

Our Solution(s)

Run Code

Your Solutions

Run Code

Solution 1

```
1 # Copyright © 2020 AlgoExpert, LLC. All rights reserved.
2
3 # O(c1 + c2) time | O(c1 + c2) space - where c1 and c2 are the respective lengths of the two calendars
4 def calendarMatching(calendar1, dailyBounds1, calendar2, dailyBounds2):
5     updatedCalendar1 = updateCalendar(calendar1, dailyBounds1)
6     updatedCalendar2 = updateCalendar(calendar2, dailyBounds2)
7     mergedCalendar = mergeCalendars(updatedCalendar1, updatedCalendar2)
8     flattenedCalendar = flattenCalendar(mergedCalendar)
9     return getMatchingAvailabilities(flattenedCalendar, meetingDuration)
10
11
12 def updateCalendar(calendar, dailyBounds):
13     updatedCalendar = calendar[:]
14     updatedCalendar.insert(0, ["0:00", dailyBounds[0]])
15     updatedCalendar.append([dailyBounds[1], "23:59"])
16     return list(map(lambda m: [timeToMinutes(m[0]), timeToMinutes(m[1])], updatedCalendar))
17
18
19 def mergeCalendars(calendar1, calendar2):
20     merged = []
21     i, j = 0, 0
22     while i < len(calendar1) and j < len(calendar2):
23         meeting1, meeting2 = calendar1[i], calendar2[j]
24         if meeting1[0] < meeting2[0]:
25             merged.append(meeting1)
26             i += 1
27         else:
28             merged.append(meeting2)
29             j += 1
30     while i < len(calendar1):
31         merged.append(calendar1[i])
32         i += 1
33     while j < len(calendar2):
```

Solution 1

Solution 2

Solution 3

```
1 def calendarMatching(calendar1, dailyBounds1, calendar2, dailyBounds2, meetingDuration):
2     # Write your code here.
3     pass
4
```

Our Tests

```
1 def testCalendarMatching():
2     calendar1 = ["0:00", "1:00", "2:00", "3:00", "4:00", "5:00", "6:00", "7:00", "8:00", "9:00", "10:00", "11:00", "12:00", "13:00", "14:00", "15:00", "16:00", "17:00", "18:00", "19:00", "20:00", "21:00", "22:00", "23:00"]
3     dailyBounds1 = [0, 1440]
4     calendar2 = ["0:00", "1:00", "2:00", "3:00", "4:00", "5:00", "6:00", "7:00", "8:00", "9:00", "10:00", "11:00", "12:00", "13:00", "14:00", "15:00", "16:00", "17:00", "18:00", "19:00", "20:00", "21:00", "22:00", "23:00"]
5     dailyBounds2 = [0, 1440]
6     meetingDuration = 30
7     result = calendarMatching(calendar1, dailyBounds1, calendar2, dailyBounds2, meetingDuration)
8     assert result == 12
```

Custom Output

Submit Code

```

10         result.append([1, 1, 1, 1, 1])
11     result.append([1, 1, 1, 1, 1])
12     return result
13
14 def main():
15     result = generateResult(5)
16     print(result)
17
18 if __name__ == '__main__':
19     main()
20
21 result = generateResult(5)
22 print(result)
23
24 result = generateResult(5)
25 print(result)
26
27 result = generateResult(5)
28 print(result)
29
30 result = generateResult(5)
31 print(result)
32
33 result = generateResult(5)
34 print(result)
35
36 result = generateResult(5)
37 print(result)
38
39 result = generateResult(5)
40 print(result)
41
42 result = generateResult(5)
43 print(result)
44
45 result = generateResult(5)
46 print(result)
47
48 result = generateResult(5)
49 print(result)
50
51 result = generateResult(5)
52 print(result)
53
54 result = generateResult(5)
55 print(result)
56
57 result = generateResult(5)
58 print(result)
59
60 result = generateResult(5)
61 print(result)
62
63 result = generateResult(5)
64 print(result)
65
66 result = generateResult(5)
67 print(result)
68
69 result = generateResult(5)
70 print(result)
71
72 result = generateResult(5)
73 print(result)
74
75 result = generateResult(5)
76 print(result)
77
78 result = generateResult(5)
79 print(result)
80
81 result = generateResult(5)
82 print(result)
83
84 result = generateResult(5)
85 print(result)
86
87 result = generateResult(5)
88 print(result)
89
90 result = generateResult(5)
91 print(result)
92
93 result = generateResult(5)
94 print(result)
95
96 result = generateResult(5)
97 print(result)
98
99 result = generateResult(5)
100 print(result)

```

Run or submit code when you're ready.