meet his XKCD-like requirements.

The Problem:

Given a name with a particular length, print out all names of that same length whose ordinals for each character add up to 42, are strictly decreasing when the last letters are swapped and are non-pronounceable (meaning that it contains no vowels including "y").

The Input:

The input will be lines containing single positive integers, n ($2 \leq n \leq 42$), representing the length of the names that you must print out that are "xkcd-like." The number 42 indicates end of input.

The Output:

For each length requested in the input, output "`XKCD-like name(s) of length: n`" where *n* is the length given in the input. On the following lines, output all names (in alphabetical order) that are XKCD-like that are also in the universe of *n* letter strings. If there are no such strings for the input length, output the phrase "`Mostly Harmless`" instead. Note that the end of input marker of 42 should not be processed normally; instead, you should output the phrase "`The answer to life, the universe and everything!`" and end.

Sample Input:

```
2
6
28
42
```

Sample Output:

```
XKCD-like name(s) of length: 2
pz
rx
sw
tv
XKCD-like name(s) of length: 6
kjhfcd
kjhgbd
ljgfcd
ljhfbd
ljhgbc
lkgfbd
lkhfbc
lkjdbc
mjgfbd
mjhfbc
mkgfbc
mlhdbc
nhgfcd
njgfbc
nkhdbc
nlgdbc
nmfdbc
phgfbc
pjgdbc
pkfdbc
qjfdbc
rhgdbc
shfdbc
tgfdbc
XKCD-like name(s) of length: 28
Mostly Harmless
The answer to life, the universe and everything!
```

Problem C: N Digit Prefix Composites

We define an n digit prefix composite as follows: A number with n digits (no leading digit 0 allowed) such that the prefix of k digits of the number is divisible by k. Thus, 14125 is a 5^{th} degree prefix composite because 1 is divisible by 1, 14 is divisible by 2, 141 is divisible by 3, 1412 is divisible by 4 and 14125 is divisible by 5.

The Problem

Given an input value n, print a list of all n digit prefix composites in numerical order.

The Input

The first line of the input file will contain a single positive integer, *T (T < 10)*, representing the number of test cases. Each test case will be on a line by itself, with a single integer, *N (N < 10)*, the input size for that case.

The Output

For each case, output each prefix composite of the desired length, in numerical order, one per line.

Sample Input

```
1
1
```

Sample Output

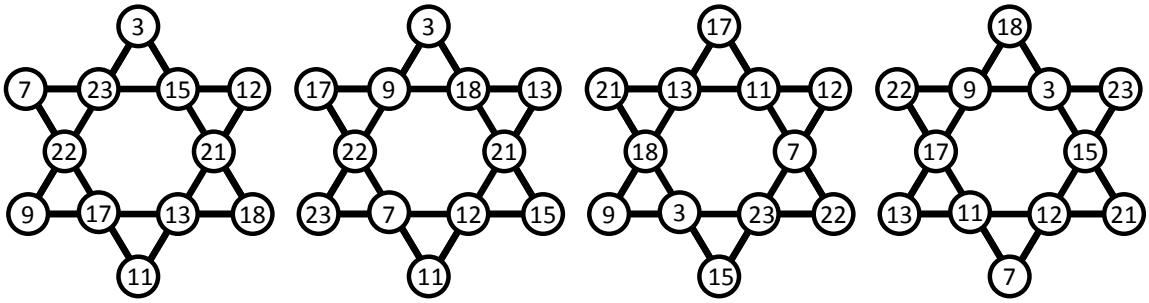
```
1
2
3
4
5
6
7
8
9
```

B: Hexagram

A Hexagram is a 6-pointed star, sometimes called the Star of David. Given these numbers:

```
3 17 15 18 11 22 12 23 21 7 9 13
```

There are four unique ways of assigning the numbers to vertices of the hexagram such that all of the sets of four numbers along the lines have the same sum (57 in this case). All other ways may be obtained from these by rotation and/or reflection.



Given 12 distinct numbers, in how many ways, disregarding rotations and reflections, can you assign the numbers to the vertices such that the sum of the numbers along each of 6 straight lines passing through 4 vertices is the same?

The Input

There will be several test cases in the input. Each test case will consist of twelve unique positive integers on a single line, with single spaces separating them. All of the numbers will be less than 1,000,000. The input will end with a line with twelve 0s.

The Output

For each test case, output the number of ways the numbers can be assigned to vertices such that the sum along each line of the hexagram is the same. Put each answer on its own line. Output no extra spaces, and do not separate answers with blank lines.

Sample Input

```
3 17 15 18 11 22 12 23 21 7 9 13
1 2 3 4 5 6 7 8 9 10 11 13
0 0 0 0 0 0 0 0 0 0 0 0
```

Sample Output

```
4
0
```