**CO322: Data Structures and Algorithms**

**Lab 03: HR problems set 02**

**WIJERATHNE E.S.G**

**E/18/397**

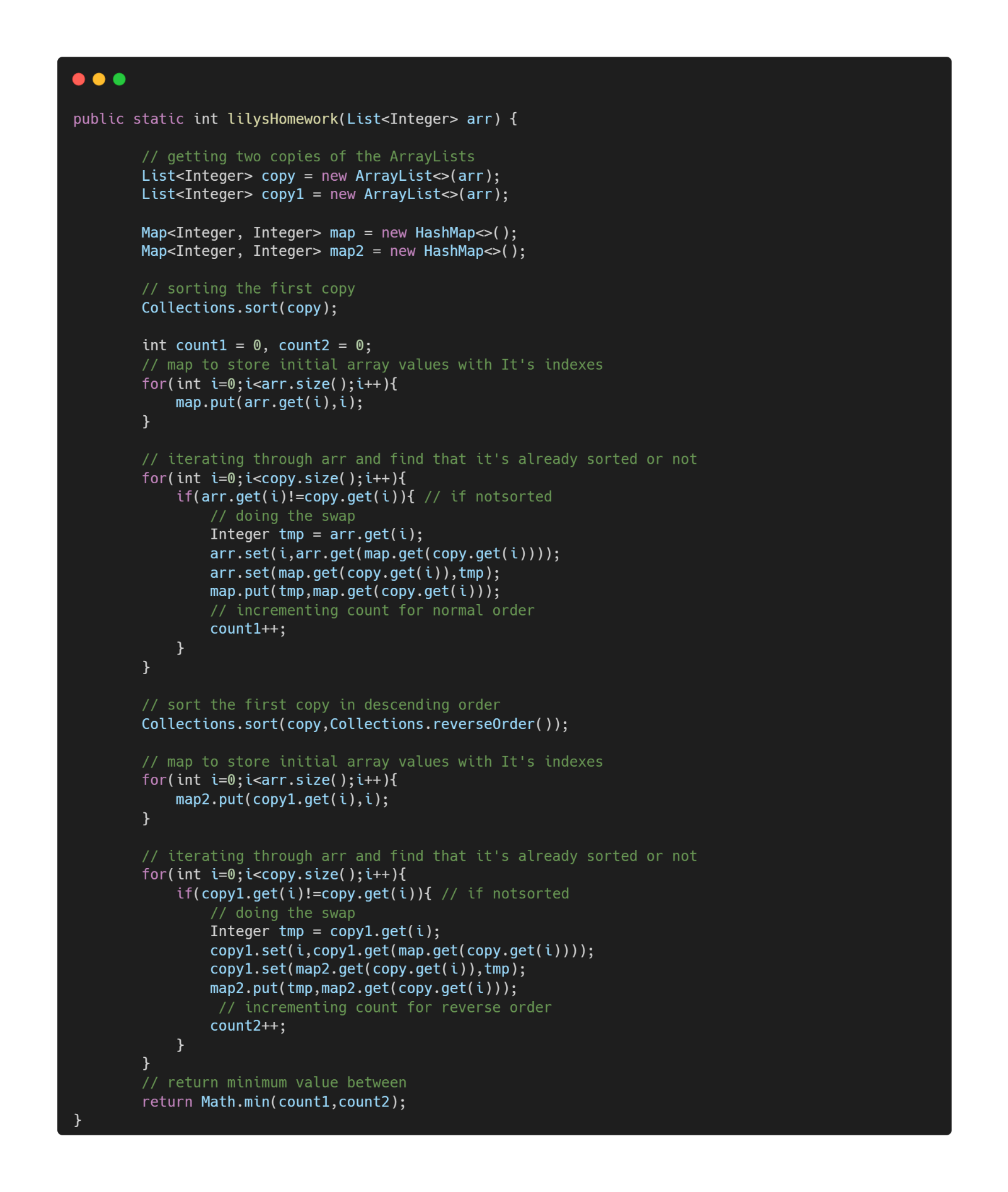
01. Closest Numbers

Text

Description automatically generated

First, I sorted the array and searched through the list to find the shortest difference between the two elements. Used break statements to avoid unnecessary loops. Once I found the minimum difference value, I again went through the list and checked for the nearest element, which gave the same difference as the low value. Once I find an occurrence, I will add both closest values to the output Integer List.

02. Lily's Homework



First, I have to make two copies of the initial array for my future use. Then I initialized two HashMap also. A copy of the initial array was sorted. Then add data to one of my HashMaps. The data was the initial array element as the key and the index of that element as the value. Then iterated through the array and checked whether the array is sorted or not. If not swapping was done and incremented the swapping count. After that, redo the above procedure with the initial array and reversed version of the sorted initial value. It will also give another answer according to the reverse sorted array. From I returned the smallest value as the final answer from both answer.

03. Fraudulent Activity Notifications

Text

Description automatically generated

A restriction that states that no element in the input Array, including any daily debit values, may have a value greater than 200 can be seen in the function description. We can construct the frequency Array needed for counting sort thanks to this relatively tiny amount.

The sum is a counter variable that we have. This will show the cumulative sum over all iterations of the values in counts. For counting sort, calculating an accumulative sum is a prerequisite. Using a for loop, counts are iterated over, with each iteration adding the current number to the sum.

 Within the loop, there are two conditionals—two if statements—that, if true, cause the loop to end and a return value to be generated. Multiplication is employed to detect if a day is odd or even rather than division, which is more computationally expensive and can lead to issues like floating point decimals. The multiplier we'll use is 2 since the main function checks to see if the median times two (\* 2) is more than or equal to the daily debit, and we want to twice the value we're evaluating.

Days may be divided by two in an even number of equal parts, hence if total \* 2 equals days on any iteration, we can conclude that sum is even. We may set days as the upper limit for sum because of the way counting sort hashes indices. This breaks the loop and aims for the current index I which corresponds to the value from the debits Array and, as a result, corresponds to the median value of the current window.

Once I have done with the calculations for single expenditures[i] element I remove the old element frequency from the array and add a new element frequency to the array.

04. Project Euler #22: Names scores

Text

Description automatically generated

First, I define string for the capital alphabet, which I will use to get indexes of the characters. Then I get user strings according to the need. Once I have the string list, I will sort it using Collections.sort(). Then I get the Q value from the user and create a loop to get strings according to that Q value. Once I get a string, I will get the current position of the string in the sorted list. Then I go through that string character by character and sum up values according to each character. Once I have the sum, I will multiply it by the position of the string in the sorted list. This will be my output.