```
Assignment 1
```

```
Written by: Mukesh Kumar Angrish (Student ID - 40203596)
```

Part I

1a. Given an array of integers of any size, $n \geq 1$, and a number *m*, develop an algorithm as a **pseudo code** (not a program!) that would move all numbers with value m to beginning of the array keeping order of remaining elements in the array same. You must perform this operation in place *i.e.,* without creating an additional array and keeping the number of operations as small as possible. For example, given an array [1, 3, 2, 7, 2, 4, 2], and a number 2, the algorithm will return [2, 2, 2, 1, 3, 7, 4]. Finally, your algorithm **must not** use any auxiliary/additional storage to perform what is needed.

Given array A of n elements, our algorithm works as follows:

```
for(i=2->n)
    if(A[i] is equal to m) then
    for(j = i-1->1)
        if(A[j] is not equal to m) then
            swap(A[i], A[j])
            i = j
            if(i==1) then
                break out of the loop
            else
                set j = i-1
        else
            break out of the loop
    endfor
endfor
```

1b. What is the time complexity of your algorithm, in terms of Big-O?
Since we have two nested loops, the time complexity is $O(n*n) = O(n^2)$

1c. What is the space complexity of your algorithm, in terms of Big-O?
We're using O(1) storage for temporarily storing a value. This gets added to the storage for n numbers of the array which is O(n). Our total space complexity is O(1)+O(n) = O(n).

2. 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 swap every (i+1)th character with the (i-1)th character, starting from the second character. For instance, given a string "assignment1" the algorithm should return the string: "signment1sa".

Given array A of n elements, our algorithm works as follows:

```
for(i=2->n-1)
    temp = A[i-1]
    A[i-1]=A[i+1]
    A[i+1]=temp
endfor
```

2a. What is the time complexity of your algorithm, in terms of Big-O?
Since we loop over the array just once, the time complexity is O(n).

2b. What is the space complexity of your algorithm, in terms of Big-O?
We're using O(1) storage for temporarily storing a value. This gets added to the storage for n numbers of the array which is O(n). Our total space complexity is O(1)+O(n) = O(n).

3i. Develop a well-documented pseudo code that finds two farthest elements in the array with the difference of their digit sum equal to 1, and two consecutive elements with the largest difference in their digit sum. The code must display values and indices of these elements. For instance, given the following array [20, 52, 400, 3, 30, 70, 72, 47, 28, 38, 41, 53, 21] 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 elements with difference of their digit sum equal to 1 are 52 and 53 which have 9 elements between them. Two consecutive elements with the largest difference in their digit sum are 20 and 52. In case of multiple occurrences of the pairs with farthest distance or largest difference, your code must display the first found pair.

Given array A of n elements, our algorithm works as follows:

```
// abs() finds the absolute value of the integer

for(i=0->n-1)
    for(j=n-1->i+1)
        if(abs(sum of digits of A[i] - sum of digits of A[j]) = 1) then
            print A[i] and A[j]
            break out of both loops
    endfor
endfor

initialize max, number1, number2 = 0
for(i=1->n-1)
    if(abs(sum of digits of A[i] – sum of digits of A[j]) > max) then
        max = abs(sum of digits of A[i] – sum of digits of A[j])
        number1=A[i]
        number2=A[j]
endfor
```

3ii. Briefly justify the motive(s) behind your design.
Since, we are required to find the absolute difference between the sum of digits of two numbers, we only need a way to loop over a pair of numbers in the array. That is why I chose nested loop for finding the two farthest elements in the array with the difference of their digit sum equal to 1 and a single loop for finding and two consecutive elements with the largest difference in their digit sum.

3iii. 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.
For finding the two farthest elements in the array with the difference of their digit sum equal to 1, we use nested loops which run in O(n) time each. Therefore, total time complexity is O(n*n) = O(n)

3iv. What is the maximum size of stack growth of your algorithm? Explain clearly
Since the iterative approach is used, the stack growth is just one. This is because only one function is used to execute the algorithm

Part II A

This Vaccine.java program defines a class Vaccine with different attributes corresponding to the information regarding the Vaccine.

The Vaccine class also has multiple accessors and mutators in order to provide flexibility during object creation and modification.

There are also methods to convert object to string and to compare different Vaccine objects

The solution available in Vaccine.java file.

Part II B

VaccineDriver implements an inventory system that allows user to manage a stock of vaccines

Multiple functions:

1. entering a stock of vaccines

2. modifying its information of vaccines by number

3. displaying vaccines by brand

4. displaying vaccines less than a certain price

The solution available in VaccineDriver.java file

Also included in the submission folder are the HTML files generated from javaDocs and this pdf file.