Assignment 1

Question 1

a) Given an array of integers of any size, n ≥ 1, write an algorithm as a pseudo code (not a program!) that would reverse every two consecutive elements of the left half of the array (i.e. reverse elements at index 0 &1, at index 2 & 3, etc.). Additionally, for the right half of the array, the algorithm must change the second element of every two consecutive elements to have to the sum of the two elements. For instance, if the right half starts at index 14, and the values at index 14 & 15 are 72 and 11, then the value at index 15 is changed to 83. If the given array is of an odd size, then the exact middle element should not be manipulated by the algorithm in any way. Finally, your algorithm must use the smallest auxiliary/additional storage to perform what is needed.

Algorithm rearrangeArray(arr,n)
Input --> Array arr with n elements
Output --> Resultant array

```
if ((n % 4 = 0 AND n % 2 = 0) OR n % 4 = 3) then
        mid ← ceiling(size/2)
else if (n % 2 = 0 OR n % 2 = 1) then
        mid ← ceiling((size + 2)/2)
endif
i ← 0
while (i is less than or equal to n) then
        if (i is greater than or equal to (mid – 1)) then
                break the loop
        end if
        a ← arr[i]
        b ← arr[i+1]
        a ← a + b
        b ← a – b
        a ← a – b
        increment i by 2
end while
i ← mid
while (i is less than n) then
        if (i is greater than or equal to (size – 1)) then
                break the loop
        end if
        a ← arr[i]
        b ← arr[i+1]
        b ← a + b
        increment i by 2
end while
```

b) What is the time complexity of your algorithm, in terms of Big-O?
   - O(n), n is number of elements in the array.

c) What is the space complexity of your algorithm, in terms of Big-O?
   - O(1)

Question 2

Given a string of random length and random contents of characters, that do not include digits (0- 9), write an algorithm, using pseudo code that will shorten the representation of that string by adding the number of consecutive characters, then finds out the first found character that occurred most. For instance, given str as "gggN@@@@@KKeeeejjjjjdsmmu" the algorithm should show that the character that occurred most was @ and returns the following representation of the string: "g3N@5K2e4j5dsm2u".

Algorithm stringManupulation(str)
Input --> String with multiple characters
Output --> Resultant string

```
length ← LEN(str)
s ← ""
count ← 1
max ← 1
c ← 0
i ← 1
while (i is less than the length of the string) then
        c ← character at index (i-1)
        if (character at i = c) then
                increment count
        else
                if (max < count) then
                        max ← count
                        character ← c
                end if
                s ← s + c + count
                count ← 1
        end if
        increment i
end while
s ← s + c + count
Display (s)
Display (character)
Display (max)
```

a) What is the time complexity of your algorithm, in terms of Big-O?
   - O(n), n is the length of the string

b) What is the space complexity of your algorithm, in terms of Big-O?
   - O(1)

Question 3

i) Develop a well-documented pseudo code that finds the two consecutive elements in the array with the smallest difference in between, and the two consecutive elements with the biggest difference in between. The code must display the values and the indices of these elements. For instance, given the following array [20, 52,400, 3, 30, 70, 72, 47, 28, 38, 41, 53, 20] your code should find and display something similar to the following (notice that this is just an example. Your solution must not refer to this particular example.):

The two conductive indices with smallest difference between their values are: index 5 and index 6, storing values 70 and 73.
The two conductive indices with largest difference between their values are: index 2 and index 3, storing values 400 and 3.

Algorithm minMaxDifference(arr,n)
Input --> Array with n elements
Output --> Indices with smallest difference between their values and the difference
Indices with largest difference between their values and the difference

```
Initialize min_i, min_j, max_i, max_j to 0
min ← + infinite
max ← - infinite
diff ← 0
for i ← 0 to i ← n-1
        diff ← absolute(arr[i] – arr[i+1])
        if (diff is less than or equal to min) then
                min ← diff
                min_i ← i
                min_j ← i + 1
        end if
        if (diff is greater than or equal to max) then
                max ← diff
                max_i ← i
                max_j ← i + 1
        end if
        increment i
end for

display (“Indices with maximum difference: “, max_i, max_j)
display (“The maximum difference is:”, max)
display (“Indices with minimum difference: “, min_i, min_j)
display (“The minimum difference is:”, min)
```

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

- Initially, I have created two variables (min & max) to store minimum and maximum value. These variables have been initialized to minus (-) infinity and plus (+) infinity, respectively. Then, I have loop through the array. During each iteration, I have found the absolute difference (diff) between the values of successive indices. Then I have compared this difference with “min” variable. If “diff”

is less than “min” then this value is assigned to “min” and corresponding index values are stored in variables. Then, I have compared this difference with “max” variable. If “diff” is greater than “max” then this value is assigned to “max” and corresponding index values are stored in variables. This process is followed for all the values in the array.

iii) What is the time complexity of your solution?

- O(n), n is the number of elements in the array. We loop through the array only one time.

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

- O(1). The function call occurs only one time as the algorithm is non-recursive.