Week-6 Graded Assignment (Programming)

Problem 1

Question

Write a function `freqWords` which accepts a list of words as a parameter and returns a dictionary which has the following structure:

- key: frequency of words in the list
- value: list of all words that have the above frequency

For example, consider this input:

```
['No', 'sentence', 'can', 'begin', 'with', 'because', 'because', 'because',
'is', 'a', 'conjunction.']
```

The output dictionary corresponding to this is:

Key	Value
`1`	`['no', 'sentence', 'can', 'begin', 'with', 'is', 'a', 'conjunction']`
`3`	`['because']`

```
def freqWords(words):
    '''
       Input: list of strings
       Output: dictionary
          key: integer
          value: list of strings
    '''
```

Note

- All keys of the returned dictionary should be in lowercase. This means that words like 'It' and 'it' in the input list are equivalent.
- Remove any occurrence of these characters from all the words present in the list `,;:.!`. These characters correspond to: comma, semi-colon, colon, full-stop, exclamation mark.
- You don't need to accept the input from the user or print the output to the console. This will be processed internally.
- You only need to fill the details in the body of the function.

Prefix (Shown)

```
malgudi = ['It', 'was', 'Monday', 'morning.', 'Swaminathan', 'was',
'reluctant', 'to', 'open', 'his',
 'eyes.', 'He', 'considered', 'Monday', 'specially', 'unpleasant', 'in',
'the', 'calendar.', 'After',
 'the', 'delicious', 'freedom', 'of', 'Saturday', 'And', 'Sunday,', 'it',
'was', 'difficult', 'to',
 'get', 'into', 'the', 'Monday', 'mood', 'of', 'work', 'and', 'discipline.',
'He', 'shuddered', 'at',
 'the', 'very', 'thought', 'of', 'school:', 'the', 'dismal', 'yellow',
'building;', 'the',
 'fire-eyed', 'Vedanayagam,', 'his', 'class', 'teacher,', 'and',
'headmaster', 'with', 'his',
 'thin', 'long', 'cane...']
```

Answer

```
def freqWords(wordList):
    freq = {} # dictionary to store word as key and frequenct as value
    for wd in wordList:
        for i in ',;:.!': # iterating over the string contains the
characters to be removed
            while i in wd: # do until no removeable left
                wd = wd.replace(i,'')
        if wd.lower() not in freq.keys(): # initialization of new words as
keys
            freq[wd.lower()] = 0
        freq[wd.lower()] += 1
    freqWords_ = {} # dictionary to store frequency as key and word as value
    for wd in freq.keys():
        if freq[wd] not in freqWords_.keys(): # initialization for new
frequency as key with empty list
            freqWords_[freq[wd]] = []
        freqWords_[freq[wd]].append(wd) # dictionary inversion
    return freqWords_
```

Suffix (Hidden)

```
dataset = {}
dataset['malgudi'] = malgudi
dataset['A Red, Red Rose'] = '''O my Luve's like a red, red rose
That’s newly sprung in June;
O my Luve's like the melodie
That’s sweetly play'd in tune.

As fair art thou, my bonnie lass,
So deep in luve am I:
And I will luve thee still, my dear,
Till a’ the seas gang dry:

Till a’ the seas gang dry, my dear,
And the rocks melt wi’ the sun:
I will luve thee still, my dear,
While the sands o’ life shall run.

And fare thee well, my only Luve
```

```
And fare thee well, a while!
And I will come again, my Luve,
Tho’ it were ten thousand mile.'''.strip().split()

dataset['Song of Myself'] = '''I celebrate myself, and sing myself,
  And what I assume you shall assume,
  For every atom belonging to me as good belongs to you.

  I loafe and invite my soul,
  I lean and loafe at my ease observing a spear of summer grass.

  My tongue, every atom of my blood, form'd from this soil, this air,
  Born here of parents born here from parents the same, and their
      parents the same,
  I, now thirty-seven years old in perfect health begin,
  Hoping to cease not till death.

  Creeds and schools in abeyance,
  Retiring back a while sufficed at what they are, but never forgotten,
  I harbor for good or bad, I permit to speak at every hazard,
  Nature without check with original energy.'''.strip().split()

fw = freqWords(dataset[input().strip()])
for i in sorted(list(fw.keys())):
    print(i)
    print(*sorted(fw[i]))
```

Test cases

Public

Input	Output

`malgudi`	`1`
	`after at building calendar cane class considered delicious difficult discipline dismal eyes fire-eyed freedom get headmaster in into long mood morning open reluctant saturday school shuddered specially sunday swaminathan teacher thin thought unpleasant vedanayagam very with work yellow`
	`2`
	`he it to`
	`3`
	`and his monday of was`
	`6`
	`the`

Private

Input	Output

`A Red, Red Rose`	`1`
	`again am art as bonnie come deep fair it june lass life melodie melt mile newly only o’ play'd rocks rose run sands shall so sprung sun sweetly ten thou thousand tho’ tune were wi’`
	`2`
	`a a’ dry fare gang like luve's o red seas still that’s till well while`
	`3`
	`dear in will`
	`4`
	`i thee`
	`5`
	`and luve`
	`6`
	`the`
	`8`
	`my`

`Song of Myself`	`1` `abeyance air are as back bad begin belonging belongs blood but cease celebrate check creeds death ease energy forgotten form'd grass harbor hazard health hoping invite lean me nature never not now observing old or original perfect permit retiring schools shall sing soil soul speak spear sufficed summer their they thirty-seven till tongue while with without years` `2` `a assume atom born for from good here in loafe myself same the this what you` `3` `at every of parents` `4` `my to` `6` `and` `7` `i`

Problem 2

Question

An institution decides to allow students to create student groups for each subject where students with similar marks can help each other. But it draws up a set of constraints for creating student groups:

- A group is associated with a particular `subject`.
- The group can consist of any number of students.
- The maximum difference of marks in this `subject` between the highest and lowest scorer within the group should be `markLimit`.

The **`Scores Dataset`** from the CT course will have the student details. Write a function called `crowdedGroup` that accepts three parameters as input:

- `scores`
- `subject`
- `markLimit`

The function should return a list of groups, where each group is a list of **`SeqNo`** of students of the largest possible group in that subject.

```
def crowdedGroup(scores, subject, markLimit):
    '''
        Input:
            scores: list of dictionaries; Scores Dataset
            subject: string
            markLimit: integer
        Output: list of lists
                inner list: list of integers (SeqNo)
                each inner list represents a group
    '''
```

Note

- `scores` is a list of dictionaries that represents the **`Scores Dataset`**. Look at the prefix code shown below to get an idea of the structure of `scores`.
- You don't need to accept the input from the user or print the output to the console. This will be processed internally.
- You only need to fill the details in the body of the function.

Prefix (Shown)

```
scores = [ { 'Chemistry': 78,
    'CityTown': 'Erode',
    'DateOfBirth': '7 Nov',
    'Gender': 'M',
    'Mathematics': 68,
    'Name': 'Bhuvanesh',
    'Physics': 64,
    'SeqNo': 0,
    'Total': 210},
```

```
  { 'Chemistry': 91,
    'CityTown': 'Salem',
    'DateOfBirth': '3 Jun',
    'Gender': 'M',
    'Mathematics': 62,
    'Name': 'Harish',
    'Physics': 45,
    'SeqNo': 1,
    'Total': 198},

  { 'Chemistry': 77,
    'CityTown': 'Chennai',
    'DateOfBirth': '4 Jan',
    'Gender': 'M',
    'Mathematics': 57,
    'Name': 'Shashank',
    'Physics': 54,
    'SeqNo': 2,
    'Total': 188},

  { 'Chemistry': 78,
    'CityTown': 'Chennai',
    'DateOfBirth': '5 May',
    'Gender': 'F',
    'Mathematics': 42,
    'Name': 'Rida',
    'Physics': 53,
    'SeqNo': 3,
    'Total': 173},

  { 'Chemistry': 89,
    'CityTown': 'Madurai',
    'DateOfBirth': '17 Nov',
    'Gender': 'F',
    'Mathematics': 87,
    'Name': 'Ritika',
    'Physics': 64,
    'SeqNo': 4,
    'Total': 240},

  { 'Chemistry': 84,
    'CityTown': 'Chennai',
    'DateOfBirth': '8 Feb',
    'Gender': 'F',
    'Mathematics': 71,
    'Name': 'Akshaya',
    'Physics': 92,
    'SeqNo': 5,
    'Total': 247},

  { 'Chemistry': 87,
    'CityTown': 'Ambur',
    'DateOfBirth': '23 Mar',
    'Gender': 'M',
    'Mathematics': 81,
    'Name': 'Sameer',
    'Physics': 82,
```

```
    'SeqNo': 6,
    'Total': 250},

  { 'Chemistry': 76,
    'CityTown': 'Vellore',
    'DateOfBirth': '15 Mar',
    'Gender': 'M',
    'Mathematics': 84,
    'Name': 'Aditya',
    'Physics': 92,
    'SeqNo': 7,
    'Total': 252},

  { 'Chemistry': 51,
    'CityTown': 'Bengaluru',
    'DateOfBirth': '28 Feb',
    'Gender': 'M',
    'Mathematics': 74,
    'Name': 'Surya',
    'Physics': 64,
    'SeqNo': 8,
    'Total': 189},

  { 'Chemistry': 73,
    'CityTown': 'Bengaluru',
    'DateOfBirth': '6 Dec',
    'Gender': 'M',
    'Mathematics': 63,
    'Name': 'Clarence',
    'Physics': 88,
    'SeqNo': 9,
    'Total': 224},

  { 'Chemistry': 68,
    'CityTown': 'Chennai',
    'DateOfBirth': '12 Jan',
    'Gender': 'F',
    'Mathematics': 64,
    'Name': 'Kavya',
    'Physics': 72,
    'SeqNo': 10,
    'Total': 204},

  { 'Chemistry': 92,
    'CityTown': 'Bengaluru',
    'DateOfBirth': '30 Apr',
    'Gender': 'M',
    'Mathematics': 97,
    'Name': 'Rahul',
    'Physics': 92,
    'SeqNo': 11,
    'Total': 281},

  { 'Chemistry': 71,
    'CityTown': 'Chennai',
    'DateOfBirth': '14 Jan',
    'Gender': 'F',
    'Mathematics': 52,
```

```
    'Name': 'Srinidhi',
    'Physics': 64,
    'SeqNo': 12,
    'Total': 187},

  { 'Chemistry': 89,
    'CityTown': 'Madurai',
    'DateOfBirth': '6 May',
    'Gender': 'M',
    'Mathematics': 65,
    'Name': 'Gopi',
    'Physics': 73,
    'SeqNo': 13,
    'Total': 227},

  { 'Chemistry': 93,
    'CityTown': 'Trichy',
    'DateOfBirth': '23 July',
    'Gender': 'F',
    'Mathematics': 89,
    'Name': 'Sophia',
    'Physics': 62,
    'SeqNo': 14,
    'Total': 244},

  { 'Chemistry': 90,
    'CityTown': 'Theni',
    'DateOfBirth': '22 Sep',
    'Gender': 'F',
    'Mathematics': 76,
    'Name': 'Goutami',
    'Physics': 58,
    'SeqNo': 15,
    'Total': 224},

  { 'Chemistry': 43,
    'CityTown': 'Trichy',
    'DateOfBirth': '30 Dec',
    'Gender': 'M',
    'Mathematics': 87,
    'Name': 'Tauseef',
    'Physics': 86,
    'SeqNo': 16,
    'Total': 216},

  { 'Chemistry': 67,
    'CityTown': 'Chennai',
    'DateOfBirth': '14 Dec',
    'Gender': 'M',
    'Mathematics': 62,
    'Name': 'Arshad',
    'Physics': 81,
    'SeqNo': 17,
    'Total': 210},

  { 'Chemistry': 97,
    'CityTown': 'Erode',
    'DateOfBirth': '9 Oct',
```

```
    'Gender': 'F',
    'Mathematics': 72,
    'Name': 'Abirami',
    'Physics': 92,
    'SeqNo': 18,
    'Total': 261},

  { 'Chemistry': 62,
    'CityTown': 'Trichy',
    'DateOfBirth': '30 Aug',
    'Gender': 'M',
    'Mathematics': 56,
    'Name': 'Vetrivel',
    'Physics': 78,
    'SeqNo': 19,
    'Total': 196},

  { 'Chemistry': 91,
    'CityTown': 'Vellore',
    'DateOfBirth': '17 Sep',
    'Gender': 'M',
    'Mathematics': 93,
    'Name': 'Kalyan',
    'Physics': 68,
    'SeqNo': 20,
    'Total': 252},

  { 'Chemistry': 74,
    'CityTown': 'Bengaluru',
    'DateOfBirth': '15 Mar',
    'Gender': 'F',
    'Mathematics': 78,
    'Name': 'Monika',
    'Physics': 69,
    'SeqNo': 21,
    'Total': 221},

  { 'Chemistry': 57,
    'CityTown': 'Nagercoil',
    'DateOfBirth': '17 Jul',
    'Gender': 'F',
    'Mathematics': 62,
    'Name': 'Priya',
    'Physics': 62,
    'SeqNo': 22,
    'Total': 181},

  { 'Chemistry': 88,
    'CityTown': 'Bengaluru',
    'DateOfBirth': '13 May',
    'Gender': 'F',
    'Mathematics': 97,
    'Name': 'Deepika',
    'Physics': 91,
    'SeqNo': 23,
    'Total': 276},

  { 'Chemistry': 58,
```

```
    'CityTown': 'Madurai',
    'DateOfBirth': '26 Dec',
    'Gender': 'M',
    'Mathematics': 44,
    'Name': 'Siddharth',
    'Physics': 72,
    'SeqNo': 24,
    'Total': 174},

  { 'Chemistry': 92,
    'CityTown': 'Chennai',
    'DateOfBirth': '16 May',
    'Gender': 'F',
    'Mathematics': 87,
    'Name': 'Geeta',
    'Physics': 75,
    'SeqNo': 25,
    'Total': 254},

  { 'Chemistry': 82,
    'CityTown': 'Chennai',
    'DateOfBirth': '22 Jul',
    'Gender': 'M',
    'Mathematics': 74,
    'Name': 'JK',
    'Physics': 71,
    'SeqNo': 26,
    'Total': 227},

  { 'Chemistry': 52,
    'CityTown': 'Madurai',
    'DateOfBirth': '4 Mar',
    'Gender': 'M',
    'Mathematics': 81,
    'Name': 'Jagan',
    'Physics': 76,
    'SeqNo': 27,
    'Total': 209},

  { 'Chemistry': 83,
    'CityTown': 'Madurai',
    'DateOfBirth': '10 Sep',
    'Gender': 'F',
    'Mathematics': 74,
    'Name': 'Nisha',
    'Physics': 83,
    'SeqNo': 28,
    'Total': 240},

  { 'Chemistry': 81,
    'CityTown': 'Vellore',
    'DateOfBirth': '13 Oct',
    'Gender': 'M',
    'Mathematics': 72,
    'Name': 'Naveen',
    'Physics': 66,
    'SeqNo': 29,
    'Total': 219}]
```

```
300
```

Answer

```
def crowdedGroup(scores, sub, markLimit):
    subDict = {} # dictionary to store subject mark as key and list of students' ID as value
    for sDict in scores:
        if sDict[sub] not in subDict.keys():
            subDict[sDict[sub]] = [] # Initialization for new keys
            subDict[sDict[sub]].append(sDict['SeqNo']) # Append the student ID inside the list in key of subject marks
        else:
            subDict[sDict[sub]].append(sDict['SeqNo'])
    maxNumber, crowdedGroup_ = 0, [] # variable to hold maximum group count and list of IDs of group
    for i in range(0, 101-markLimit): # Iterating from zero to maximum mark - limit
        group = []
        for m in subDict.keys(): # Iterating through each marks
            if i <= m <= i+markLimit:
                group.extend(subDict[m]) # Append all the IDs if it is in the valid limit
        if len(group) > maxNumber:
            maxNumber = len(group) # Updating the maximum group count so far
            crowedGroup_ = [group] # Updating the maximum group list so far
        if len(group) == maxNumber and group not in crowedGroup_:
            crowedGroup_.append(group) # adding another group having same count
    return crowdedGroup_
```

Suffix (Hidden)

```
sub, markLimit = input().strip().split()
markLimit = int(markLimit)
members = crowdedGroup(scores, sub, markLimit)
for m in members:
    print(*sorted(m))
```

Test cases

Public

Input	Output
`Physics 25`	`0 4 6 8 10 12 13 14 15 17 19 20 21 22 24 25 26 27 28 29` `0 4 6 8 10 12 13 14 16 17 19 20 21 22 24 25 26 27 28 29`

Private

Input	Output
`Chemistry 30`	`0 1 2 3 4 5 6 7 9 10 11 12 13 14 15 17 18 20 21 23 25 26 28 29`
`Mathematics 30`	`0 1 2 4 5 6 7 8 9 10 13 15 16 17 18 21 22 25 26 27 28 29` `0 1 4 5 6 7 8 9 10 13 14 15 16 17 18 21 22 25 26 27 28 29`
`Physics 32`	`0 4 5 6 7 8 9 10 11 12 13 14 16 17 18 19 20 21 22 23 24 25 26 27 28 29`

Tags

Problem 3

Question

The **`Scores Dataset`** from the CT course has the details of a class of students. A student `X` can mentor another student `Y` in **`subject`** if the following conditions are satisfied:

- `X` has scored more than `Y` in **`subject`**.
- The difference in marks between `X` and `Y` in **`subject`** lies in the range $[10, 20]$, both end points included.

Write a function `topMentors` that accepts the following parameters as input:

- `scores`
- `subject`

The function should return a dictionary having the following structure as output:

- key: **`SeqNo`** of the student who can mentor the largest number of students in `subject`
- value: list of **`SeqNo`** of students who can be mentored by the above student

```
def topMentors(scores, subject):
    '''
        Input:
            scores: list of dictionaries; Scores dataset
            subject: string
        Output:
            dictionary:
                key: integer (SeqNo)
                value: list of integers (SeqNo)
    '''
```

Note

- `scores` is a list of dictionaries that represents the **`Scores Dataset`**. Look at the prefix code shown below to get an idea of the structure of `scores`.
- You don't need to accept the input from the user or print the output to the console. This will be processed internally.
- You only need to fill the details in the body of the function.

Prefix (Shown)

```
scores = [ { 'Chemistry': 78,
    'CityTown': 'Erode',
    'DateOfBirth': '7 Nov',
    'Gender': 'M',
    'Mathematics': 68,
    'Name': 'Bhuvanesh',
    'Physics': 64,
    'SeqNo': 0,
    'Total': 210},

  { 'Chemistry': 91,
    'CityTown': 'Salem',
```

```
    'DateOfBirth': '3 Jun',
    'Gender': 'M',
    'Mathematics': 62,
    'Name': 'Harish',
    'Physics': 45,
    'SeqNo': 1,
    'Total': 198},

  { 'Chemistry': 77,
    'CityTown': 'Chennai',
    'DateOfBirth': '4 Jan',
    'Gender': 'M',
    'Mathematics': 57,
    'Name': 'Shashank',
    'Physics': 54,
    'SeqNo': 2,
    'Total': 188},

  { 'Chemistry': 78,
    'CityTown': 'Chennai',
    'DateOfBirth': '5 May',
    'Gender': 'F',
    'Mathematics': 42,
    'Name': 'Rida',
    'Physics': 53,
    'SeqNo': 3,
    'Total': 173},

  { 'Chemistry': 89,
    'CityTown': 'Madurai',
    'DateOfBirth': '17 Nov',
    'Gender': 'F',
    'Mathematics': 87,
    'Name': 'Ritika',
    'Physics': 64,
    'SeqNo': 4,
    'Total': 240},

  { 'Chemistry': 84,
    'CityTown': 'Chennai',
    'DateOfBirth': '8 Feb',
    'Gender': 'F',
    'Mathematics': 71,
    'Name': 'Akshaya',
    'Physics': 92,
    'SeqNo': 5,
    'Total': 247},

  { 'Chemistry': 87,
    'CityTown': 'Ambur',
    'DateOfBirth': '23 Mar',
    'Gender': 'M',
    'Mathematics': 81,
    'Name': 'Sameer',
    'Physics': 82,
    'SeqNo': 6,
    'Total': 250},

```

```
  { 'Chemistry': 76,
    'CityTown': 'Vellore',
    'DateOfBirth': '15 Mar',
    'Gender': 'M',
    'Mathematics': 84,
    'Name': 'Aditya',
    'Physics': 92,
    'SeqNo': 7,
    'Total': 252},

  { 'Chemistry': 51,
    'CityTown': 'Bengaluru',
    'DateOfBirth': '28 Feb',
    'Gender': 'M',
    'Mathematics': 74,
    'Name': 'Surya',
    'Physics': 64,
    'SeqNo': 8,
    'Total': 189},

  { 'Chemistry': 73,
    'CityTown': 'Bengaluru',
    'DateOfBirth': '6 Dec',
    'Gender': 'M',
    'Mathematics': 63,
    'Name': 'Clarence',
    'Physics': 88,
    'SeqNo': 9,
    'Total': 224},

  { 'Chemistry': 68,
    'CityTown': 'Chennai',
    'DateOfBirth': '12 Jan',
    'Gender': 'F',
    'Mathematics': 64,
    'Name': 'Kavya',
    'Physics': 72,
    'SeqNo': 10,
    'Total': 204},

  { 'Chemistry': 92,
    'CityTown': 'Bengaluru',
    'DateOfBirth': '30 Apr',
    'Gender': 'M',
    'Mathematics': 97,
    'Name': 'Rahul',
    'Physics': 92,
    'SeqNo': 11,
    'Total': 281},

  { 'Chemistry': 71,
    'CityTown': 'Chennai',
    'DateOfBirth': '14 Jan',
    'Gender': 'F',
    'Mathematics': 52,
    'Name': 'Srinidhi',
    'Physics': 64,
    'SeqNo': 12,
```

```
    'Total': 187},

  { 'Chemistry': 89,
    'CityTown': 'Madurai',
    'DateOfBirth': '6 May',
    'Gender': 'M',
    'Mathematics': 65,
    'Name': 'Gopi',
    'Physics': 73,
    'SeqNo': 13,
    'Total': 227},

  { 'Chemistry': 93,
    'CityTown': 'Trichy',
    'DateOfBirth': '23 July',
    'Gender': 'F',
    'Mathematics': 89,
    'Name': 'Sophia',
    'Physics': 62,
    'SeqNo': 14,
    'Total': 244},

  { 'Chemistry': 90,
    'CityTown': 'Theni',
    'DateOfBirth': '22 Sep',
    'Gender': 'F',
    'Mathematics': 76,
    'Name': 'Goutami',
    'Physics': 58,
    'SeqNo': 15,
    'Total': 224},

  { 'Chemistry': 43,
    'CityTown': 'Trichy',
    'DateOfBirth': '30 Dec',
    'Gender': 'M',
    'Mathematics': 87,
    'Name': 'Tauseef',
    'Physics': 86,
    'SeqNo': 16,
    'Total': 216},

  { 'Chemistry': 67,
    'CityTown': 'Chennai',
    'DateOfBirth': '14 Dec',
    'Gender': 'M',
    'Mathematics': 62,
    'Name': 'Arshad',
    'Physics': 81,
    'SeqNo': 17,
    'Total': 210},

  { 'Chemistry': 97,
    'CityTown': 'Erode',
    'DateOfBirth': '9 Oct',
    'Gender': 'F',
    'Mathematics': 72,
    'Name': 'Abirami',
```

```
    'Physics': 92,
    'SeqNo': 18,
    'Total': 261},

  { 'Chemistry': 62,
    'CityTown': 'Trichy',
    'DateOfBirth': '30 Aug',
    'Gender': 'M',
    'Mathematics': 56,
    'Name': 'Vetrivel',
    'Physics': 78,
    'SeqNo': 19,
    'Total': 196},

  { 'Chemistry': 91,
    'CityTown': 'Vellore',
    'DateOfBirth': '17 Sep',
    'Gender': 'M',
    'Mathematics': 93,
    'Name': 'Kalyan',
    'Physics': 68,
    'SeqNo': 20,
    'Total': 252},

  { 'Chemistry': 74,
    'CityTown': 'Bengaluru',
    'DateOfBirth': '15 Mar',
    'Gender': 'F',
    'Mathematics': 78,
    'Name': 'Monika',
    'Physics': 69,
    'SeqNo': 21,
    'Total': 221},

  { 'Chemistry': 57,
    'CityTown': 'Nagercoil',
    'DateOfBirth': '17 Jul',
    'Gender': 'F',
    'Mathematics': 62,
    'Name': 'Priya',
    'Physics': 62,
    'SeqNo': 22,
    'Total': 181},

  { 'Chemistry': 88,
    'CityTown': 'Bengaluru',
    'DateOfBirth': '13 May',
    'Gender': 'F',
    'Mathematics': 97,
    'Name': 'Deepika',
    'Physics': 91,
    'SeqNo': 23,
    'Total': 276},

  { 'Chemistry': 58,
    'CityTown': 'Madurai',
    'DateOfBirth': '26 Dec',
    'Gender': 'M',
```

```
    'Mathematics': 44,
    'Name': 'Siddharth',
    'Physics': 72,
    'SeqNo': 24,
    'Total': 174},

  { 'Chemistry': 92,
    'CityTown': 'Chennai',
    'DateOfBirth': '16 May',
    'Gender': 'F',
    'Mathematics': 87,
    'Name': 'Geeta',
    'Physics': 75,
    'SeqNo': 25,
    'Total': 254},

  { 'Chemistry': 82,
    'CityTown': 'Chennai',
    'DateOfBirth': '22 Jul',
    'Gender': 'M',
    'Mathematics': 74,
    'Name': 'JK',
    'Physics': 71,
    'SeqNo': 26,
    'Total': 227},

  { 'Chemistry': 52,
    'CityTown': 'Madurai',
    'DateOfBirth': '4 Mar',
    'Gender': 'M',
    'Mathematics': 81,
    'Name': 'Jagan',
    'Physics': 76,
    'SeqNo': 27,
    'Total': 209},

  { 'Chemistry': 83,
    'CityTown': 'Madurai',
    'DateOfBirth': '10 Sep',
    'Gender': 'F',
    'Mathematics': 74,
    'Name': 'Nisha',
    'Physics': 83,
    'SeqNo': 28,
    'Total': 240},

  { 'Chemistry': 81,
    'CityTown': 'Vellore',
    'DateOfBirth': '13 Oct',
    'Gender': 'M',
    'Mathematics': 72,
    'Name': 'Naveen',
    'Physics': 66,
    'SeqNo': 29,
    'Total': 219}]
```

Answer

```
def adjMat(scores, sub): # Function to find adjacency matrix
    adj = {} # adjacency matrix as nested dictionary
    for mentor in scores: # Outer dictionary / Row for mentor
        adj[mentor['SeqNo']] = {} # initializing with an empty dictionary for all SeqNo
        for mentored in scores: # Inner dictionary / column for mentored
            # mentoring condition and prevention of same student comparison
            if 10 <= mentor[sub] - mentored[sub] <= 20 and mentor['SeqNo'] != mentored['SeqNo']:
                adj[mentor['SeqNo']][mentored['SeqNo']] = 1
    return adj

def topMentors(scores, sub):
    adj = adjMat(scores, sub) # getting an adjacency matrix
    topMentors_ = {} # dictionary to store mentor as key and mentored SeqNo list as value
    maxMentored = 0 # value to hold the maximum mentored students
    for i in adj.keys():
        if len(adj[i]) > maxMentored:
            maxMentored = len(adj[i]) # replacing the maximum mentored count
            topMentors_ = {} # reinitialization with empty dictionary
            topMentors_[i] = list(adj[i].keys()) # adding entry with mentor as key and mentored list value
        if len(adj[i]) == maxMentored:
            topMentors_[i] = list(adj[i].keys()) # adding entry with mentor as key and mentored list value
    return topMentors_
```

Suffix (Hidden)

```
tm = topMentors(scores, input())
for i in sorted(list(tm.keys())):
    print(i)
    print(*sorted(tm[i]))
```

Test cases

Public

Input	Output
`Physics`	`6` `0 4 8 10 12 14 20 21 22 24 26 29`

Private

Input	Output
`Chemistry`	`14` `0 2 3 7 9 21 26 28 29`
`Mathematics`	`7` `0 5 8 10 13 18 26 28 29`