Big Data Programming 2: 2021 (25 Points)

Assignment 2: Python programming

Due date: 10.Mar.2021 23:55

How to submit:

The assignment must only be done in Python programming language using version 3.

assignments must be submitted by each individual in zip format with the name convention : matriculationNumber-fullName-assignment-1.zip

The zip file must contain :

1. the source code for Exercise 1 in python (.py) file format (if modules / packages are used then zip all python files)
2. the source code for Exercise 2 in python (.py) file format (if modules / packages are used then zip all python files)

Exercise 1: Calander(15 Points)

Imagine that you want to schedule a meeting of a certain duration with a co-worker. You have access to your calendar and your co-worker's calendar (both of which contain your respective meetings for the day, in the form of `[startTime,endTime]` , as well as both of your daily working hours (i.e., the earliest and latest times at which you're available for meetings every day, in the form of `[earliestStartTime,latestEndTime]` during which you could schedule the meeting.

Write a function that takes in your calendar, your daily working hours, your co-worker's calendar, your co-worker's working hours, and the duration of the meeting that you want to schedule, and that returns a list of all the time blocks (in the form of `[startTime,endTime]` during which you could schedule the meeting.

Note that times will be given and should be returned in 24 hour clock. For example: `[8:30,23:59]` and meeting durations will always be in minutes

Sample Input

```
YourCalendar = [['9:00', '10:30'], ['12:00', '13:00'], ['16:00', '18:00']]
YourWorkingHours = ['9:00', '20:00']
YourCoWorkersCalendar = [['10:00', '11:30'], ['12:30', '14:30'], ['14:30',
'15:00'], ['16:00', '17:00']]
YourCoWorkersWorkingHours = ['10:00', '18:30']
meetingDuration = 30
```

Sample Output:

```
[['11:30', '12:00'], ['15:00', '16:00'], ['18:00', '18:30']]
```

Exercise 2: Disks (10 Points)

You're given a non-empty array of arrays where each subarray holds three integers and represents a disk. These integers denote each disk's width, depth, and height, respectively. Your goal is to stack up the disks and to maximize the total height of the stack. A disk must have a strictly smaller width, depth, and height than any other disk below it.

Write a function that returns an array of the disks in the final stack, starting with the top disk and ending with the bottom disk. Note that you can't rotate disks; in other words, the integers in each subarray must represent `[width, depth, height]` at all times

Sample Input

```
[[2, 1, 2], [3, 2, 3], [2, 2, 8], [2, 3, 4], [1, 3, 1], [4, 4, 5]]
```

Sample Output (Read the disks from left to right)

```
[[4, 4, 5], [3, 2, 3], [2, 1, 2]]
```

Explaination : 10 (5 + 3 + 2) is the tallest height we can get by stacking disks

When more than combination equals the maximum height , all combinations must be present in the output.