Mercer Island Programming Contest 2017

Problem Set

Rules:

1. Each question is worth 50 points. With each incorrect submission, the value of that question decreases by 5 points. You do not need to solve the problems in order.

2. The internet may not be accessed during the contest coding phase. Only one computer per team can be out during this time, meaning phones also need to be put away.

3. The teams with the highest score at the end of the contest will receive awards. In the case of a tie, the team that made their final successful submission first will be the winner.

Problems:

1. Small Car
2. Sorting Distances
3. Word Spelling
4. Magic Square
5. Valid DNA
6. Counting Time
7. Playground
8. Multiplying Ducks
9. Alien Message
10. New Palindrome
11. Best Bootcamps
12. Closest ZIP Codes
13. Sudoku
14. Full Bookshelf
15. Base Palindrome

1. Small Car
Input File: None

In *Small Car*, you must print out the car shown below. There is no input in this problem, but your car must match **exactly** with the one shown, down to the last dots and spacing.

Input:

None.

Output:

You will output the exact car is shown below.

Example Output:

```
    .-----.
 __//.{)|_\\._
/___         ___:
--(_)------(_)--
```

2. Sorting Distances
Input File: sorting.txt

In *Sorting Distances*, you are a taxi driver who is trying to find the different paths to your destination in descending order. However, all you have been given are a bunch of scrambled distances. Your job is to return the distances you have been given sorted from **greatest to least**.

Input:

The first line contains an integer N. The following N lines will contain one number each, representing a path's distance to the desired destination. All distances will be greater than 0, and no two distances will be the same.

Output:

You will need to output all of the distances sorted from greatest to least, with each distance on a different line.

Example Input:

```
4
489
507
23
17001
```

Example Output:

```
17001
507
489
23
```

3. Word Spelling
Input File: spelling.txt

In *Word Spelling*, you are given a word, and multiple different letters following the word. You must determine if those letters can be arranged to spell the word shown. Not all the letters have to be used, and no letter can be used twice. Capitals are different from lowercase.

Input:

The first line contains an integer N. The following N lines will contain one word, followed by an integer K representing the number of characters following.

Output:

You will need to output “yes” or “no” depending on whether or not the list of letters can be used to spell the word.

Example Input:

```
4
apple 6 l y a p e l
John 3 J h n
bottle 10 e e t l b o o e a c
Poster 9 r t a n o P p s e
```

Example Output:

```
no
no
no
yes
```

4. Magic Square
Input File: magic.txt

A magic square is a 3x3 square where the numbers 1-9 fill up each tile, and each row, column, and diagonal adds up to 15. The table below shows a magic square.

2	9	4
7	5	3
6	1	8

As you can see, if you take any three numbers that make up a row, column, or diagonal, the sum is always 15. Unfortunately, the pattern shown above is the only way to make it perfectly. Your friend is trying to make these magic squares, but most of the time, a few of the rows don't perfectly add up to 15. Your task is to count up how many rows, columns, and diagonals that need to be fixed before it can be a magic square.

Input:

The first line contains an integer N, representing the number of sets of data. The next N sets of data each contain three lines with three numbers on each line.

Output:

Output the number of rows, columns, and diagonals that don't add up to 15 for each set of data.

Example Input:

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

Example Output:

```
2
3
```

5. Valid DNA
Input File: dna.txt

DNA is made of of nitrogen bases that form a long sequence. There are four nitrogen bases: A, T, C, and G. Since DNA is made of two strands, each of these sequences is paired with another strand that contains the exact opposite pattern. An A on one strand always pairs up with a T on the other and a C on one strand always pairs up with a G on the other. Therefore, if one sequence is ATCGC, the other sequence will be TAGCG. Your task is to determine whether the given DNA sequences follow this rule or not.

Input:

The first line contains an integer N, representing the number of sets of DNA. The following N sets of data each contain two lines, with one ten-letter sequence of A's, T's, C's, and G's on each line.

Output:

For each set of data, determine whether the two sequences of nitrogen bases are valid to be paired together (A must go with T, C must go with G). Either output "Valid" or "Invalid."

Example Input:

```
3
ACTACGCATC
TGATGCGTAG
GCATGACTAG
CGTAGACATC
GTCATCGATC
CAGTAGCTAG
```

Example Output:

```
Valid
Invalid
Valid
```

6. Counting Time
Input File: time.txt

All of the clocks you own have broken, so the only way for you to know how much time has passed is by counting each second individually. Given how many seconds have passed by, calculate that time in terms of days, hours, minutes, and seconds to make it easier to get a sense of how long it has been.

Input:

The first line contains an integer N. The following N lines each contain an integer representing the amount of time that has passed in seconds.

Output:

Convert the time passed in seconds to be in terms of days, hours, minutes, and seconds, in the following format: # days # hrs # min # sec.

Example Input:

```
3
122
6734
5917380
```

Example Output:

```
0 days 0 hrs 2 min 2 sec
0 days 1 hrs 52 min 14 sec
68 days 11 hrs 43 min 0 sec
```

7. Playground
Input File: playground.txt

Your local elementary school wants to build a new playground. However, they have a limited amount of money, and can only afford a certain amount of fencing needed to surround the playground. Given the price for each foot of fencing and the school's budget, calculate the maximum area of the playground. Note that the playground must be in the shape of a rectangle (four sides, with 90° angles), meaning no circular or triangular playgrounds.

Input:

The first line contains an integer N. The following N lines each contain two positive integers, the first representing the price per feet for fencing and the second representing the school budget.

Output:

Output the maximum area of the playground that can be surrounded by fencing. Your answer should be in square feet and rounded to the nearest integer.

Example Input:

```
3
5 1000
8 100
6 600
```

Example Output:

```
2500
10
625
```

8. Multiplying Ducks
Input File: ducks.txt

Ducks are the main species that live on Duck Island. The island contains a variety of food for the ducks and no predators, so the ducks grow in population very fast. Every 10 days, the duck population suddenly doubles. Every 24 days, one fourth of the duck population passes away from old age (if these two events occur on the same day, assume the population doubles before any ducks pass away). Given the initial population on January 1st, find the population of ducks on a later date in the same year. Assume that it is not a leap year, meaning February has 28 days, January, March, May, July, August, October, and December have 31 days, and the rest have 30 days.

Input:

The first line contains an integer N. The following N lines each contains an integer P, the initial population, a space, and then a date in the format of mm/dd.

Output:

Output the duck population on the given date.

Example Input:

```
3
10 02/01
55 06/17
1 12/31
```

Example Output:

```
60
641520
918330048
```

9. Alien Message
Input File: alien.txt

You have just received a secret message from an alien planet. However, the message you received is encrypted and currently makes no sense. Fortunately, the aliens have also sent you the instructions to decode the message. Your job is to use the key and instructions provided to decrypt the text, and print out the correct message.

Input:

The first line contains an integer N, which represents the number of messages you must decrypt. The next line will contain two integers separated by a space, indicating the number of rows and columns, respectively, in the matrix message below. This is followed by a matrix of that size. This pattern will repeat N times (representing the number of messages you receive).

Output:

In order to decode the message, you must first divide each element in every row by 7, then subtract **every other element** in every **column** by 19 (first element you subtract, second you do nothing, third element you subtract, etc.). Then, you convert all the numbers to chars, getting the equivalent ASCII character (i.e. 65 corresponds to 'A'), and print out every **column** all on the same line. You do this for each message, and have a new line for each new message.

Example Input:

```
1
7 7
742 357 357 910 854 854 945
707 812 588 700 798 224 735
889 910 840 840 812 602 945
693 224 679 224 763 777 735
910 945 896 693 896 896 910
763 728 805 798 735 784 770
840 840 602 910 903 840 364
```

Example Output:

```
Welcome to the TeamsCode Programming Competition!
```

10. New Palindrome
Input File: palindrome.txt

You've been given a sentence by your friend, and he has a special task for you. You must determine the minimum number of characters you must remove in order to make that sentence a palindrome. The palindrome is case-sensitive ('S' is not equal to 's'), and spaces do **not** count as characters.

Input:

The first line contains an integer N, which represents the number of sentences you are given (each line is a separate sentence and a new problem). The next N lines each contain one sentence.

Output:

You must output the minimum number of characters you must remove in order to make the sentence a palindrome. You must do this N times (once for every new sentence), and each new answer should be on the next line.

Example Input:

```
3
Redivide
please do not step on the pets there.
The owl Ate my metal worm
```

Example Output:

```
1
15
7
```

11. Best Bootcamps
camps.txt

In order to keep learning how to code, you are trying to find different coding bootcamps to attend. To keep track of which camp best, you've created your own scale (the LP scale) to measure how much you're learning. Initially, some camps are better than others, and therefore offer more LP points. However, you plan on attending more than just one camp, and you've also found that going to different camps in a certain order can help boost (or lower) the amount of learning you take in. For example, taking the bootcamp A class and then the bootcamp B class will give you the LP points of A, while giving you a bonus of a certain number of LP points for following up with B. Similarly, if you follow up camp B with another camp C, you might gain or lose another amount of points, depending on whether camp C builds upon or subtracts from B. Your task is to find the best order of classes to take to maximize your LP points.

Input:

The first line contains two integers. The first integer N represents the number of bootcamps there are, and the second integer represents the number of camps you have the budget to attend (assume you will attend as many camps as your budget allows. The next line contains the name of N bootcamps followed by the LP points they offer you for choosing them as your initial bootcamp. The next line contains another integer L. Each of the following L lines contains the name of two bootcamps, followed by an integer representing how many extra points you gain (or lose) from following up with the second bootcamp after taking the first bootcamp.

Output:

Output the bootcamps you attend, in order, followed by the maximum amount of LP points you can gain.

Example Input:

```
5 4
CodeNow 50 SmartCoder 75 BasicOOP 55 BigHack 65 CodingTeam 80
7
BasicOOP SmartCoder 35
CodeNow CodingTeam 15
BigHack BasicOOP -10
SmartCoder BigHack 20
CodingTeam BasicOOP -5
BigHack CodeNow 10
SmartCoder CodingTeam 5
```

Example Output:

```
CodingTeam BasicOOP SmartCoder BigHack 130
```

12. Closest ZIP Codes
Input File: zipcode.txt

You have just graduated from college and don't know where you want to live. One of your friends from high school suggests that you buy a house nearby because the prices are very cheap. The problem is, however, that you have a strict policy that you can only move to places where the ZIP code contains the same digits as your current address. Given your current ZIP code, rearrange the numbers so that the new number is as close to your friend's ZIP code as possible (determined by the difference between the two numbers).

Input:

The first line contains an integer N. The following lines each contain two five-digit numbers, the first of which is your current ZIP code and the second is your friend's.

Output:

Output your new rearranged ZIP code that's closest to your friend's ZIP code, followed by the difference of the two numbers.

Example Input:

```
3
45467 75885
98040 56470
29399 30106
```

Example Output:

```
75644 241
49800 6670
29993 113
```

13. Sudoku
Input File: sudoku.txt

You are given a 9x9 sudoku grid. The grid is almost completely solved. Your task is to find the remaining squares that are unsolved to finish the puzzle. To make it easier, there will be at most two empty squares in each of the smaller 3x3 grids.

Input:

The first line contains an integer, which gives you how many numbers are missing. The following nine rows each contain nine numbers separated by a space. The numbers will be between 1 and 9. Blank spaces will be marked with a '?'.

Output:

Output the missing numbers in order of appearance (left to right, then top to bottom).

Example Input:

```
11
2 9 ? 7 4 3 8 6 1
4 ? 1 8 6 5 9 ? 7
8 7 6 1 9 2 5 4 3
3 8 7 4 5 9 2 1 6
6 1 2 3 ? 7 4 ? 5
? 4 9 2 ? 6 7 3 8
? ? 3 5 2 4 1 8 9
9 2 8 6 7 1 ? 5 4
1 5 4 9 3 ? 6 7 2
```

Example Output:

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

14. Full Bookshelf
Input File: bookshelf.txt

You have a bookshelf in your room that you use to hold all your books. However, as you get older and start to read books with more and more pages, your bookshelf doesn't seem to have enough room anymore. Given a bookshelf with a certain number of rows, each of which a certain width, find out if it is possible to keep all of your books on the bookshelf, and if not, the least amount of books that you must get rid of in order to fit the rest.

Input:

The first row contains an integer N, followed by N sets of input. In each set of input, the first line contains two integers, the first of which is the number of rows your bookshelf has and the second is the width of each row. The next line contains an integer K, and the last line contains K integers, each one representing the width of one of your books.

Output:

For each input set, if you are able to fit all of the books on the bookshelf, output '0'. Otherwise, output the least amount of books that you must get rid of.

Example Input:

```
3
3 20
8
7 8 6 7 9 8 7 6
2 30
11
5 6 4 1 7 12 6 2 9 5 3
4 10
9
15 11 2 6 9 1 13 5 3
```

Example Output:

```
1
0
3
```

15. Base Palindrome
Input File: palindrome2.txt

You have been given a list of numbers and your boss wants you to determine which of these numbers are palindromes. However, there's a catch - your boss wants you to check if these numbers are palindromes in multiple different bases. Given a list of numbers in base-10, determine a list of in which bases each number is a palindrome (from bases 2-20).

Input:

The first row contains an integer N, followed by N sets of input. In each input, the line contains one positive integer in base-10 format.

Output:

For each line, you must print the different bases in which the number is a palindrome (from bases 2-20). The format will be the different bases separated by commas, with a new line for every new input you are testing. If there is no palindrome for any base, simply print "None". The integer input value will always be less than one billion.

Example Input:

```
5
10
86
717
87
829100
```

Example Output:

```
3, 4, 9, 11, 12, 13, 14, 15, 16, 17, 18, 19, 20
6
2, 10, 14
None
16
```