CodeLM 2017

Intermediate Division

Combination Lock

Rules: You will have until 12:15 to complete this question. This question will be worth up to 12 points. Eight points will be awarded based on the rubric as graded by developers from SIG. Four points will be awarded to the team whose algorithm is able to demonstrate that it can determine the combination to the lock in the fewest amount of guesses through a benchmark test that will be run while lunch is being served. Two points will be awarded to the team that comes in second place, and one point will be awarded to the team that comes in third place.

Starter code should be downloaded from codeLM.com. **Do not change the names of the classes in this starter code or method headers.** If you do your code will not compile.

When writing the code, only code written in the correct place (as marked by comments) will be evaluated. The class's implementation for the evaluate method, the main method and any other code that you may include throughout the class will not be included. You may write other classes to help you solve this problem if you would like to. All class files will be uploaded through codeLM.com when you are finished.

Combination Lock

New Wave Computers keeps their classified information inside a highly secure vault. The vault is guarded by a four digit combination lock. Each of the four digits in this lock ranges from 0 to 5, inclusive. For unknown reasons, the lock will give feedback regarding the accuracy of each attempted guess that you make. If your guess has a **correct digit in the correct place, a '0'** will be returned to you. If **a digit appears in your guess that is also in the combination, however this digit is in the wrong location, a '1'** is returned. For each digit in your guess which does not appear in the combination, a dash '-' is returned.

Mr. Swope has "asked" you to develop an algorithm that will help him determine this combination in as few tries as possible. Because of how important it is that this algorithm be developed quickly, you will be provided with starter code that will allow you to test the efficiency of your algorithm. This starter code contains three methods. **`guess`**, which is passed the last guess that was made and an evaluation string for the last guess that was made. You will modify this method so that it returns a better guess. The other two are **`evaluate`** and **`main`**. You will not modify these two methods. In the starter code the first guess that will be made will have a value of "9999" and will have an evaluation of "----".

Sample Data

Combination	Guess	Feedback	Explanation
0243	4215	10--	The first digit (4) in your guess appears in the combination, but is in the wrong position. The second digit in your guess (2) is the right number and is also in the right position. The last two digits in your guess do not appear in the combination so the evaluation string ends with two dashes.
0243	0234	0011	The first two digits in your guess are both correct. The last two digits in your guess appear in the combination but are in the wrong location.
0534	1552	-0--	The first and last digits in your guess are not in the combination so they will return dashes. The second digit is correct so it will return a 0 in the feedback string. The third digit (5) does appear in the combination, however since the 5 was already correctly guessed as the second digit, it will not count again and will also return a dash.

Combination Lock Rubric:

Category	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 attempt to determine the correct combination..	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 a valid algorithm for determining the combination .
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 are chosen for all variables. * Code uses the most appropriate structures (i.e. if, else if, else, methods and loops.)
Algorithm	Your program does make guesses as to what the combination is, however it does so with very little accuracy or thought. It does not take past guesses into account when making its next guess.	Your algorithm makes guesses about the combination based on feedback about the last guess, but does so with little accuracy. * Your algorithm shows some thought and design.	Your algorithm makes guesses about the combination based on the feedback of several past guesses. * Your algorithm shows that it has been well thought out and designed. * Your algorithm approaches the correct combination as it continues to make guesses.