**CalendarProblem**

Problema va primi de la utilizator intervalele indisponibile a celor 2 persoane, intervalele in care cei 2 lucreaza si durata unui meeting. Programul imi transforma datele in liste de obiecte INTERVAL(pentru listele de intervale indisponibile) si obiecte INTERVAL(pentru intervalele in care cei 2 sunt la munca)**(1)**. Cele 2 liste vor fi reunite astfel incat sa obtin o lista de intervale in care cel putin o persoana nu este disponibila**(2)**. Reunesc intervalele in care cele 2 persoane sunt la munca astfel incat sa obtin intervalul in care cele 2 persoane ar putea participa la un meeting**(3)**. Programul va restrange lista de intervale dupa intervalul obtinut anterior(pentru ca ar putea exista timp liber pentru un metting dar una din cele 2 persoane sa nu este la munca)**(4)**. Acum parcurg lista de intervale obtinute, calculez timpul liber intre 2 intervale iar daca acesta este >= decat timpul pentru un meeting atunci acel interval este valid**(5)**.

**Exemplu:**

**(1):**

firstPerson: [['8:00','9:30'], ['10:30','11:30'], ['12:00','15:00'], ['16:00','17:00']]

secondPerson: [['9:30','10:30'], ['12:30','13:00'], ['14:30','15:00']]

firstPersonRange: ['8:00','17:00']

secondPersonRange: ['9:00','16:30']

meetingTime: 45

**(2):**

[['8:00','11:30'], ['12:00','15:00'], ['16:00','17:00']]

**(3):**

['9:00','16:30']

**(4):**

[['9:00','11:30'], ['12:00','15:00'], ['16:00','16:30']]

**(5):**

['15:00','16:00']

**PS:**

Programul nu functioneaza si pentru program nosturn(['22:00','02:00'] de exemplu). Daca s-ar fi specifiat in cerinta atunci as fi luat si data calendaristica in considerare nu doar orele si minutele.

**Functii folosite:**

**convertStringInDateTime** : primeste data sub forma de string si o transforma in DateTime. Pentru stringuri ca '8:00' le transforma mai intai in '08:00' apoi in DateTime.

**findMin** : primeste 2 DateTime uri si returneaza DateTime ul cu valoarea mai mica

**findMax** : primeste 2 DateTime uri si returneaza DateTime ul cu valoarea mai mare

**convertRange** : primeste un interval sub forma de string si-l transforma in obiectul Interval

**findMinimumRange** : primeste 2 intervale si returneaza intervalul orar comun (intersectia lor)

ex: pentru intervalele ['10:00','20:00'] si ['9:00','18:30'] va returna ['10:00','18:30']

**findMaximumRange** : primeste 2 intervale si returneaza intervalul maxim format de cele 2 (reuniunea lor)

ex: pentru intervalele ['10:00','11:00'] si ['10:30','12:30'] va returna ['10:00','12:30']

**ifExistRange** : primeste un interval si verifica daca acesta exista (startTime < endTime)

**timeDifference** : primeste 2 intervale si calculeaza diferenta de timp dintre ele

ex: ['10:00','12:00'] si ['14:30','18:30'] va returna 02:30

**convertTimeMeetingInDateTime** : primeste timpul pentru un meeting in minute si-l transforma in TimeSpan. ex: 70 -> 1:10 adica 70 minute se transforma in 1 ora si 10 minute

**verifyCombination** : primeste 2 intervale si verifica daca exista un interval comun intre cele 2 (adica intersectia celor 2 intervale va da un interval valid sau nu)

**convertCalendar** : parseaza stringul primit si creeaza lista de Intervale

**combineTwoRanges** : Primeste o lista si un interval. Verifica daca poate reuni ultimul interval din lista si intervalul primit. Daca o poate face atunci va elimina din lista ultimul inverval, le va reuni si apoi le va pune in lista intervalul reunit. Daca nu le poate reuni va pune doar intervalul primit in lista

[exemplu:](#ExempluCombineTwoRanges) (CTRL + Click)

**listOfInaccessibleRange** : primeste 2 liste de intervale ocupate/indispinibile si reuneste, reunind si intervalele daca este posibil

[exemplu:](#ExempluListOfInaccessibleRange) (CTRL + Click)

**fitTheRange** : primeste lista de intervale indisponibile si intervalul in care cei doi utilizatori pot participa(intersectia intervalelor in care cei 2 activeaza). Imi restange lista de intervale indisponibile in functie de intervalul primit

[exemplu:](#exempluFitTheRange) (CTRL + Click)

**findRanges** : primeste lista de intervale si intervalul de timp pentru un meeting(in minute). Parcurge lista, calculeaza timpul dintre 2 intervale iar daca e mai mare decat timpul pentru un meeting atunci salveaza acel interval ca fiind valid

[exemplu:](#ExempluFindRanges) (CTRL + Click)

**Exemplu combineTwoRanges:** [back](#combineTwoRanges)(CTRL + Click)

EX1:

Range: ['12:30','15:00']

List: [['9:30','11:00'], ['11:30','13:00']]

Intervalele ['12:30','15:00'] si ['11:30','13:00'] se pot reuni deci lista va deveni:

[['9:30','11:00'], ['11:30','15:00']]

EX2:

Range: ['13:30','15:00']

List: [['9:30','11:00'], ['11:30','12:00']]

Intervalele ['13:30','15:00'] si ['11:30','12:00'] NU se pot reuni deci lista va deveni:

[['9:30','11:00'], ['11:30','12:00'], ['13:30','15:00']]

**Exemplu listOfInaccessibleRange:** [back](#listOfInaccessibleRange)(CTRL + Click)

EX:

firstList : [['8:00','9:30'], ['10:30','11:00'], ['12:00','15:00'], ['15:00','17:00']]

secondList: [['9:30','10:30'], ['11:30','13:00'], ['14:30','15:00']]

1. Initial incearca sa reuneasca intervalele ['8:00','9:30'] si ['9:30','10:30'] si le reuneste in ['8:00','10:30'].

Lista va deveni [['8:00','10:30']]

1. Acum va incerca sa reuneasca intervalele ['10:30','11:00'] si ['11:30','13:00'] si nu reuseste. Ar trebui sa insereze in lista intervalul care incepe primul adica ['10:30','11:00'] dar mai intai va inserca sa-l reuneasca cu ultimul interval din lista adica cu ['8:00','10:30'] si va reusi asta, rezultand intervalul ['8:00','11:00']

Lista va deveni [['8:00','11:00']]

1. Acum va incerca sa reuneasca intervalele ['12:00','15:00'] si ['11:30','13:00'] si va reusi rezultand intervalul ['11:30','15:00']. Ar trebui sa insereze in lista intervalul obtinut, adica ['11:30','15:00'] dar va incerca mai intai sa-l reuneasca cu ultimul interval din lista adica cu ['8:00','11:00'] si nu va reusi.

Lista va deveni [['8:00','11:00'], ['11:30','15:00']]

1. Acum va incerca sa reuneasca intervalele ['14:30','15:00'] si ['15:00','17:00'] si va reusi, rezultand intervalul ['14:30','17:00']. Ar trebui sa-l insereze in lista dar va incerca mai intai sa-l reuneasca cu ultimul interval din lista adica cu ['11:30','15:00'] si va reusi, rezultand intervalul ['11:30','17:00'] care va fi si inserat.

Lista va deveni [['8:00','11:00'], ['11:30','17:00']]

result: [['8:00','11:00'], ['11:30','17:00']]

**Exemplu fitTheRange:** [back](#fitTheRange)(CTRL + Click)

EX:

List : [['8:00','11:00'], ['11:30','17:00']]

range: ['9:30','15:00']

Incearca sa reuneasca intervalele ['8:00','11:00'] si ['9:30','15:00'] si reuseste, intervalul obtinut fiind ['9:30','11:00']. Acesta il va inlocui pe cel din lista(cel cu care am facut reuniunea). A aparut modificare deci inceputul listei este rezolvata. Facem acelasi lucru si pentru sfarsitul listei. Incarcam sa reunim intervalele ['11:30','17:00'] si ['9:30','15:00'] si obtinem ['11:30','15:00'] iar acesta va deveni ultimul interval din lista, inlocuind intervalul care a intrat in reuniune

result: [['9:30','11:00'], ['11:30','15:00']]

**Exemplu findRanges:** [back](#findRanges)(CTRL + Click)

EX1:

List: [['9:00','11:30'], ['12:00','15:00'], ['16:00','16:30']]

meetingTime: 45 min

1. Calculeaza timpul dintre primele 2 intervale (['9:00','11:30'] si ['12:00','15:00']) si obtine 30 minute. Timpul pentru un meeting este de 45 de minute deci acest interval nu este valid.
2. Calculeaza timpul dintre urmatoarele 2 intervale (['12:00','15:00'] si ['16:00','16:30']) si obtine 60 minute. Timpul pentru un meeting este de 45 de minute deci acest interval este valid asa ca va adauga la lista de rezultate intervalul ['15:00','16:00']