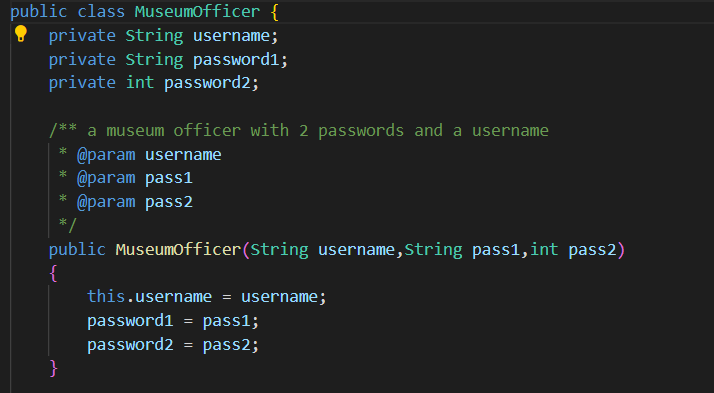
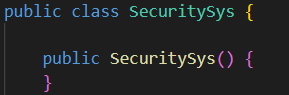
**200104004049 – Hüseyin Emre Sekanlı HW4 Report**

**A MuseumOfficer class to hold username and passwords.**

****

**A Security System class to check for validations.**

****

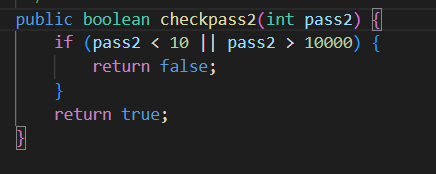
**First Check-** A function that checks if the given password has length more than 1 , doesnt have any other characters than brackets and letters, and has letter count more than 1

Time Complexity = O(n) ->Function iterates through the string on 1 loop

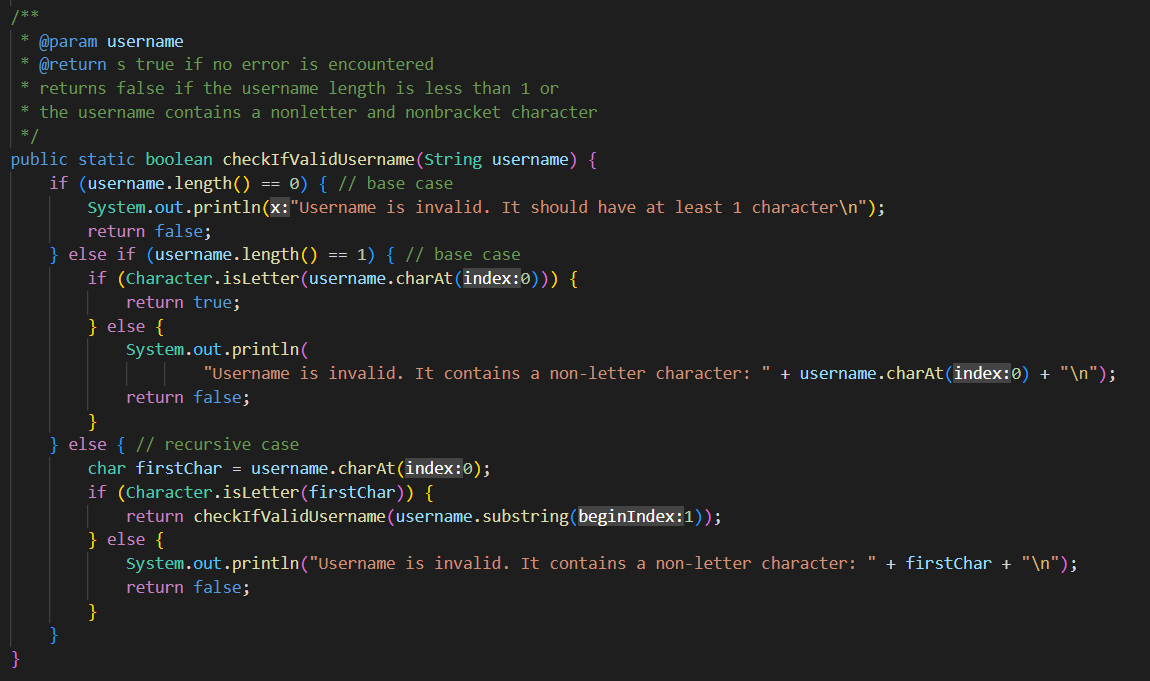
****

**Second Check-**Checking if the password2 is within 10 and 10000

Time Complexity = O(1)

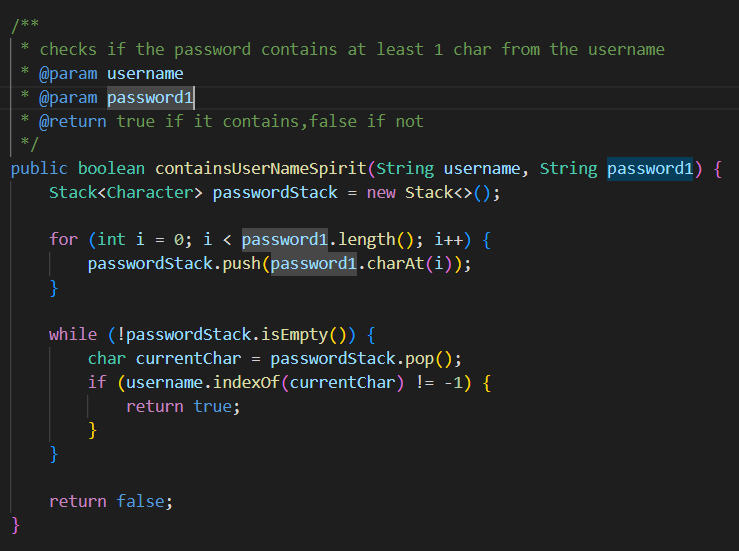
****

**CheckIfValidUsername –** A recursive function that checks if the username has more than 1 characters and all of its characters are letters. Error message printed here (i normally printed error messages outside the check functions when i called them) to avoid complex operations.

Time Complexity = O(n) -> The function visits every character in the string one by one.****

**ContainsUserNameSpirit –** pushes the password values into stack.Pops them one by one to see if a match in the username is found.

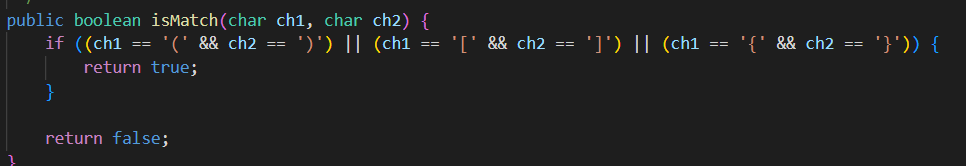
Time Complexity = O(n\*m) -> Function first iterates through string password1 then iterates through stack passwordStack.

****

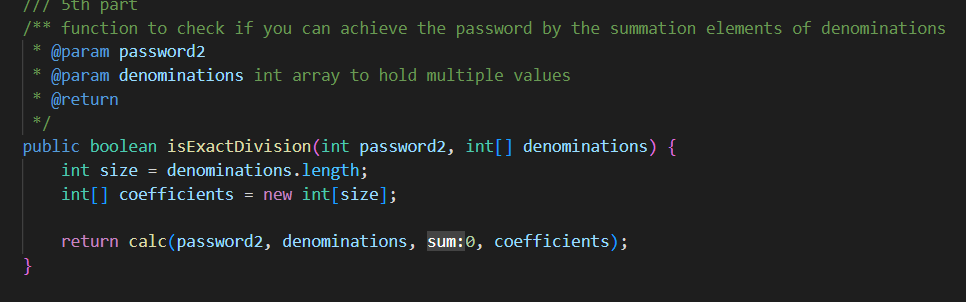
**IsBalancedPassword –** The function first identifies open and closed brackets.It then stores the open brackets in a stack.If a closed bracket is encountered,it goes to the isMatch function to see if the 2 brackets are of the same type..If the stack is empty on a closed bracket found,it returns false.

isMatch updates the balanced boolean value and the function returns balanced && p.empty() .If balanced is true and the stack is emptied of brackets,it returns true.If the stack isnt emptied or the balanced isnt true,it returns false.

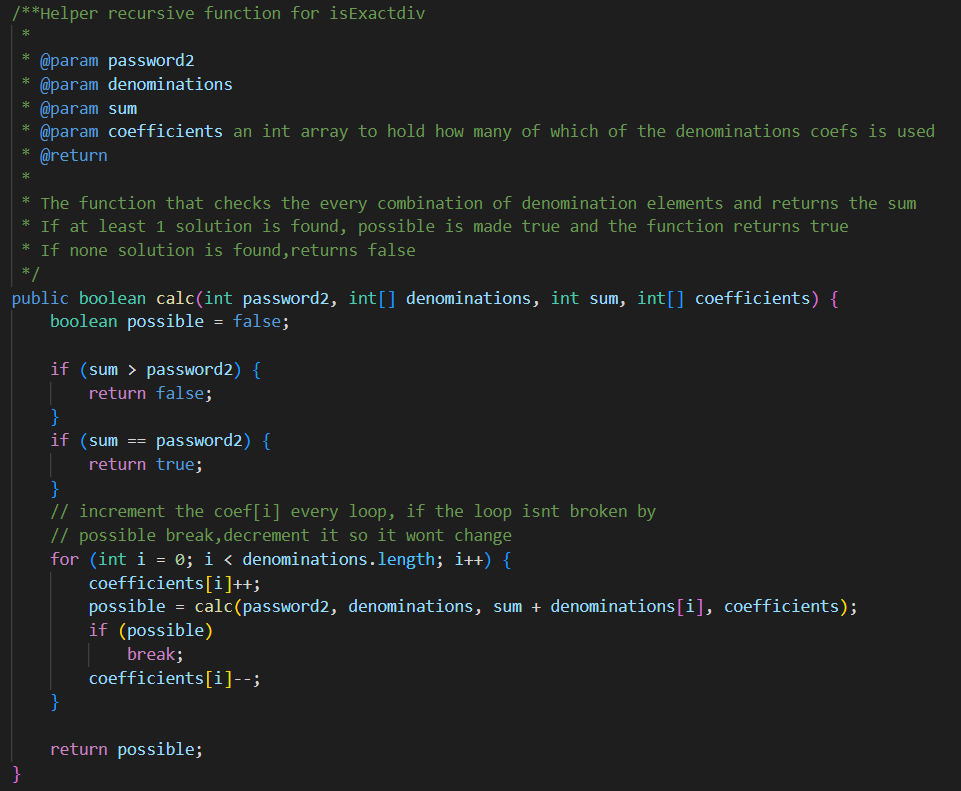
Time Complexity = O(n) -> The function iterates through the loop,adds the open brackets to the stack and does the other things inside the current loop.

**** ****

**IsExactDivision –** Function takes 2 parameters, password of the officer and the denominations to check for possible exact divisions.It then sends it to a recursive helper array.

****

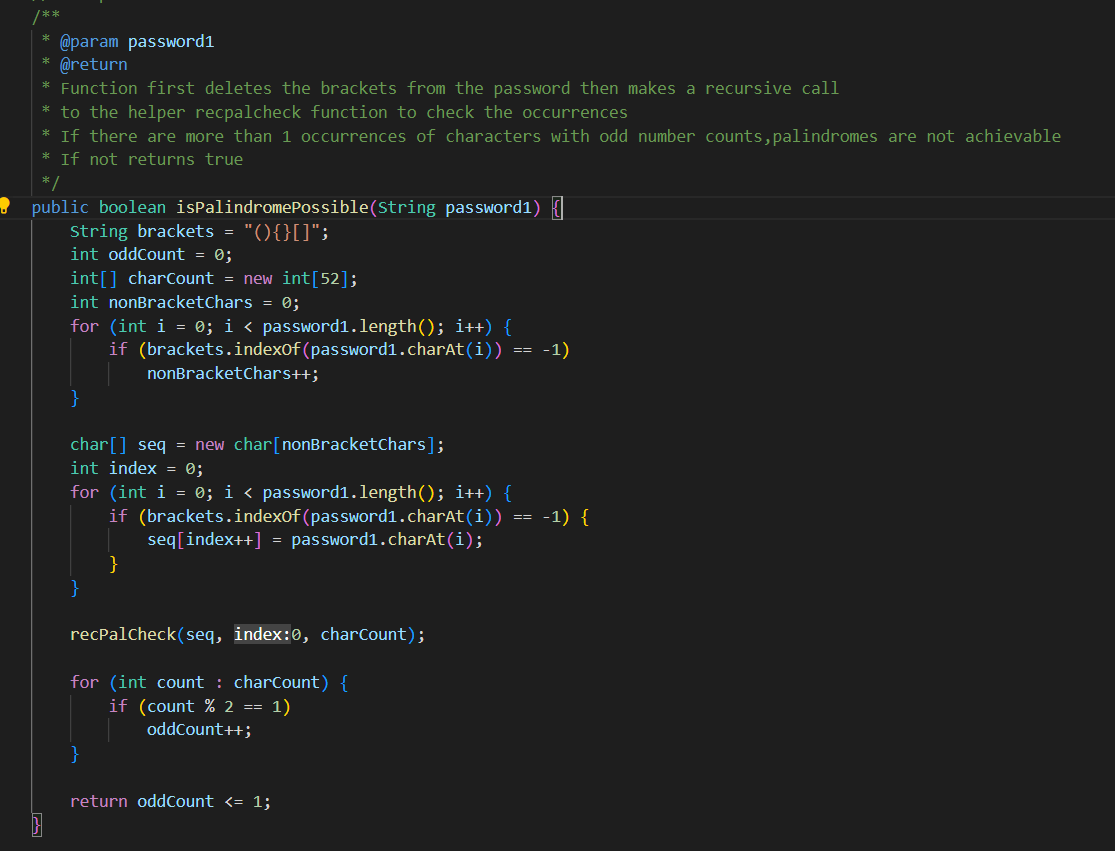
This helper array checks for all possible combinations of the given denominations and returns a boolean value based on it.

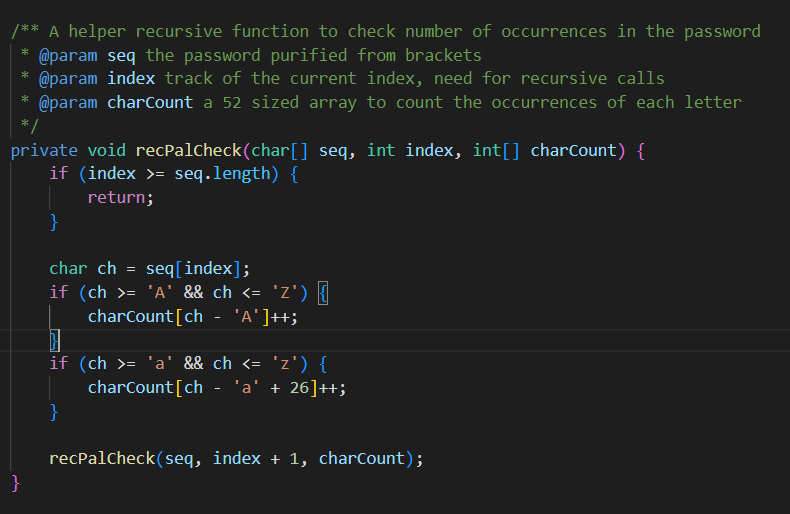
Time Complexity = O(n^m) -> The function iterates each n (length of denoms) m times (maximum recursive call count).

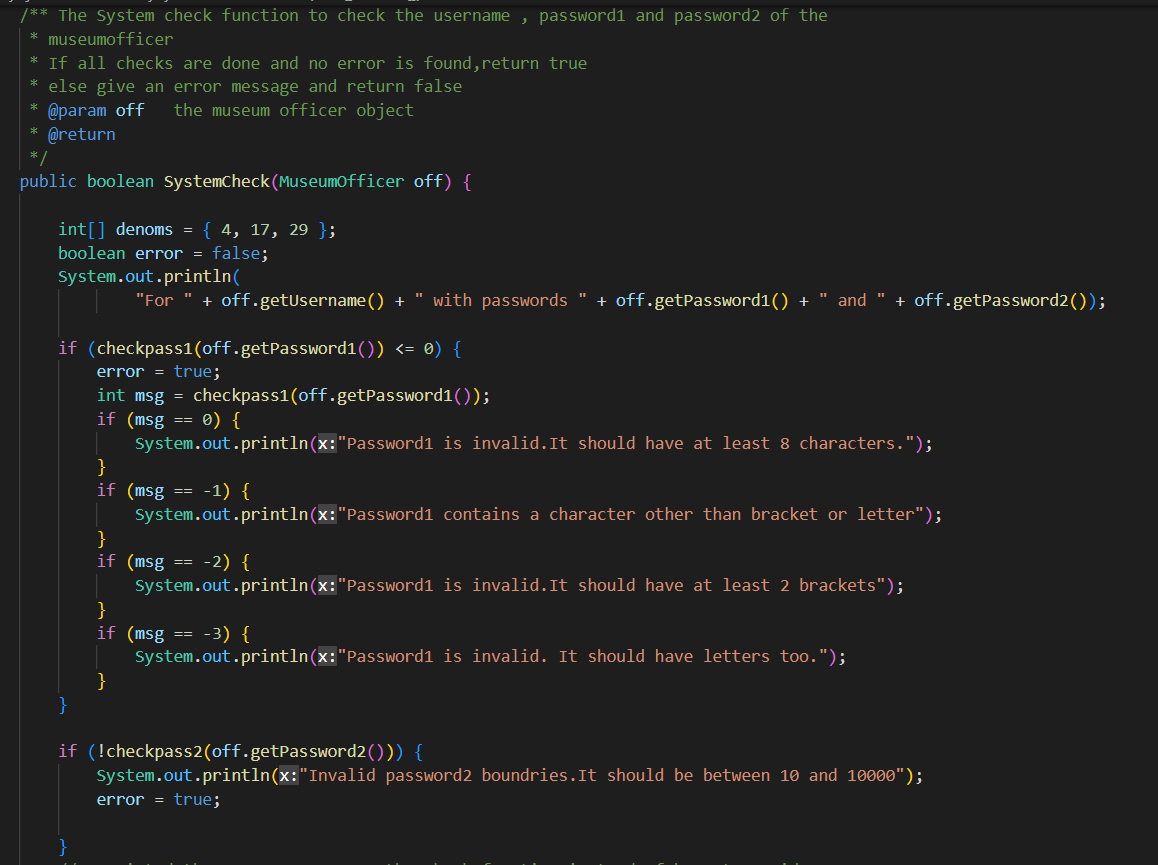
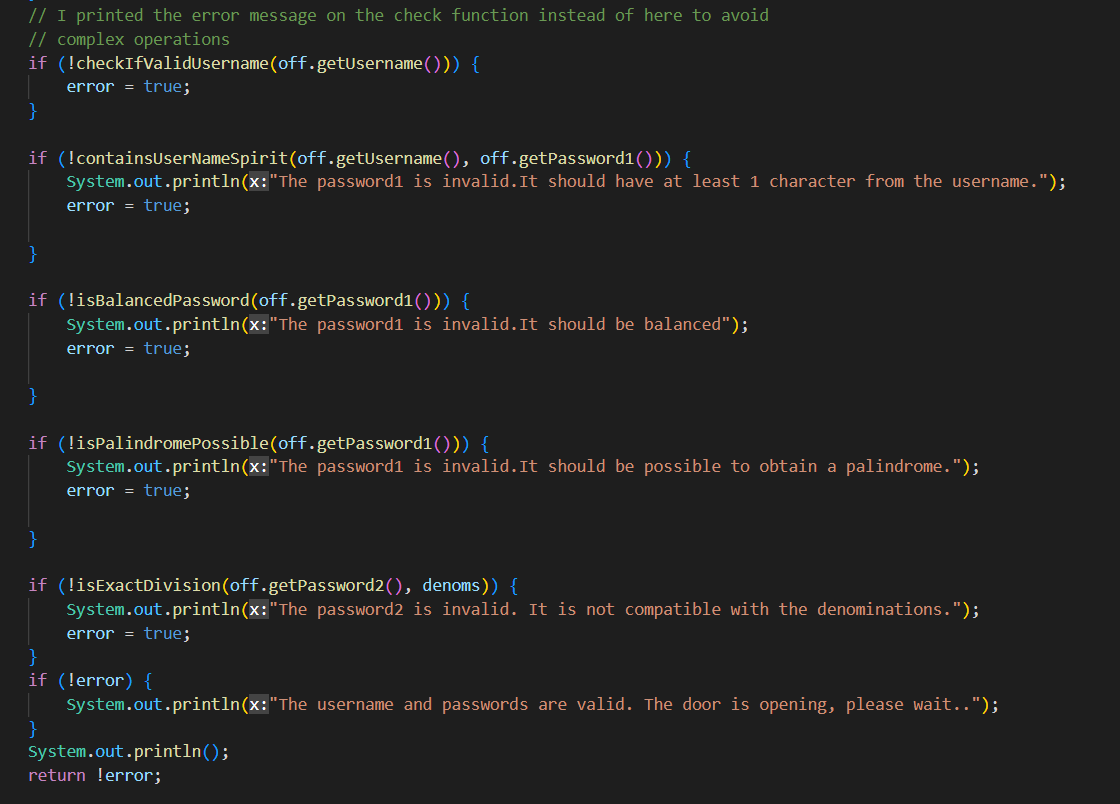
**IsPalindromePossible –** The function first counts the brackets from the password.It then creates an array sized of the password-brackets length and fills it.It then calls a helper function to count the occurrences of the each letter in the password.If the odd number of occurrences is more than 1 , palindrome can not be achieved.

Time Complexity = O(n) for first loop + O(n) for second loop +

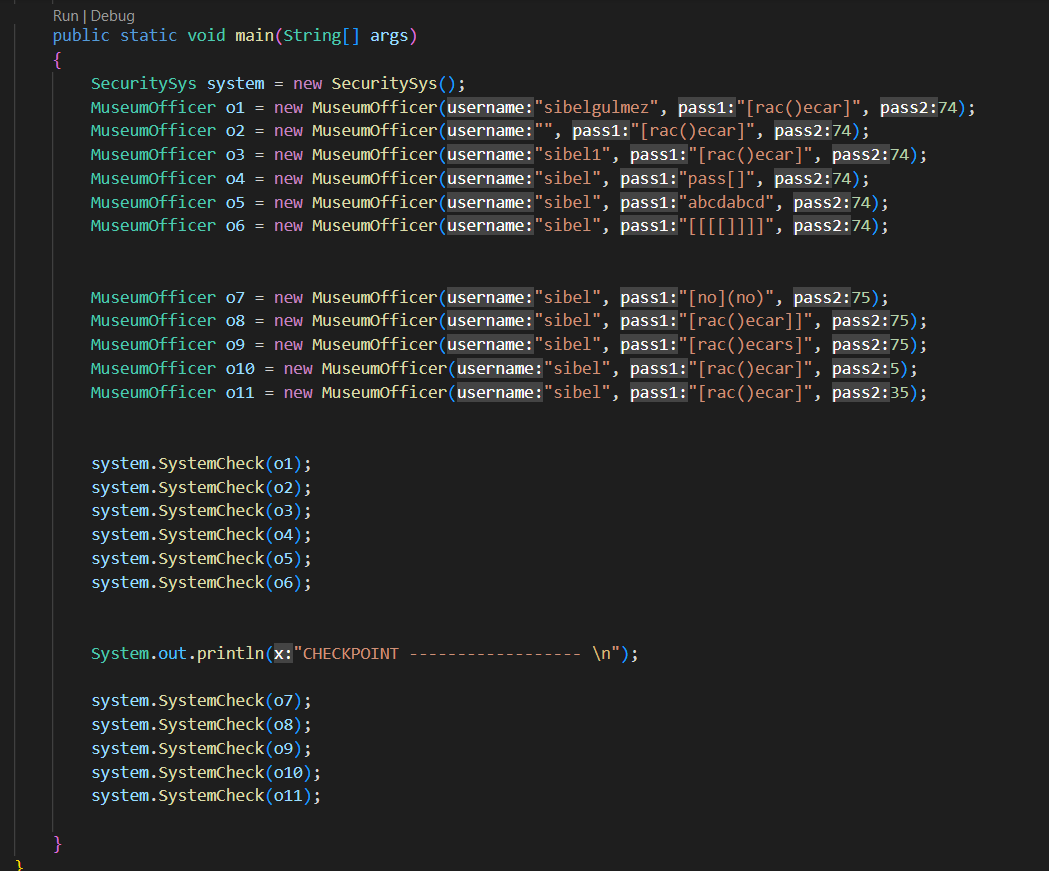
The recPalCheck worst case scenario is O(n) –no brackets in the initial password- + the loop iterating the occurrences count which has size 52.O(n) + O(n) + O(n) + O(1) = O(n)

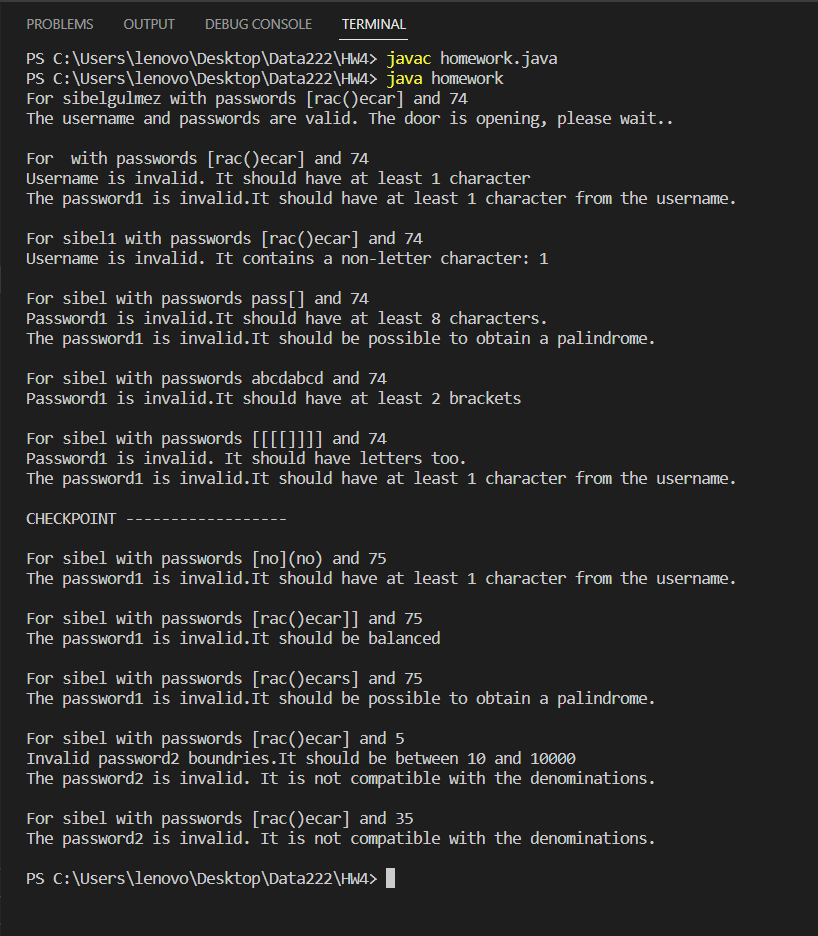
****

Helper recPalCheck – counts the occurrences of each letter in a recursive call.

**SystemCheck –** A function that checks for every possible error and prints the related error messages.If no error is found, it prints both are valid opening door ... Initially boolean error value is defined as false,if an error is found , it is converted to true. O(1) 

**Inputs and Outputs of the code**

**Test Class** ****

**Running commands and outputs : **