MIHS 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. Inspirational Message
2. Distance to Shore
3. Triangle Creator
4. Compound Interest
5. Texas Hold'em
6. Find Prime
7. Decoder
8. Best Football Team
9. Changing a 2D Array
10. Pattern Sequence
11. Underground Maze
12. Road Race
13. Hiking
14. Longest Common Sequence
15. Binary Code

Inspirational Message
Input File: inspiring.txt

In *Inspirational Message*, you are trying to cheer up your friend. You must write an inspirational note to him to brighten his day. You will be given a certain number of words that you must print to the console in the specific format given below.

Input:

The first line will have a number specifying the number of lines below. The next lines will each contain one word (string). You do not need to check the validity of the input.

Output:

You will output the words in the specific format shown in the example output below. You must print out **everything** shown below the example output, including all of the dashes, slashes, etc. in the answer.

Example Input:

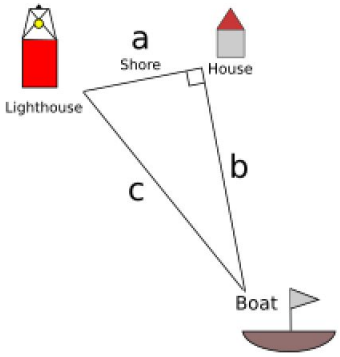
```
3
Amazing
Awesome
Fantastic
```

Example Output:

```
/----------\
\*!Amazing!*/
\*!Awesome!*/
\*!Fantastic!*/
\----------/
```

Distance to Shore
Input File: distance.txt

In *Distance to Shore*, you are in a boat a certain distance from shore. On the shore there is a house, and a little farther to the left is a lighthouse. The boat, the house, and the lighthouse form a right triangle. The following diagram illustrates the positions of the boat, the shore, and the lighthouse. You know the distances between two of the different objects, and must find the third using the Pythagorean Theorem.



Input:

The first line contains an integer N. The following N lines will contain two numbers, with an a, b, or c preceding each number. The letters represent one of the sides that is shown above - c is a hypotenuse, and a and b are legs. You can assume that c will be greater than a and b.

Output:

You will need to output the length of the side that is not given using the Pythagorean Theorem. Round the answer **down** to the nearest integer.

Example Input:

```
2
A 15.2 C 18.3
B 5 A 12
```

Example Output:

```
10
13
```

Triangle Creator
Input File: triangle.txt

You have a math exam next period, and your friend tells you one of the questions, but forgets the exact values. He tells you that one of the problems asks to draw x number of triangle of varying N heights. You are unaware of the number of triangles you will need to draw. Create a program that will print a triangle, given N, with '/' and '\' for the sides and '_' for the bottom, with N being the triangle's height in lines.

Input:

The first line represents x, or the number of triangles to draw. Each following line will contain N, or the height of the triangle.

Output:

Output x number of triangles, using '/' and '\' for the sides and '_' for the bottom, with the given height(s) of N, including a line to separate the printed triangles.

Example Input:

```
2
3
5
```

Example Output:

```
  /\
 /  \
/____\

    /\
   /  \
  /    \
 /      \
/________\
```

Compound Interest
Input File: compound.txt

Suppose Michael has P dollars to invest in an account that pays r% interest compounded quarterly. How much money does Michael have after t years?

The formula for compounded interest is:

$A = P\,(1 + r/n)^{(nt)}$

Input:

The first line is the number of lines to follow. Each line will contain the initial investment into the bank account (P), the number of times the interest is compounded annually (n), the interest rate (r) (in decimals), followed by the time in years since the initial investment was deposited into the account (t) (in order).

Output:

Output the amount of money in the account after t years rounded **down** to the nearest integer, including a dollar sign ($) before the value.

Example Input:

```
2
1000 4 0.035 5
2500 2 0.05 10
```

Example Output:

```
$1190
$4096
```

Texas Hold'em
Input File: poker.txt

You are playing a game of Texas Hold'em poker against your friend. There are five cards on the flop and you and your friend both have two cards in hand. You know that you have a pair or a triple, but need to find what is the highest pair or triple you have in a hand of seven cards. In Texas Hold'em, 2 is the lowest card and ace is the highest. Find the highest pair you have in a hand of five cards, or the highest triple if it exists.

Input:

The first line represents the number of lines to follow. Each line contains the sequence of seven cards you have in your hand and on the flop, separated by a space.

A - Ace
K - King
Q - Queen
J - Jack

Output:

Output the highest pair from the seven cards. If a pair and a triple exists in your hand of seven cards, the triple supersedes the pair and is therefore the only value outputted.

Example Input:

```
2
10 10 9 9 A J K
Q 2 J Q J Q A
```

Example Output:

```
10 10
Q Q Q
```

Find Prime
Input File: prime.txt

You are just learning what prime numbers are. In order to help you learn, you want to create a program to print out the first N prime numbers. However, this comes with a twist. To showcase your coding skills, you want to square every number between the integers K1 and K2.

Input:

An integer L, followed by L lines. Each following line will contain three positive integers, N, K1, and K2.

Output:

Print the first N prime numbers. If the prime number is between the values of K1 and K2 (inclusive), square the prime number before printing it.

Example Input:

```
3
8 5 15
3 1 5
26 80 82
```

Example Output:

```
2 3 25 49 121 169 17 19
4 9 25
2 3 5 7 11 13 17 19 23 29 31 37 41 43 47 53 59 61 67 71 73 79 83 89 97 101
```

Decoder
Input File: decoder.txt

You are sent a message by your company, however, your company's message has to be decoded. The message is all numbers, and each number corresponds to a letter like follows: a = 1, b = 2, c = 3, and so on. Also, a "*SPACE"* = 27, and a period = 28. You must run the message through your program, decode the message, and print the decoded message to the console.

Input:

You are given a random message, where each number is separated by spaces, and the number 27 represents a space in your decoder. The different lines have no meaning. You can assume that the input is valid.

Output:

You should output the decoded message. You do not need to worry about correct capitalization.

Example Input:

```
8 5 12 12 15 27 16 5 18 19 15 14 28 27 27 20 8 9 19 27 9 19 27 20 8 5 27 13 5 19 19 1
7 5 27 9 27 23 1 14 20 27 20 15 27 19 8 1 18 5 27 23 9 20 8 27 25 15 21 28
```

Example Output:

```
hello person.  this is the message i want to share with you.
```

Best Football Team
Input File: football.txt

In your state, there are several school football teams. You want to find which team has the best record. Given the results of the games, list the football teams in the order they appear, followed by their win loss record. Assume there are no ties.

Input:

An integer N, followed by N lines. Each following line represents a game and contains two team names. The first name is the winner of the game and the second name is the loser.

Output:

In the order that the names appear, print their team name, followed by the number of games they won, a forward slash, and the number of games they lost.

Example Input:

```
7
Thunderstrikes Gorillas
Termites Earthquakes
Waves Emperors
Earthquakes Gorillas
Thunderstrikes Waves
Termites Emperors
Thunderstrikes Termites
```

Example Output:

```
Thunderstrikes 3/0
Gorillas 0/2
Termites 2/1
Earthquakes 1/1
Waves 1/1
Emperors 0/2
```

Changing a 2D Array
Input File: array2d.txt

You are given a file with a printed 2D array. You must change the array in the following manner:

- If the number is a multiple of 3, you must square that number
- If the number is a multiple of 5, you must take the nth root of the number, where n equals the row number (the first line describing the number of columns/rows does not count as a row number, and the rows start at 1, not 0)
- If the number is a multiple of 3 AND 5, you must divide the number by 5

All your numbers must be rounded to the nearest integer. 0 is **not** considered a multiple of 3 or 5 or both.

Input:

The first line has a number that represents the number of columns, followed by a space, followed by a second number that represents the number of rows. Then the printed array is shown, where each row has integers separated by spaces, and the columns on separate lines.

Output:

You will output the 2D array modified using the changes described above.

Example Input:

```
3 4
5 71 2
8 9 15
2 20 1645
57 40 105
```

Example Output:

```
5 71 2
8 81 3
2 3 12
3249 3 21
```

Pattern Sequence
Input File: pattern.txt

You have created a survey that asks 20 participants to rate their level of happiness in school from 0 to 10. Your goals is to find the longest recurring pattern of values from the sequence of 20 values given by the participants. Your requirements for a recurring pattern is that the pattern of numbers occurs at least twice in sequence of 20 values. Given the sequence of 20 numbers, find how many numbers is in the longest recurring pattern.

Input:

The first line represents the number of lines to follow. Each line contains the sequence of 20 values, ranging from 0 to 10, including a space separating each value.

Output:

Output the longest recurring pattern followed by an integer that represents the length of the longest recurring pattern on the next line in the input of 20 values, with a space separating each value.

Example Input:

```
2
4 5 2 4 5 4 3 9 10 3 2 6 2 7 6 9 4 5 2 4
10 10 10 3 2 1 9 1 2 0 9 2 10 10 10 3 2 1 9 6
```

Example Output:

```
4 5 2 4
4
10 10 10 3 2 1 9
7
```

Underground Maze
Input File: maze.txt

You are trapped in an underground labyrinth and need to get above ground fast. You must find your way through an 8x8 maze.

Input:

A number N, representing the number of mazes to follow, followed by mazes with an ‘S’ representing the starting position, an ‘X’ representing the ending position, a ‘#’ representing a wall, and a ‘.’ at every place you are allowed to move to.

Output:

The maze with a space replacing each ‘.’ you must pass in order to get from start to finish. The ‘S’ should be replaced with a space and the ‘X’ should be replaced with the ‘S’.

Example Input:	Example Output:
2 ######## #S.....# ######.# ##X###.# ##.#...# ##.###.# #......# ######## ######## #.....S# ###.#### #...#..# #.##...# #.#..#.# #....#X# ########	######## # # ###### # ##S### # ## #.. # ## ### # #. # ######## ######## #.. # ### #### # #..# # ## # # #. # # # #S# ########

Road Race
Input File: roadrace.txt

You are competing with your friend to see who can travel across the United States and make it through all the major cities the fastest. Your starting position is in Seattle and your ultimate target is Miami. However, there are no direct paths to Miami; you must travel through other cities on your way there. You must print the cities that you travel through which allows you to take the shortest route from Seattle to Miami.

Input:

An integer N, followed by N lines. Each of the following lines contains a city name, followed by a space and another city name, with an integer afterwards representing the distance between the two cities.

Output:

Print the cities (in order) that you must travel through that allows you to take the shortest route to Miami. Include Seattle at the beginning and Miami at the end.

Example Input:

```
9
Boise Detroit 13
Seattle Sacramento 10
Phoenix Dallas 18
Detroit Sacramento 15
Seattle Portland 5
Miami Detroit 14
Dallas Miami 8
Phoenix Seattle 12
Dallas Boise 7
```

Example Output:

```
Seattle Phoenix Dallas Miami 38
```

Hiking
Input File: hiking.txt

You are going on a hike on Mount Rainier. Unfortunately, you have an old backpack that only can carry 20 pounds worth of equipment. Your job is to decide which items to bring along to maximize the value of things you are bringing in your backpack.

Input:

An integer N, representing the number of following cases. For each case, there will be five lines, each representing an object. Each line will contain two numbers: the first being the weight of the object and the second being the value of the object.

Output:

The highest total value of all the objects you can put in your backpack without going over the 20 pound weight limit.

Example Input:

```
2
5 6
10 11
3 2
7 4
4 5

12 2
5 12
7 7
4 2
8 9
```

Example Output:

```
22
28
```

Longest Common Subsequence
Input File: longestcs.txt

You are given two strings, both of which are a random jumble of letters. The two strings share a few common letters here and there. Your tasks is to find the longest common subsequence in both strings. This subsequence does not need to be continuous, but must be in order relative to the other common characters.

Input:

An integer N, followed by N lines. Each of the following N lines contains two strings, separated by a space.

Output:

The longest common substring of the two strings. If there are multiple substrings of the same longest length, print the first occurring one.

Example Input:

```
3
sadkspoqe kdkjjsceoe
abcdef fedcba
apldfpaple gijapebpjkele
```

Example Output:

```
dksoe
a (could also be b, c, d, e, or f)
apple
```

Binary Code
Input File: binary.txt

You are passed a string of 0s and 1s. However, this binary string passed to you is incomplete. Given the correct string, your job is to determine whether it is possible to modify the incomplete binary string to make it equal to the correct version using only two modification techniques: adding the characters “10” to the end of the string or moving the last digit of the string to the beginning. For example, you could change “011” to “01110” by adding “10” or change “011” to “101” by moving the last digit to the beginning.

Input:

An integer N, followed by N lines. Each following lines contains two strings: the first is the incomplete string and the second is the complete string.

Output:

Print “Possible” if you can use a combination of adding “10” to the end of the incomplete string and moving the last digit of the incomplete string to the beginning in order to make it match the complete string. Print “Impossible” otherwise.

Example Input:

```
3
1011 1001110
111 011011101
1100 01100110
```

Example Output:

```
Impossible
Possible
Possible
```