CodeLM 2016

Advanced Division

School Name

Your password is: _____

Wi-Fi SSID: Programming

Wi-Fi Key: comp1234

Rules: You will have two and a half hours to complete your division's seven questions. All questions must be submitted through the website codeLM.com and have a runtime of less than 10 seconds. You will submit your code by copying and pasting the code from your IDE to each problem's textbox on the website and then hitting submit.

All method headers must exactly match the method header that is provided in the starter code and submission textbox. You may also submit other methods that your code may use. You should not submit a main method.

Problems 1 and 2 are each worth 2 points; Problems 3 and 4 are each worth 3 points; Problems 5 and 6 are each worth 4 points. In order to receive credit for these questions your code must produce the correct results for all test data. The test data will not be the same as the sample data that is provided with each question in this packet, so be sure to thoroughly test your code before you submit it. Incorrect submissions that compile will result in the loss of one point. If your submission does not compile you will not lose a point. If you believe that your code has incorrectly been marked as wrong you should inform a judge who will then have the question reviewed. You may make as many submissions for each question as you would like, although you will only receive credit for one correct answer per problem.

A real-life developer will grade problem 7. The rubric by which it will be graded is located at the back of this packet. Note that a non-working or partially working project can still receive points for design and organization. The final question will also be submitted through the website codeLM.com. Unlike the first six questions, you will upload the source code through the website as opposed to copying and pasting your code into a textbox. There are no restrictions to method and class names for the final problem.

Teams will be ranked by total points earned. If several teams earn the same amount of points, the teams will be ranked according to the following criteria:

1. Most correct answers.
   then
2. Least incorrect answers.
   then
3. Time last auto-graded solution was entered (earliest to latest).

Problem 1: Loading Symbol (2 points)

You are designing an hourglass loading symbol for New Wave's new patent-pending word processor. Your loading icon needs to be resizable in order to work well on screens of different sizes.

Input
An integer, **n**, representing the number of rows the hourglass should have.

Output
An hourglass pattern made with **n** rows where each row is made out of the first **m** characters of the string "NewWave". If a row has more than 8 characters you should repeat the characters in the string "NewWave" as needed.

If **n** is odd, the first row should be **n** characters long, with each succeeding row having 2 fewer characters with spaces to center it until there is only 1 character, at which point the pattern is reversed until there are **n** characters again. There will only be one row that has a single character.

If **n** is even, the first row should be **n-1** characters long, with each succeeding row having 2 fewer characters with spaces to center it until there is only 1 character, at which point the pattern is reversed until there are **n-1** characters again. There will be exactly two rows that have a single character.

Note
There should not be any spaces between characters. Your output for this problem will be printed to the console. Your method will not return anything; it is a void method.

Sample Data

Input	Output
1	N
2	N N
3	New N New
4	New N N New
5	NewWa New N New NewWa
10	NewWaveNe NewWave NewWa New N N New NewWa NewWave NewWaveNe

Problem 2: Weekly Report (2 points)

Due to recent inconsistencies observed by the SEC, New Wave Computers™ must release weekly earnings reports. Unfortunately these reports often contain several mistakes. The SEC also requests that all monetary values be converted to euros so they can better analyze New Wave's oversea transactions.

Input

A string, **`report`**, containing the full report.

Output

The same report, but with the following modifications made:

- The first letter of every sentence must be a capital letter, including the first sentence in the report. A sentence can end with either a period, a question mark, or an exclamation point. There will always be exactly one space between a punctuation mark and the first letter of the next sentence.
- In all instances of "mr", "ms", or "mrs", the m should be capitalized. All instances of these titles will be followed with a period and a space.
- All dollar values should be converted to euro values. A dollar value is a dollar sign followed by one or more integer digits with no spaces. The dollar sign should be replaced with a euro sign (we'll just use a capital E), and the value should be converted based on the equation *euro* = .75 * *dollar*. All numeric values greater than 999 will be written without commas and all dollar values will be whole numbers.

Sample Data

Input	**Output**
here at New Wave, we are proud to launch our brand new wPhone 2, starting at just $500. our CEO mr. swope is very proud of our teams hard work.	Here at New Wave, we are proud to launch our brand new wPhone 2, starting at just E375. Our CEO Mr. Swope is very proud of our teams hard work.
in our last quarter, we made a profit of $375 per wPhone and $10 per software package. our stock price also grew 3.1%.	In our last quarter, we made a profit of E281.25 per wPhone and E7.5 per software package. Our stock price also grew 3.1%.
employee complaint #687517: mr. John has been acting up again. i cannot believe he makes $10000 a year. fire him. now!	Employee complaint #687517: Mr. John has been acting up again. I cannot believe he makes E7500 a year. Fire him. Now!

Problem 3: Yacht Sea (3 points)

At New Wave Computers'™ annual picnic, employees compete in a dice game called Yacht Sea. The game is played by rolling five dice and then filling in a table based on these values that have been rolled. You will write a program that will determine the highest score that can be achieved with a single roll.

Input

`roll` - An array of 5 integers.

Output

An **`integer`** value that is the highest score that can be achieved from the roll based on the following table:

	How to score
Ones	Count and score only the Aces
Twos	Count and score only the Twos
Threes	Count and score only the Threes
Fours	Count and score only the Fours
Fives	Count and score only the Fives
Sixes	Count and score only the Sixes
Full House	Score 25
Small Straight (sequence of 4)	Score 30
Yacht Sea (all dice have the same value)	Score 50

Sample Data

Input	**Output**	**Explanation**
roll = {6, 2, 6, 6, 2}	25	Full House
roll = {1, 4, 6, 4, 1}	8	Fours
roll = {1, 2, 4, 3, 2}	30	Small Straight
roll = {1, 1, 1, 1, 1}	50	Yacht Sea
roll = {2, 2, 5, 3, 1}	5	Fives

Problem 4: Newdoku (3 points)

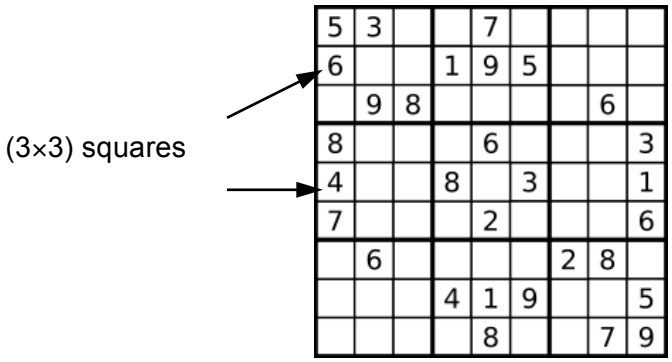
Several New Wave employees plan to enter a programming competition. Each competitor brings a Sudoku solver and compete to see which can solve the most challenging Sudoku. They need a little help making sure the solutions from their solvers are correct.

Input

A two-dimensional 9x9 array of integers, **`board`**, representing a completed board.

Output

An **`integer`** value, representing the number of total errors on the board. An error occurs in any row (left to right), column (up and down), or box (3x3 square) if the row, column, or box does not contain the digits 1-9, each exactly once. The squares do not overlap.



Sample Data

Input	Output	Explanation

board = {
{5, 3, 4, 6, 7, 8, 9, 1, 2},
{6, 7, 2, 1, 9, 5, 3, 4, 8},
{1, 9, 8, 3, 4, 2, 5, 6, 7},
 	
{8, 5, 9, 7, 6, 1, 4, 2, 3},
{4, 2, 6, 8, 5, 3, 7, 9, 1},
{7, 1, 3, 9, 2, 4, 8, 5, 6},
 	
{9, 6, 1, 5, 3, 7, 2, 8, 4},
{2, 8, 7, 4, 1, 9, 6, 3, 5},
{3, 4, 5, 2, 8, 6, 1, 7, 9}
};
	0	

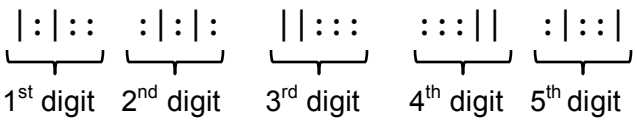
board = {
{1, 7, 5, 8, 3, 9, 4, 2, 6},
{6, 3, 6, 2, 7, 4, 9, 1, 5},
{4, 2, 9, 6, 5, 1, 3, 7, 8},
 	
{8, 1, 8, 3, 9, 5, 7, 4, 2},
{5, 4, 7, 1, 6, 2, 8, 3, 9},
{2, 9, 3, 4, 8, 7, 6, 5, 1},
 	
{7, 5, 4, 9, 2, 6, 1, 8, 3},
{9, 8, 1, 5, 4, 3, 2, 6, 7},
{3, 6, 2, 7, 1, 8, 5, 9, 4}
};
	4	Rows 2 and 4 are incorrect as well as the upper left and middle left box

board = {
{5, 3, 4, 6, 7, 8, 9, 1, 2},
{2, 7, 6, 1, 9, 5, 3, 4, 8},
{1, 9, 8, 3, 4, 2, 5, 6, 7},
 	
{8, 5, 9, 7, 6, 1, 4, 2, 3},
{4, 2, 6, 8, 5, 3, 7, 9, 1},
{7, 1, 3, 9, 2, 4, 8, 5, 6},
 	
{9, 6, 1, 5, 3, 7, 2, 8, 4},
{2, 8, 7, 4, 1, 9, 6, 3, 5},
{3, 4, 5, 2, 8, 6, 1, 7, 9}
};
	2	Columns 1 and 3 are incorrect

board = {
{9, 5, 3, 2, 1, 4, 7, 6, 8},
{2, 7, 6, 8, 5, 3, 4, 1, 9},
{8, 1, 4, 6, 7, 9, 5, 3, 2},
 	
{7, 4, 8, 5, 3, 1, 6, 9, 2},
{6, 9, 1, 7, 4, 5, 2, 8, 3},
{5, 3, 2, 9, 6, 8, 1, 7, 4},
 	
{1, 6, 9, 4, 8, 5, 3, 2, 7},
{3, 2, 5, 1, 9, 7, 8, 4, 6},
{8, 4, 7, 3, 2, 6, 9, 5, 1}
};
	6	Column 6, 9, 1, and 2 are incorrect as well as the middle and middle right box

Problem 5: Smudged Barcodes (4 points)

New Wave Computers'™ places a five-digit barcode on each of their products that looks like the following:



Each digit in the barcode is comprised of a combination of five full bars '|' and half bars ':'. Each digit is separated by a single space.

To decode a digit, convert each full bar to a 1 and each half bar to a 0. Then, use the table below to turn the sequences of 0s and 1s into an integer.

	7	4	2	1	0
1	0	0	0	1	1
2	0	0	1	0	1
3	0	0	1	1	0
4	0	1	0	0	1
5	0	1	0	1	0
6	0	1	1	0	0
7	1	0	0	0	1
8	1	0	0	1	0
9	1	0	1	0	0
0	1	1	0	0	0

Note that they represent all combinations of two full and three half bars. Each digit can be computed easily from the bar code using the column weights 7, 4, 2, 1, 0. For example, 01100 is

0 * 7 + 1 * 4 + 1 * 2 + 0 * 1 + 0 * 0 = 6

The only exception is 0, which would yield 11 according to the weight formula.

Barcodes on several items in New Wave's warehouse have been smudged making it impossible to read all full bars and half bars. You are to write a program that reads in smudged barcodes as a String of |'s, :'s, spaces and underscores '_', an underscore represents a smudged full bar or half bar, and returns a collection of all possible barcodes that the smudged barcode could represent. For example if your method was passed the String

|:|:: :|:|: ||::: :::|| :|:_|

The first four digits in this barcode would evaluate to 9501. The missing character in the last digit would have to be a half bar since each digit must consist of three half bars and two full bars, which would make the only possible barcode for this smudged String 95014. If however your method was passed the String

|:|:: :|:|: ||::: :::|| :_:_|

the final digit could be either a 1 or a 4 so there would be two possible barcodes that you could generate from this String, 95011 or 95014. **There will not be more than three underscores in a single digit of the barcode.**

Input

String barCode - A String of |'s, :'s, spaces and underscores '_'. An underscore represents a smudged full bar or half bar.

Output

Java - An ArrayList of Strings representing all combinations that could be generated from the input barCode. Barcode must be sorted in ascending order.

Python - A list of Strings representing all combinations that could be generated from the input barCode. Barcode must be sorted in ascending order.

C++ - An array of Strings representing all combinations that could be generated from the input barCode. Barcode must be sorted in ascending order.

Sample Data

Input	Output
|:|_: ||_:: |_:_| ||_:_ :|__|	90704
|___: ||__: |___: ::___ :|::|	00014, 00024, 00034, 00814, 00824, 00834, 00914, 00924, 00934, 80014, 80024, 80034, 80814, 80824, 80834, 80914, 80924, 80934, 90014, 90024, 90034, 90814, 90824, 90834, 90914, 90924, 90934
|_:_: ||_:: |___: ::_|_ :|_:|	00014, 00034, 00814, 00834, 00914, 00934, 80014, 80034, 80814, 80834, 80914, 80934
|_|_: |__:: |___| |__:_ :___|	90701, 90702, 90704, 90771, 90772, 90774, 90791, 90792, 90794, 99701, 99702, 99704, 99771, 99772, 99774, 99791, 99792, 99794

Problem 6: Special Summations: The Unsolvable Problem (4 points)

New Wave's hiring department has designed a new question for when they interview programmers. However, they can't solve the problem. Given a positive integer, n, they would like to know all unique combinations of positive numbers that would add up to n. A summation cannot contain more than one of each digit, that is, you cannot have a summation that uses two 1s or two 2s, etc. These summations should be given in ascending order.

Input
An integer, **n** with a maximum value of 20

Output
All summations of **n**. Each summation is a sequence of numbers that, when added together, result in **n**. A summation cannot contain more than one of each digit, that is, you cannot have a summation that uses two 1s or two 2s, etc. Also, the numbers in the sequence must be arranged from smallest to largest. Each summation should be represented as a string with the numbers separated by spaces and addition signs.

Java - An ArrayList of Strings representing all unique summations for n.

Python - A List of Strings representing all unique summations for n.

C++ - An array of Strings representing all unique summations for n.

Note
Your output should look exactly like the samples with no extraneous spaces or addition signs.

For this problem only, the runtime can be greater than 10 seconds. If your solution times out, tell a judge.

Sample Data

Input	Output
5	[“1 + 4”, “2 + 3”]
8	[“1 + 7”, “2 + 6”, “3 + 5”, “1 + 3 + 4”, “1 + 2 + 5”]
12	["1 + 11", "1 + 2 + 9", "1 + 2 + 3 + 6", "1 + 2 + 4 + 5", "1 + 3 + 8", "1 + 4 + 7", "1 + 5 + 6", "2 + 10", "2 + 3 + 7", "2 + 4 + 6", "3 + 9", "3 + 4 + 5", "4 + 8", "5 + 7"]

Problem 7: Diecisiete (8 points)

New Wave Computers' Gaming Division's newest release will be the game of Diecisiete. This game is played using a modified Uno deck. The object of the game of Diecisiete is to beat the computer in one of the following ways: reach a final score higher than the computer without exceeding 17 or be dealt 5 cards without exceeding a value of 17.

Rules

At the start of the game the player and computer are each dealt two cards. The computer and player take turns throughout the game either drawing another card or staying. Once the computer or player has decided to 'stay' they can no longer 'draw' additional cards. Play will continue until the the other contestant either 'stays', is dealt five cards or exceeds a sum of 17. The player will be given the option to either stay or draw throughout the game. The computer should continue to draw until it's hand has a total greater than 13 or it can see that the player has exceeded 17. The computer can not 'see' the first card in the player's hand so the summation that the computer uses to to determine if it should draw or stay should not take this first card into account.

The modified Uno deck contains 80 cards. There are four suits: red, green, yellow and blue. Each color consists of one **0** card, two **1**s, two **2**s, two **3**s, two **4**s, two **5**s, two **6**s, two **7**s, two **8**s and two **9**s. The player's and dealer's score is calculated by summing the values of each card in their hands. When calculating this sum each rank card has a value equal to its rank.

During game play you cannot see the dealer's first card but will be able to see any card after this. Cards should be randomly dealt and you should keep track of which cards have already been dealt so that it is not dealt again during a game.

An asterisk should represent the dealer's first card. Wilds should be represented with just a 'W'. All other cards should be represented by a combination of either the letter 'R', 'B', 'G' or 'Y' and it's 1 through 9 value.

Generating a Random Number – The following code will generate a random number between 1 and `maxNumber`.

Java

```
int index = (int) (Math.random()*maxNumber-1);
```

Python

```
import random
index = random.randint(1, maxNumber)
```

C++

```
/*must import time.h */
/* initialize random seed: */
srand (time(NULL));
/* generate secret number between 1 and maxNumber: */
index = rand() % maxNumber + 1;
```

Note

This problem will be graded by a real-life developer. The rubric by which it will be graded is at the back of this packet. Note that a non-working or partially-working project can still receive points for design and organization. To receive full credit you should handle user input errors.

Sample game 1:

```
Welcome to Diecisiete.

Computer’s hand: * B3
Your hand: G6 G9

The computer has chosen to draw another card.

Computer’s hand: * B3 B1
Your hand: G6 G9

Enter a 1 to draw or a 2 to stay.
1

Computer’s hand: * B3 B1
Your hand: G6 G9 Y2

The computer has chosen to draw another card.

Computer’s hand: * B3 B1 G7
Your hand: G6 G9 Y2

Enter a 1 to draw or a 2 to stay.
2

Computer’s hand: Y5 B3 B1 G7
Your hand: G6 G9 Y2

The computer’s total is 16 Your total is 17

You have more points than the computer and didn't go over 17.  You win!
```

Sample game 2:

```
Welcome to Diecisiete.

Computer’s hand: * G6
Your hand: B6 R7

The computer has chosen to draw another card.

Computer’s hand: * G6 G8
Your hand: B6 R7

Enter a 1 to draw or a 2 to stay.
2

The computer has chosen to stay

Computer’s hand: Y0 G6 G8
Your hand: B6 R7

The computer’s total is 14 Your total is 13

The computer wins.
```

Sample game 3:

```
Welcome to Diecisiete.

Computer’s hand: * Y8
Your hand: B1 B4

The computer has chosen to draw another card.

Computer’s hand: * Y8 B5
Your hand: B1 B4

Enter a 1 to draw or a 2 to stay.
1

Computer’s hand: * Y8 B5
Your hand: B1 B4 Y0

The computer has chosen to stay

Computer’s hand: * Y8 B5
Your hand: B1 B4 Y0

Enter a 1 to draw or a 2 to stay.
1

Computer’s hand: * Y8 B5
Your hand: B1 B4 Y0 G3

The computer has chosen to stay

Computer’s hand: * Y8 B5
Your hand: B1 B4 Y0 G3

Enter a 1 to draw or a 2 to stay.
1

Computer’s hand: * Y8 B5
Your hand: B1 B4 Y0 G3 B0

The computer has chosen to stay

Computer’s hand: * Y8 B5
Your hand: B1 B4 Y0 G3 B0

Enter a 1 to draw or a 2 to stay.
2

Computer’s hand: Y4 Y8 B5
Your hand: B1 B4 Y0 G3 B0

The computer’s total is 17 Your total is 8

You got 5 cards and didn't go over 17.  You win!
```

Sample game 4:

```
Welcome to Diecisiete.

Computer’s hand: * B0
Your hand: G7 R5

The computer has chosen to draw another card.

Computer’s hand: * B0 R8
Your hand: G7 R5

Enter a 1 to draw or a 2 to stay.
r
Invalid Input.  Enter a 1 to draw or a 2 to stay.
Enter a 1 to draw or a 2 to stay.
3
Enter a 1 to draw or a 2 to stay.
1

Computer’s hand: * B0 R8
Your hand: G7 R5 G1

The computer has chosen to draw another card.

Computer’s hand: * B0 R8 B3
Your hand: G7 R5 G1

Enter a 1 to draw or a 2 to stay.
1

Computer’s hand: * B0 R8 B3
Your hand: G7 R5 G1 R0

The computer has chosen to stay

Computer’s hand: * B0 R8 B3
Your hand: G7 R5 G1 R0

Enter a 1 to draw or a 2 to stay.
2

Computer’s hand: Y3 B0 R8 B3
Your hand: G7 R5 G1 R0

The computer’s total is 14 Your total is 13

The computer wins.
```

Diecisiete Rubric:

	0 points	1 point	2 points
Function	A program solution is submitted but fails to compile. -- or - The submitted program compiles successfully. * The program does simulate playing the game of Diecisiete.	The submitted program compiles successfully. * The submitted program includes run-time and/or logic errors that result in incorrect output. * Implementation is incomplete.	The submitted program compiles successfully. * The submitted program is free of run-time and logic errors. * Your code fully implements the game of Diecisiete.
Code Readability	Code contains no documentation. * Code is unformatted and is difficult to read. * variables are ambiguous (i.e. x) and do not indicate the purpose of the variable.	The submitted solution is inconsistently documented. * Code is inconsistently formatted and can be difficult to read. * Numerous variables are ambiguous (i.e. x) and do not indicate the purpose of the variable.	The submitted solution is well documented. * Code is properly formatted (i.e. indentation within brackets and appropriate spacing) and is easy to read. * All variables are self-documented (i.e. named in a way that the name indicated the purpose of the variable).
Design	Code shows little to no thought about design. * Structures and data types are poorly chose..	Code shows some thought about design. * Appropriate structures are often used but not consistently throughout the program.	The program effectively chooses and implements concepts that would best model and solve the problem. * Appropriate data types and data structures are chosen for all variables. * Code uses the most appropriate structures (i.e. if/else if/else, methods, loops and classes)
Interface	The user interface demonstrates appropriate spacing and descriptive instructions. * Cards are not displayed correctly. * Game flow is incorrect..	The user interface includes minor spacing problems which result in inconsistent or confusing input/output. * Cards are displayed, but there may be some inconsistencies or errors. * Game flow is inconsistent.	The user interface demonstrates appropriate spacing and descriptive instructions. * Cards are properly displayed. * Game flow is correct. * The program appropriately handles when a wild card is dealt to the computer or the player. * The program effectively handles invalid user input.