UCF Programming Team Practice

September 20, 2014

Developmental Team

Individual Problem Set

Problem Name	**Filename**
Bunnies	bunnies
Magic Multiple	magic
Mines	mines
Reverse Nonogram	nonogram
A No-Win Situation	nowin
Toga Party	party
Taco Bell Combos	tacobell
Upwards	upwards
XKCD	xkcd

Bunnies

Filename: bunnies

The Problem

Jill, the new lab assistant has made new cages for the lab rabbits, Peter and Cottontail. Unfortunately, these new cages are more like mazes with walls in every direction. Help Arup determine if the rabbits are in the same section of the cage or different sections. Each rabbit can hop one square directly up, down, right or left. They can hop as many times as they want from one free square to another. Each rabbit must stay on the grid.

The Input

Each input will start with a single integer T ($1 \le T \le 100$) on the first line. The number T will denote the number of test cases that follow. Each test case will begin with two integers, R and C ($1 \le R, C \le 10$) separated by a space. Each of the next R lines will contain C characters of either '_','#', 'P' or 'C' (Quotes for clarity). This will form a grid that represents the cage. A '_' represents a cell free of obstructions, '#' represents a wall, 'P' is Peter's location and 'C' is Cottontail's location. Each grid is guaranteed to have one and only one P and C.

The Output

For each test case output a single line containing "yes" if Peter and Cottontail are in the same section of the cage and "no" if they are not in the same section of the cage.

Sample Input

```
4
2 2
P#
#C
2 2
P_
C_
8 7
__P____
####_##
_____#_
_____#C
##_###_
_____#_
___#_#_
___#___
5 7
__P____
####_##
_____#_
_____#C
##_###_
```

Sample Output

```
no
yes
yes
no
```

Problem B — limit 5 seconds

Magic Multiple

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:

$$n, \quad 2n, \quad 3n, \quad 4n, \quad 5n, \quad \ldots, \quad kn$$

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

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.

Output

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	Sample Output

`1`
`10`
`123456789`
`3141592`
	`10`
	`9`
	`3`
	`5`

Mines

Filename: `mines`

The Problem:

Acme Mining Co. has been in the mining business for a long time now, and they are one of the biggest in the world. However they are getting worried with all these new companies starting to use computer simulations to determine how much underground volume a blast will clear, they don't want them to gain an unfair competitive advantage, and that is where you come in! Help Acme Mining Co. write a computer simulation of the volume a blast will clear given information about the surrounding structures and spaces.

The simulation will work as follows. The spaces in the mine will be represented as 1's and 0's. A 1 represents solid rock and a 0 represents empty space. When a blast is detonated it will blast away solid rock, but will not do anything in empty space. In this sense, once a blast is detonated it will continue to spread throughout the solid rock of the mine until it hits a pocket(s) of empty space. The blast can only move up, down, left, right, forward, and backward (no diagonals). If a blast is detonated in empty space it does not clear any volume. Each space represents one cubic foot of volume.

The Input:

The first number, *n*, in the input file will be a non-negative integer denoting the dimension of a cubic underground space. Following this will be a single line containing all the info for which three-dimensional coordinates (*x, y, z*) are rock and which are empty space. Valid coordinates for the structure will range from 0 to *n-1* for all three dimensions. The coordinates are ordered in lexicographical order. For example, for a 3 x 3 x 3 space: (0, 0, 0) is the first value, (0, 0, 1) is the next value, (0, 0, 2) is next, (0, 1, 0) is next, ... and the last coordinate would be (2, 2, 2).

Following this is another non-negative integer *t*, which is the number of simulations that will follow. On each of the following *t* lines will be 3 non-negative integers representing the coordinates (*x, y, z*) at which the blast is detonated.

The Output File:

For each simulation, print out a line with the following format:

```
Simulation #k, volume cleared is C cubic feet.
```

where k is the number of the simulation, starting with 1 and C is the number of cubic feet cleared by the blast. Follow the output for each simulation with a blank line.

Sample Input:

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

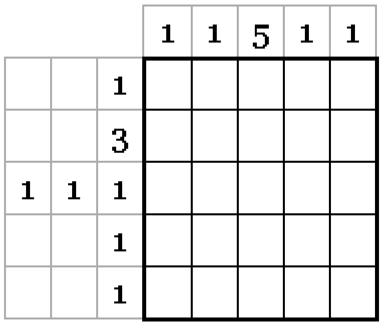
Sample Output:

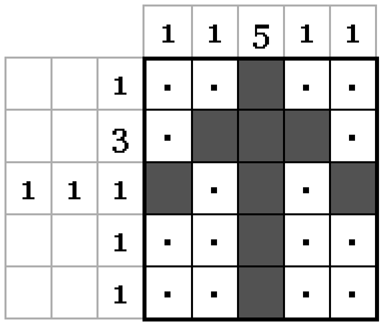
```
Simulation #1, volume cleared is 16 cubic feet.

Simulation #2, volume cleared is 0 cubic feet.
```

G: Reverse Nonogram

A Nonogram is a pencil puzzle played on a grid. The grid is initially blank. There are numbers on the side and top of the grid, which indicate how the grid squares should be filled in. The numbers measure how many unbroken lines of filled-in squares there are in any given row or column. For example, a clue of "4 8 3" would mean there are sets of four, eight, and three filled squares, in that order, with at least one blank square between successive groups. Here is a small example, with its solution.





You are going to work backwards. Given a Nonogram solution, produce the numbers which should be at the side and top of the grid.

Input

There will be several test cases in the input. Each test case will begin with an integer ***n*** (2≤***n***≤100) indicating the size of the grid. Each of the next ***n*** lines will have exactly ***n*** characters, consisting of either '**.**' for a blank square, or '**X**' for a square which has been filled in. The input will end with a line with a single **0**.

Output

For each test case, print 2***n*** lines of output. The first ***n*** lines represent the numbers for the rows, from top to bottom. The next ***n*** lines represent the numbers across the top, from left to right. If any row or column has no squares filled in, output a **0**. Put a single space between numbers on the same line. Do not output any lines with leading or trailing blanks. Do not output blank lines between any lines of output.

Sample Input	Sample Output

```
3
XXX
.XX
.X.
3
X.X
..X
X..
5
..X..
.XXX.
X.X.X
..X..
..X..
0
```
	```
	3
	2
	1
	1
	3
	2
	1 1
	1
	1
	1 1
	0
	2
	1
	3
	1 1 1
	1
	1
	1
	1
	5
	1
	1
	```

G: A No-Win Situation

Consider a simple variation of the card game Blackjack. In this game, a single player plays against the dealer. The game uses a standard deck of cards, where numbered cards are worth the number of points on the card for the cards numbered 2 to 10, 10 points for the face cards (King, Queen, and Jack) and either 1 or 11 points for the Aces.

The dealer deals the first card in the deck to the player, the second to the dealer, the third to the player, and the fourth to the dealer. The player then may continue to draw cards until s/he decides that the total is as close as possible to 21 and stops voluntarily or until s/he goes over 21. If the player goes over 21, the player loses. Then the dealer must draw cards until s/he reaches 17 or more points (with aces counting as 11 when possible). If the dealer goes over 21, the dealer loses. If neither of them goes over 21, the winner is the one who comes as close as possible to 21. If the player and the dealer have the same total, the player wins.

For example, suppose the first cards in the deck are Queen, 6, 4, 9, and 10. On the initial deal, the player will receive Queen and 4 (for a total of 14) and the dealer will receive 6 and 9 (for a total of 15). If the player does not select a card, the dealer will have to draw (because the dealer's total is less than 17) and will draw the 10, going over, so the player will win. But if the player draws a card (the 10), the player's total will be 24, so the player will lose.

In some situations, it is impossible for the player to win. Consider the case when the cards in the deck are: 10, 3, 4, King, 3, 5. The player will be dealt the cards 10 and 4. The dealer will have 3 and King. The table below illustrates what happens for each number of cards the player might draw:

Cards drawn	Player's hand (Points)	Dealer's hand (Points)
0	10, 4 (14)	3, King, 3, 5 (21)
1	10, 4, 3 (17)	3, King, 5 (18)
2	10, 4, 3, 5 (22)	3, King (13)

No matter how many cards the player draws, the player cannot win.

In this problem, you will analyze decks to determine if they lead to a situation in which the player cannot win.

Input

The input to the program will be one or more decks. Each deck will be represented by a string, on its own line. Each deck will consist of at least 4 cards. where a card is either an integer ***d***, **2** <= ***d*** <= **9**, representing a numbered card, or one of the letters **A**, **K**, **Q**, **J** or **T**, representing Ace, King, Queen, Jack, or Ten, respectively. The letters will be in upper case. There will be no other characters on a line. In particular, there will be no spaces. There will always be enough cards to try all valid draws. End of input is indicated by the word **JOKER**, alone on a line.

Output

Print a list of responses for the input sets, one per line. Print the word **Yes** if there is a number of cards the player can draw and win, and **No** if there is no way for the player to win. Print these words exactly as they are shown. Do not print any blank lines between outputs.

Sample input

```
Q649T
T34K35
AA2T34A5KKQAJ
JOKER
```

Sample Output

```
Yes
No
Yes
```

Toga Party

Input file: `party.in`

The guys at Delta House are at it again. They've decided to throw the ultimate toga party. Since they want this to be the best party Faber College has ever seen, they've decided to match up the men and women attending the party in such a way as to maximize the "happiness quotient". The happiness quotient contributed by a pairing is determined by the following sequence of rules:

1. If a pairing has a man and a woman who like each other, a happiness quotient of 4 is contributed to the party.
2. Otherwise, if a pairing has a man and a woman who at least tolerate each other, a happiness quotient contributed is 3.
3. Otherwise, if the woman in a pairing at least tolerates the man, the happiness quotient contributed is 2.
4. Otherwise, if the man in a pairing at least tolerates the woman, the happiness quotient contributed is 1.
5. In all other cases, no happiness quotient is contributed.

So, if Fred, Bob, and Greg were at the party with Janet, Mary, Cindy, and Dianne, and the following preference facts were known:

- Fred likes Janet
- Fred likes Mary
- Fred tolerates Cindy
- Bob tolerates Janet
- Greg tolerates Mary

- Janet likes Fred
- Mary tolerates Fred
- Janet tolerates Bob
- Mary likes Bob
- Mary tolerates Greg
- Cindy tolerates Fred
- Cindy likes Bob

the highest happiness quotient that the party could have would be 9, obtained by matching Fred and Janet (4 points) and Greg and Mary (3 points), and Bob and Cindy (2 points).

Given a number of preference facts, you should determine the maximum happiness quotient that is possible for a party. Each person can be matched with at most one other person at the party. There may be a different number of men and women.

Input:

The first line of each data set contains a non-negative integer *n*, which is the number of rules for that party. There will then be *n* lines, each with three letters on the line, *c1Fc2*. Within a rule, the men are identified by lowercase letters ('*a*'–'*z*') and the women are identified by uppercase letters ('*A*'–'*Z*'). Character *c1* and *c2* will identify a man and woman (in either order). The *F* is either an 'L' or 'T', with 'L' indicating *c1* likes *c2*, and 'T' indicating *c1* tolerates *c2*. (For example, aLB represents "a likes B" and GTg represents "G tolerates g".) Each combination *c1* and *c2* will appear at most one time in the input, so you do not need to be concerned with conflicting input (e.g., you will not have both aLB and aTB in the same set, but you may have aLB and BTa, since in the first *c1* = a and in the second *c1* = B). There will always fewer than 9 men, and fewer than 9 women.

The last data set is flagged by the number of rules being zero. This set should not be processed.

Output:

Each data set should generate a single line of output in the form

`Party` *n* `has a maximum happiness quotient of` *h*

where *n* is the number of the data set (starting at 1), and *h* is the maximum happiness quotient.

Sample Input:

```
12
fLJ
fLM
fTC
bTJ
gTM
JLf
MTf
JTb
MLb
MTg
CTf
CLb
4
ATa
ATb
aTA
bTA
6
ALa
CLa
CLb
aTA
aLC
bTC
0
```

Output Corresponding to Sample Input:

```
Party 1 has a maximum happiness quotient of 9
Party 2 has a maximum happiness quotient of 3
Party 3 has a maximum happiness quotient of 6
```

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 ($N \leq 10$), and K ($K \leq N$), 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
```

Upwards

Filename: upwards

In a previous contest hosted by Kyle, the question was posed whether or not an input word was an “upword.” In order to be such a word, all of its letters have to appear in alphabetical order, with no repeated letters. For example, “act” is an “upword”, but “cat” and “deep” are not. Arup really likes these words and wants you to investigate them further. In addition, he’s added a new definition. The original “upward” is a level-0 upward. In order to be a level-1 upward, all the letters in the word have to be in alphabetical order and the gap between consecutive letters must contain at least one letter. Thus, “ace” is a level-1 upward, but “hit” is not. In general, we can define a level-k upward to be a word in alphabetical order where the gap between consecutive letters must contain at least k letters.

Now that Arup has made his definition, he wants to know of all the level-k upwards of exactly n letters, which one is of a particular rank, r, given values for k, n and r. The rank of a word in the list is based on alphabetical order. For example, of all level-1 upwords of length 3, “ace” has rank 1 (it’s the first) and “act” has rank 16.

The Problem

Given a level, k, number of letters, n, and a positive integer rank, r, find the rth word in alphabetical order of all the level-k upwords of length n.

The Input

The first line of the input file will contain a single positive integer, *c* $(c \le 100)$, representing the number of input cases. The input cases follow, one per line. Each of these lines will have the integers, *k* $(0 \le k \le 24)$, *n* $(2 \le n \le 26)$ and *r* $(1 \le r \le 10000)$, respectively, separated by spaces. Note that r is bounded by 10000, so even if a very large number of words exist, no more than 10000 will have to be generated. Also, r is guaranteed to be less than or equal to the total number of k-level upwords with exactly n letters.

The Output

For each case, output the correct string for the query, in all lowercase letters.

Sample Input

```
2
1 3 16
0 25 24
```

Sample Output

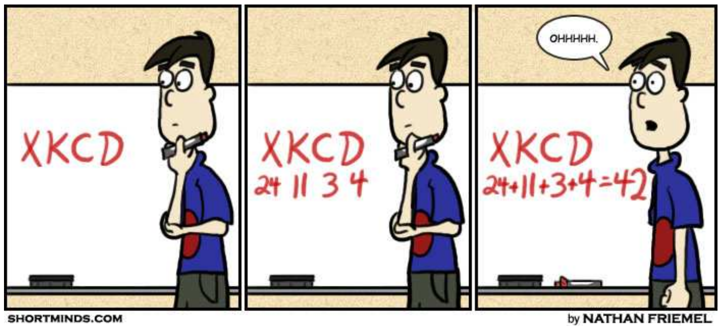
```
act
abdefghijklmnopqrstuvwxyz
```

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 $\rightarrow$ 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!
```