

Problem 0. Baby Gin

Baby Gin is played with a deck of cards, each of which is marked only with a number from 0 through 9. There are no suits. A player is dealt six cards. If the player has a Baby Gin, the casino pays off. Otherwise, the player loses.

To have a Baby Gin, the player must be able to arrange all six cards into runs or triplets. A run is a set of three cards in consecutive order, for example 345 or 789. "Around the corner" (901) is not allowed. A triplet is three cards of the same rank, for example, 000 or 444.

You have a Baby Gin if you are dealt the hand 667767 because it can be arranged into two triplets – 666 and 777. If you are dealt the hand 054060 you have a Baby Gin because it can be arranged into a triplet (000) and a run (456). But the hand 101123 cannot be arranged as a Baby Gin.

Your input file will have the following format:

- * The first line of the file contains a positive integer telling you the number of subsequent lines in the file.
- * Each subsequent line contains one hand. It consists of six numbers in the range 0 through 9, separated by spaces.

Your output file should have one line for each hand in the input file. If the hand contains a Baby Gin, the corresponding line in the output file should say, "gin." Otherwise, the line in the output file should say, "lose."

For example, consider the input file:

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

The output file should then read:

```
gin
gin
lose
```

Problem 1. Diamond Drawing

In this problem, you draw a diamond of a specified size at a specified position in a two dimensional row/column matrix. A diamond is a square, rotated 45 degrees. You draw the diamond using lower case letter x's against a background of lower case letter o's. The diamond that you draw is filled with x's (see example below).

The input file will have the following format:

* The first line of the input file contains a positive integer telling you the number of subsequent lines in the file.

* Each subsequent line specifies a diamond drawing problem, using five numbers. The numbers are separated by spaces. The first number is the number of rows in the matrix. The second number is the number of columns in the matrix. The number of rows and columns will be in the range of 1 to 20. The third number is a positive integer specifying the length of a side of the diamond. The fourth number is the row in which the top point of the diamond is located (counting the first row as row 0). The fifth number is the column in which the top point of the diamond is located (counting the first column as column 0).

You may assume that the input data is correct – that is, that the diamond is positioned it entirely within the matrix.

Your output file should draw the specified diamonds. That is, it should have a line for each row specified in a diamond drawing problem in the input file.

For example, consider the following input file.

```
2
6 8 3 0 3
8 9 4 1 3
```

The correct corresponding output file would be:

```
oooxoooo
ooxxxxoo
oxxxxxoo
ooxxxxoo
oooxoooo
oooooo
oooooooo
oooxoooo
oxxxxooo
oxxxxxoo
xxxxxxxxoo
oxxxxooo
ooxxxxooo
oooxoooo
```

Problem 2. TicTacToe

TicTacToe is a game that almost everyone has played as a child. In this game, two players (X and O) alternate placing their marks on a three by three square board. The first to succeed in placing three of their marks in a row – vertically, horizontally or diagonally – wins. It is possible for the game to end with no winner.

In this problem, you will examine a TicTacToe board to determine whether one of the players has won the game.

The **input file** will have the following format:

- * The first line contains a positive integer that tells you how many TicTacToe boards are in the input.
- * Each subsequent line contains one row of a TicTacToe game. Three rows comprise a game. Each row consists of three characters, separated by spaces. Each character is an 'X', an 'O', or a '-' (hyphen) for an empty square.

Your **output file** should consist of one line for each TicTacToe board in the input. Each line of output should contain a single character to indicate the winner – an 'X', an 'O' or a '-' (hyphen) if there is no winner.

For example, consider the following input file:

```
2
X X X
O O X
O X O
X O X
O O X
X X O
```

Your output file should look like this:

```
X
-
```

Problem 3. Hidden Word

In this problem, you must determine whether a given target word is hidden among a scrambled set of letters. For example, the word “early” is hidden among the letters

x l v y m r r a e

but “early” is not hidden among the letters

x l v y m r r e.

If the target word contains multiple instances of some letter, so must the scrambled letters. For example, “tot” is hidden among the letters

t o m a t o

but “tot” is not hidden among the letters

t o m o r r o w.

The input file will have the following format:

* The first line of the file contains an even positive integer telling you how many subsequent lines are in the file.

* The first line in each subsequent pair of lines contains the target word. The second line contains a scrambled set of letters. The first thing on each of these lines is a positive integer telling you the number of letters on the line. The letters are separated by spaces. All the letters are in lower case. There are only letters from “a” to “z” – no punctuation marks or special characters.

Your output file should contain one line for each target word in the input. If the target word is hidden among the scrambled letters, your output should read “yes.” Otherwise, it should read “no.”

For example, consider the input file:

```
4
5 e a r l y
9 x l v y m r r a e
3 t o t
8 t o m o r r o w
```

Your corresponding output file should be:

```
yes
no
```

Problem 4. Jolly Sorting

A set of numbers is sorted if all of the elements are in ascending order or all of the elements are in descending order. A set of numbers is jolly sorted if the elements are in alternating ascending and descending order.

For example, consider the set of numbers $\{1, 5, 6, 8, 9, 2, 3, 4, 7\}$. An example of a jolly sort of this set would be $\{1, 5, 2, 4, 3, 9, 6, 8, 7\}$. The first two numbers of a jolly sort (if there are two or more numbers to be sorted) should be in ascending order. Note that there may be more than one way to jolly sort a set.

In this problem, you are given a non-empty set of positive numbers and asked to jolly sort it. You may assume that no number is repeated.

The input file has the following format:

* The first line contains a single integer that tells you how many subsequent lines are in the file.

* Each subsequent line contains a sequence of numbers to be jolly sorted. The first number on the line is a positive integer no greater than 20, telling you how many numbers are in the sequence. The numbers in the sequence are separated by spaces. No two of the numbers to be sorted will be the same.

Your output file should contain one line for each input line to be jolly sorted. It should contain all the elements of the corresponding input set, jolly sorted. The numbers on the line should be separated by spaces.

For example, if the input file is:

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

then a correct output is:

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

Problem 5. Symmetric Difference

The symmetric difference of two sets, A and B, is the set C of all elements that appear in A but not in B and all elements that appear in B but not in A. For example, the symmetric difference of the sets {7, 11, 2} and {3, 7, 1} is the set {1, 2, 3, 11}.

In this problem you will be given pairs of sets of positive integers. The sets may be empty, that is, have no elements. For each pair of input sets, you will output their symmetric difference, with the elements arranged in ascending order. If the symmetric difference is empty (has no elements), you will output a hyphen ('-').

The input file has the following format:

* The first line contains a single integer in the range 2-100 that tells you how many subsequent lines are in the file. You may assume that this number is even.

* Each pair of lines (after the first line) defines the two sets whose symmetric difference you must compute.

* Each line specifies a set of positive integers. The first integer on the line, N, tells you how many integers are in the set. N is in the range 0 to 20. The number N is followed by N distinct integers separated by spaces. Each of these N numbers is in the range 1 to 100.

Consider, for example, an input file that contains these lines:

```
6
3 7 11 2
3 3 7 1
3 1 2 3
3 3 2 1
2 17 99
1 99
```

Your output file should contain one line for each of the pairs of consecutive input lines. In our example, the correct output is:

```
1 2 3 11
-
17
```

Problem 6. Tribble Population Trouble

The future world tribble population is entirely determined by two factors: the number of tribbles that are born in Year of the Tribble X (YT-X) and the number of tribbles that are one year old in YT-X. This is because tribbles have a somewhat unusual pattern of reproduction. All tribbles are born on January 1. When a tribble reaches its second birthday it has two offspring, and then it dies.

For example, suppose that in YT-0 4 tribbles were born. This would mean that 2 tribbles had reached two years old, each had two offspring, and then died. Also, in YT-0 there were 6 one-year-old tribbles. Then the future world tribble population would look like this:

Year	# born	1-year	total
0	4	6	10
1	12	4	16
2	8	12	20
3	24	8	32
4	16	24	40

In this problem, you are given a number of scenarios for the number of tribbles born in YT-0, the number of tribbles that are one year old in YT-0, and a year X. Based on the initial tribble population, you must calculate world tribble population in year X.

The input file has the following structure:

* The first line has a positive integer telling you the number of subsequent lines in the file.

* Each subsequent line has three non-negative numbers, separated by spaces, describing a tribble population scenario. The first number is the number of tribbles born in YT-0. The second number is the number of tribbles reaching one year old in YT-0. The third number is a year X on the tribble calendar (YT-X).

Your output file should contain one line for each tribble population scenario in the input file. The line should contain the projected world tribble population in year X.

For example, consider the input file:

```
1
4 6 3
```

The output file should contain:

32

Problem 7. Turkey or No Turkey

You are trying to return home from college for the Thanksgiving holiday. The only form of transportation available to you is the train, and it runs only north and south – 1,000 miles in each direction. Your home is N miles from your college. You have 10 train tickets, each containing a number in the range from 0 to 100. You may use each ticket to travel that number of miles either north or south. You can choose whether to travel north or south with each ticket, but you must travel the whole distance. If you can figure out a way to use all your tickets and end up at your home, you get turkey this Thanksgiving. Otherwise, you eat pizza.

For example, if you are 18 miles north of your home and your ten tickets are 7, 3, 9, 18, 2, 0, 56, 12, 34 and 17, then you can get home by traveling 7, 3, 9, 34 and 17 miles north and then traveling 18, 2, 56 and 12 miles south. The 0 miles can be in either direction.

In this problem, you must determine whether it is possible for you to get home for Thanksgiving with a given set of tickets.

The input file has the following structure:

* The first line has a positive integer telling you how many subsequent lines there are in the file.

* Each subsequent line has 11 integers, separated by spaces. The first integer tell you how far you are from home. The next 10 integers are the distance you can travel with each ticket. (Hint: the distance from your home matters, but it doesn't matter whether you are north or south of your home – you could always reverse the direction of travel with each ticket.)

Your output file should have one line for each travel problem. If there is a way for you to get home using all 10 tickets, the output should read, "turkey." Otherwise, the output line should read, "pizza."

For example, consider the input file:

```
2
18 7 3 9 18 2 0 56 12 34 17
7 14 92 8 24 36 18 44 14 58 20
```

The output file should read:

```
turkey
pizza
```

(It is easy to see that there is no way to get home in the second case, since all the tickets have even numbers and you are an odd number of miles from home.)