*Solution for Assignment 1:*

COMP-352

by

Shadi Jiha (40131284)

Concordia University

Department of Computer Science and Software engineering

20 May 2021

**Question 1:**

1. Solution

**Algorithm** question1(array, n)

**Input:** ***array*** *of Integers of size* ***n***

**Output**: sumEvenPos, sumOddPos *(Variable description here)*

sumEvenPos ← 0

sumOddPos ← 0

**for** i ← 0 **to** n - 1 **do**

**if** array[i] > 0 **then**

**if** array[i] % 2 = 0 **then**

sumEvenPos ← sumEvenPos + array[i]

else

sumOddPos ← sumOddPos + array[i]

{ increment counter **i** }

**return** sumEvenPos, sumOddPos

1. However, the auxiliary space is

**Question 2:**

1. if and only if

Suppose =. Thus,

|  |  |  |
| --- | --- | --- |
|  |  |  |
|  | 1 | 825 |
|  | 4 | 988 |

This relation is always true. This is relations is **approved.**

This function has a which means that is not valid because it does not go between the 2 above functions. Thus, this relation is **disapproved.**

1. if and only if

Suppose . Thus,

|  |  |  |
| --- | --- | --- |
|  |  |  |
|  | 1 | 1 |
|  | 1.74 | 2.042 |

this relation can only be true if . In other words, this relation is **disapproved.**

1. if and only if

This function has a and which means that is not valid because it does not go between the 2 above functions. Thus, this relation is **disapproved**

1. if and only if

Suppose . Thus,

|  |  |  |
| --- | --- | --- |
|  |  |  |
|  | 0 | 0 |
|  | 0.3010 | 4.337 |
|  | 0.477 | 15.614 |

this relation can only be true if . In other words, this relation is **disapproved**.

1. if and only if

Suppose . Thus,

|  |  |  |
| --- | --- | --- |
|  |  |  |
|  | 1 | 1.00000001 |
|  | 64 | 128.00 |
|  | 729 | 2187.00 |

This relation is always true. This is relations is **approved.**

**Question 3:**

1. Yes, we can, here’s the new algorithm:

**Algorithm** arraySpecialSum(A, n)

**Input:** *A array of numbers of size n*

**Output**: specialSum

currentMax ← A[0]

**for** i ← 1 **to** n - 1 **do**

**if** A[i] > currentMax **then**

currentMax ← A[i]

specialSum ← 0

**for** i ← 0 **to** n - 1 **do**

**if** A[i] = currentMax **then**

specialSum ← n \* (specialSum + A[i])

**return** specialSum

* The second one is much more efficient because we calculate the special sum while calculating how many max occurrences there are.

**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)

1. Here’s the pseudo code:
2. rearrangeParticipants():

**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)

1. displaySeniorsIncreasingOrder() :

**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)

1. displayNonSeniorsInreasingOrder():

**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)

1. displayIncreasingOrder():

**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]))

1. 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:

**Algorithm** age(pDOB)

**Input:** The date of birth

**Output**: The age of member

**return** 2021 - pDOB

1. No, the algorithm is quadratic ()

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