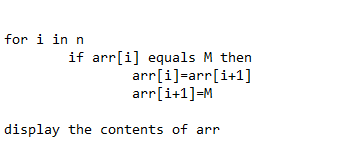
// Assignment 1  
// © Rajat Sharma  
// Written by: Rajat Sharma(ID- 40196467)

**Assignment 1**

**Q1) a) Given an array of integers of any size, n≥1, and a number m, develop an algorithm as a pseudo code (not a program!) that would swap all numbers with value m with the subsequent numbers. You must perform this operation in place i.e., without creating an additional array and keeping number of operations as small as possible. For example, given an array [5, 4, 6, 9, 6, 3, 6], and a number 6, the algorithm will return [5, 4, 9, 6, 3, 6, 6]. Finally, your algorithm must not use any auxiliary/additional storage to perform what is needed.**

a) Pseudo Code for the problem



**b) What is the time complexity of your algorithm, in terms of Big-O?**

b) The time complexity of the given code in terms of Big-O for my code is O(n). Since only 1 **FOR** loop is used to replace the elements.

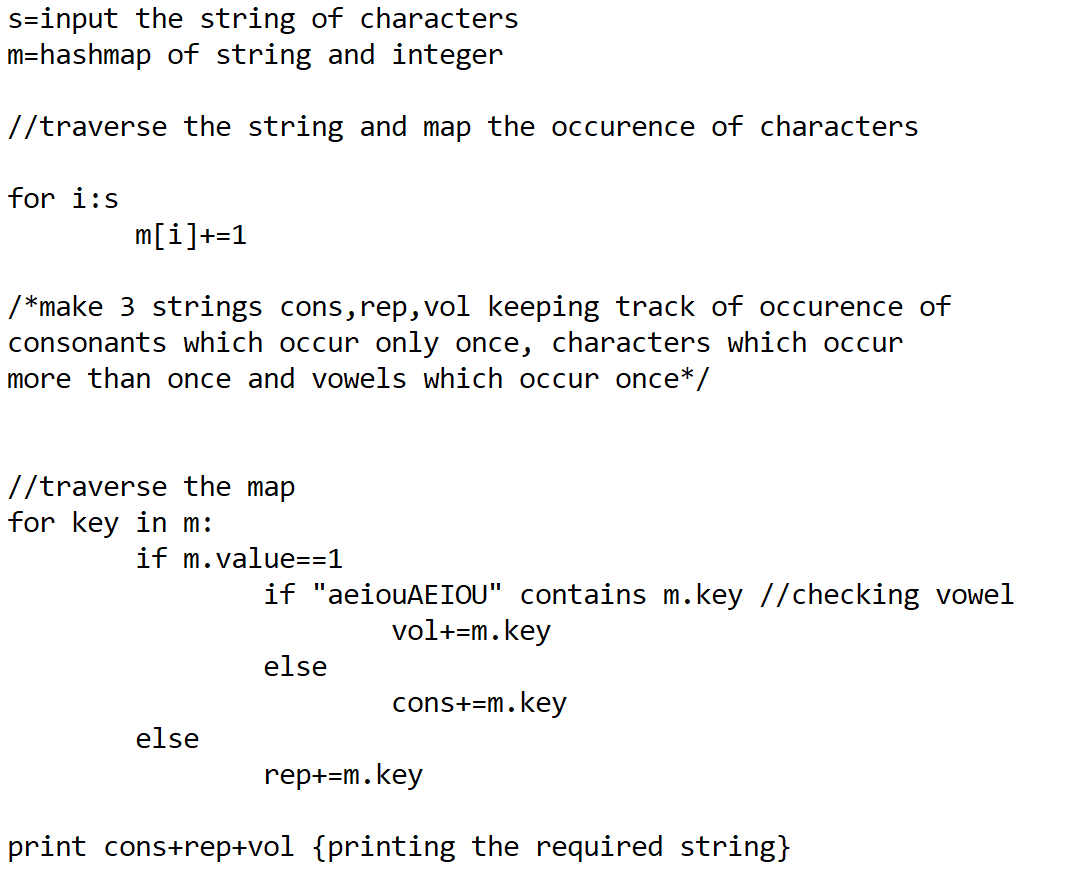
**c) What is the space complexity of your algorithm, in terms of Big-O?**

c) The space complexity for my code is O(n) for storing the elements in the array. No extra constant space is used as the swapping is in place without using extra variable.

**Q2) Given a string of random length and random contents of characters, that do not include special characters, write an algorithm, using pseudo code that will rearrange the string with all its consonants followed by all its repeating characters followed by all its vowels. For instance, given “assignment1” the algorithm must return: “gmtsn1aie”.**

**Ans2)**

Pseudo Code



**a) What is the time complexity of your algorithm, in terms of Big-O?**

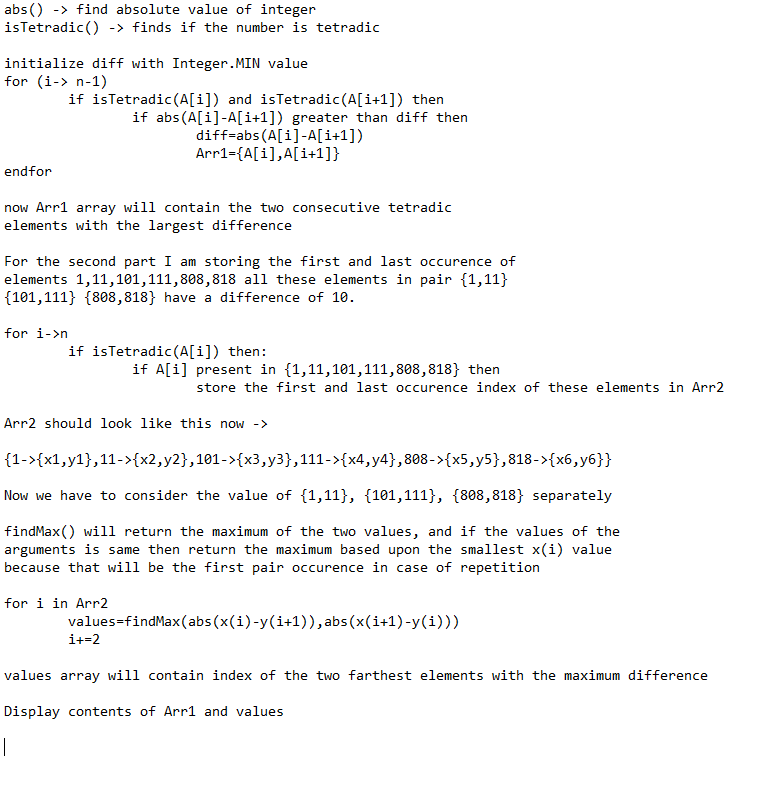
a) Time complexity of the code is O(2n) as we are traversing the array two times. This complexity is equivalent to O(n) itself.

**b) What is the space complexity of your algorithm, in terms of Big-O?**

b) The space complexity is O(m)+O(n). O(m) is the space taken by HashMap.

**Q3i) Develop a well-documented pseudo code that finds two farthest tetradic number (a four-way number) elements in an array with their difference being 10 and two successive tetradic number elements with the largest difference. The code must display values and indices of these elements. For instance, given the following array [11, 111, 88, 101, 181, 808, 1, 818, 0] your code should find and display something like the following (notice that this is just an example. Your solution must not refer to this example.) Two farthest tetradic elements with their difference equal to 10 are 11 and 1 which have 5 elements between them. Two consecutive tetradic elements with the largest difference are 818 and 0. In case of multiple occurrences of pairs with given difference or largest difference, your code must display the first found pair.**

Ans3) i



**ii) Briefly justify the motive(s) behind your design.**

For finding the two successive tetradic number elements with the largest difference a simple **FOR** loop is used to check the maximum difference between consecutive elements if a difference greater than the current difference is encountered then the **diff** variable is updated, this loops over the entire array hence making the complexity of the portion the array O(n).

For finding two farthest tetradic number elements in an array with their difference being 10, I made an analysis on the tetradic numbers and their differences. Since, tetradic numbers are palindromes and difference of 10 is just removing 1 from the ten’s place of a number since this is true… it can be observed that if 1 is removed from the ten’s place (or the second last index of a palindromic string) for the string to remain a palindrome 1 should also be removed from the 2nd position of the palindromic string in case if the 2nd last and the 2nd position of the string are not same. Since this is not happening the length of the tetradic number is reduced to less than 4 .. as having 3 as length is the only case when the middle element is both the 2nd last and the 2nd element. After this analysis it can be observed the only element having the difference of 10 are the pairs {1,11}, {101,111}, {808,818}.

So, for the computation of the second part instead of using a HashMap to store the positions of all the elements I’ll be focused on the occurrences and position of these 6 elements.

Storing the first and last positions of occurrences of these elements the answer can be computed as follows if 1->{x1,y1}, 11->{x2,y2}

Where x1,x2 and y1,y2 are the first and last position of 1 and 11 respectively

Max(x1-y2,x2-y1) should give the maximum length between 1 and 11

Computing the same of elements 101,111 and 808,818 and finding the maximum amongst them will give the final answer.

Note: We have to find the maximum only if the occurrence of the pair {1,11} is found if any of them doesn’t exist in the array we do not compute for that pair.

By doing this I am reducing the **time complexity** of my program from O(nlogn) to O(n) but also increasing the **space complexity** so a **trade-off** is being done.

**iii) What is the time complexity of your solution? You must specify such complexity using the Big-O notation. Explain clearly how you obtained such complexity.**

Time complexity of the code is O(n) for finding the consecutive elements with max difference and O(n) for finding the farthest elements with difference as 10 so the overall time complexity of the program is O(n). As for the space complexity it’s O(n) for storing the elements of the array and constant space required for storing answers. So, O(n) for space complexity as well.

**iv) What is the maximum size of stack growth of your algorithm? Explain clearly**

The stack growth of this function is linear since with the increase in input the time taken for the program to execute with increase linearly. Hence, the time complexity of the program can surely be said to be O(n).