Programming Question:

You will find here below the pseudo code, complexity function and big O. (For the java source code please view the zip file submitted with this assignment)

a) Here's the pseudo code:
   1) `rearrangeParticipants()`:
      a. $f(n) = 10n^2 + 3n + 7$
      b. $O(n^2)$

```
Algorithm rearrangeParticipants(names, pDOB, n, currentIndex ←
0)
      Input: names a string array, pDOB a dates array as
string, n The total number of members and currentIndex
optional param indicates which index the sort has reached
      Output:  The number of seniors

      if currentIndex = n then
            // Count number of seniors
             numberOfSeniors ← 0
            for i ← 0 to  i < n do
                  if agepDOB[i] >= 65 then
                        numberOfSeniors++

            // Then the arrays are ordered in decreasing
order, in this case
            // The seniors are arranged properly but the non
seniors needs to be rearranged
            for i ← numberOfSeniors to  i < n do
                  for j ← i + 1 to  j < n do
                        if agepDOB[i] > agepDOB[j] then
                              swap(names, i, j)
                              swap(pDOB, i, j)

            return numberOfSeniors


       age ← age(pDOB[currentIndex])
       highestAge ← age
       highestAgeIndex ← currentIndex

       for  i ← currentIndex to  i < n do
             temp ← age(pDOB[i])
            if temp > highestAge then
                  highestAge ← temp
                  highestAgeIndex ← i

      swap(names, highestAgeIndex, currentIndex)
      swap(pDOB, highestAgeIndex, currentIndex)

      currentIndex ← currentIndex + 1

      return rearrangeParticipants(names, pDOB, n,
currentIndex)
```

2) `displaySeniorsIncreasingOrder()` :
   a. $f(n) = n + 7$
   b. $O(n)$

```
Algorithm displaySeniorsIncreasingOrder(pName, pDOB, nSenior,
displayed ← 0)
      Input: pName the names array, pDOB the date of birth,
nSenior the number of seniors, displayed optional parameter
represents how many elements were displayed
      Output: void (Variable description here)

      if displayed = nSenior then
            return
      else
            index ← nSenior - displayed - 1
            print("%s, %d\n", pName[index], age(pDOB[index]))
            displayed ← displayed + 1

            displaySeniorsIncreasingOrder(pName, pDOB,
nSenior, displayed)
```

3) `displayNonSeniorsInreasingOrder()`:
   a. $f(n) = n + 7$
   b. $O(n)$

```
Algorithm displayNonSeniorsInreasingOrder(pName, pDOB,
nNoneSenior, total, displayed ← 0)
      Input: pName the names array, pDOB the date array,
nNoneSenior the number of non-seniors, total the total number
of members, displayed the number displayed elements (Variables
description here)
      Output: void (Variable description here)

      if displayed = nNoneSenior then
            return
       else
             index ← total - nNoneSenior + displayed
            print("%s, %d using R\n", pName[index],
age(pDOB[index]))
            displayed ← displayed + 1
            displayNonSeniorsInreasingOrder(pName, pDOB,
nNoneSenior, total, displayed)
```

4) `displayIncreasingOrder()`:
   a. $f(n) = 10n^2 - 7n + 2$
   b. $O(n^2)$

```
Algorithm displayIncreasingOrder(pName, pDOB, senior, total)

      Input: pName and pDOB arrays, senior number of seniors,
total total members
      Output:  prints the array to the console in increasing
order

      // Copy arrays
      nameCopy ← copy_array(pName) // This operations is O(n)
      pDOBCopy ← copy_array(pDOB) // This operations is O(n)


      // First sort the array
      for  a ← 0 to  a <= total - 1 do
            for  b ← 0 to  b <= total – 2 do
                  if agepDOBCopy[b + 1] < agepDOBCopy[b] then
                        swap(nameCopy, b + 1, b)
                        swap(pDOBCopy, b, b + 1)

      for  i ← 0 to  i < nameCopy.length do
            print("%s, %d\n", nameCopy[i], age(pDOBCopy[i]))
```

5) Helper function. I asked the prof and she said you can assume that the date of births are integers instead of string to simplify the use of language specific functions:
   a. $f(n) = 1 \rightarrow O(1)$

```
Algorithm age(pDOB)
      Input: The date of birth
      Output: The age of member
      return 2021 - pDOB
```

b) No, the algorithm is quadratic ($O(n^2)$)

- Yes, it is tail recursion because the last statement of the function is returning that function (no other operation is made)