Assignemnt1

1)

a)

```
BEGIN
   SET SWAP_NUMBER M
   SET INPUT sequence Array[N]
    FOR index=1 to N-1 STEP 1
        IF Array[index]==M Then
             Array[index+1] += Array[index]
             Array[index] = Array[index+1]- Array[index]
             Array[index+1] = Array[index+1]- Array[index]
        END IF
    END FOR
END
```

b) time complexity?

O(n)

c) Space complexity

Space Complexity = O(n) =Auxikary space O(1) + Input space O(n)

2)

a)
```
BEGIN
    SET INPUT InputString
    SET VowelString to Empty
    SET RepeatingConsonantString to Empty
    SET RemainingCharacterString to Empty
    SET OutputString to Empty
    SET a HashMap to hold a key and value
    FOR Character in InputString
        IF HashMap Has Character as KeyThen
            Increment the value by 1 in HashMap
        ELSE
            Add the Character to Hash Map and set the value to 1
        END IF
    END For
    FOR Character in InputString
        IF Character IS Vowel and VowelString Doesn't contain Character Then
            Append character to VowelString
        ELSE IF Character is Consonant
            IF HashMapValue for Character is >1
                Append Character to RepeatingConsonantString
            ELSE
                Append Character to OutputString
            END IF
        ELSE
            Append Character to RemainingCharacterString
        END IF
    END FOR
    APPEND RepeatingConsonantString ,  RemainingCharacterString , VowelString to OutputString in respective order
END
```

b) Time Complexity -> O(N)

C) Space Complexity -> O(N)

3)

```
BEGIN
 TAKE INPUT Array[N] //an array of tetradic numbers
 SET DIFFERENCE_REQUIRED=10
 SET MAX_INDEX_DIFF = 0, SUCCESIVE_ELMENT_DIFF=0
 SET LEFT_INDEX=0, RIGHT_INDEX=-0
 SET SUCCESIVE_LEFT_INDEX=0, SUCCESIVE_RIGHT_INDEX=-1
 FOR I=1 to N-2 STEP 1
        FOR J=N to I Step 1
            IF MOD of Array[I]-Array[J] is 10 and J-I > MAX_INDEX_DIFF THEN
                  LEFT_INDEX=I, RIGHT_INDEX=J, MAX_INDEX_DIFF=J-1
            END IF
        END FOR
      IF MOD Array[I]-Araa[I+1] >SUCCESIVE_ELEMENT_DIFF THEN
            SUCCESIVE_LEFT_INDEX=I, SUCCESIVE_RIGHT_INDEX=1, SUCCESIVE_ELMENT_DIFF= MOD Array[I]-Araa[I+1]
    END IF
 END FOR
 OUTPUT Furthest Element With Difference 10 are Array[LEFT_INDEX] and Array[RIGHT_INDEX]
OUTPUT Successive Elements with Max Difference are Array[SUCCESIVE_LEFT_INDEX] and Array[SUCCESIVE_RIGHT_INDEX]
 END
```

a) This is a simple brute force method, this can be modified to run in O(n) by just looking at the required numbers instead of calculating all possible possibilities for difference but it requires a HashMap to maintain the numbers to make O(1) time for lookup.
b) $O(n^2)$ -> This code uses 2 for loops to calculate all possible combinations, in best case we might can get result in first iteration itself. Since we run loop from 2 ends , and if the difference is 10 , it will be the furthest element. For worst case $O(n^2)$
c) This program uses one 1-D Array which consumes O(N) memory in stack , other variables such as left_index,right_index,max_diff etc consumes O(1) memory in stack , but overall we can say that this algorithm consumes O(n) memory on stack.