Solution 1

```
// Copyright © 2020 AlgoExpert, LLC. All rights reserved.

using System;
using System.Collections.Generic;

public class Program {
  // O(c1 + c2) time | O(c1 + c2) space - where c1 and c2 are the respective numbers of meetings in calendar1 and calendar2
  public static List<StringMeeting> CalendarMatching(
    List<StringMeeting> calendar1,
    StringMeeting dailyBounds1,
    List<StringMeeting> calendar2,
    StringMeeting dailyBounds2,
    int meetingDuration
    ) {
    List<Meeting> updatedCalendar1 = updateCalendar(calendar1, dailyBounds1);
    List<Meeting> updatedCalendar2 = updateCalendar(calendar2, dailyBounds2);
    List<Meeting> mergedCalendar = mergeCalendars(updatedCalendar1, updatedCalendar2);
    List<Meeting> flattenedCalendar = flattenCalendar(mergedCalendar);
    return getMatchingAvailabilities(flattenedCalendar, meetingDuration);
  }

  public static List<Meeting> updateCalendar(
    List<StringMeeting> calendar,
    StringMeeting dailyBounds
    ) {
    List<StringMeeting> updatedCalendar = new List<StringMeeting>();
    updatedCalendar.Add(new StringMeeting("0:00", dailyBounds.start));
    updatedCalendar.AddRange(calendar);
    updatedCalendar.Add(new StringMeeting(dailyBounds.end, "23:59"));
    List<Meeting> calendarInMinutes = new List<Meeting>();
    for (int i = 0; i < updatedCalendar.Count; i++) {
      calendarInMinutes.Add(new Meeting(
        timeToMinutes(updatedCalendar[i].start),
        timeToMinutes(updatedCalendar[i].end)
        ));

    }
    return calendarInMinutes;
  }

  public static List<Meeting> mergeCalendars(
    List<Meeting> calendar1,
    List<Meeting> calendar2
    ) {
    List<Meeting> merged = new List<Meeting>();
    int i = 0;
    int j = 0;
    while (i < calendar1.Count && j < calendar2.Count) {
      Meeting meeting1 = calendar1[i];
      Meeting meeting2 = calendar2[j];
      if (meeting1.start < meeting2.start) {
        merged.Add(meeting1);
        i++;
      } else {
        merged.Add(meeting2);
        j++;
      }
    }
    while (i < calendar1.Count) merged.Add(calendar1[i++]);
    while (j < calendar2.Count) merged.Add(calendar2[j++]);
    return merged;
  }

  public static List<Meeting> flattenCalendar(List<Meeting> calendar) {
    List<Meeting> flattened = new List<Meeting>();
    flattened.Add(calendar[0]);
    for (int i = 1; i < calendar.Count; i++) {
      Meeting currentMeeting = calendar[i];
      Meeting previousMeeting = flattened[flattened.Count - 1];
      if (previousMeeting.end >= currentMeeting.start) {
        Meeting newPreviousMeeting = new Meeting(
          previousMeeting.start,
          Math.Max(previousMeeting.end, currentMeeting.end)
          );
        flattened[flattened.Count - 1] = newPreviousMeeting;
      } else {
        flattened.Add(currentMeeting);
      }
    }
    return flattened;
  }

  public static List<StringMeeting> getMatchingAvailabilities(
    List<Meeting> calendar,
    int meetingDuration
    ) {
    List<Meeting> matchingAvailabilities = new List<Meeting>();
    for (int i = 1; i < calendar.Count; i++) {
      int start = calendar[i - 1].end;
      int end = calendar[i].start;
      int availabilityDuration = end - start;
      if (availabilityDuration >= meetingDuration) {
        matchingAvailabilities.Add(new Meeting(start, end));
      }
    }
    List<StringMeeting> matchingAvailabilitiesInHours = new List<StringMeeting>();
    for (int i = 0; i < matchingAvailabilities.Count; i++) {
      matchingAvailabilitiesInHours.Add(new StringMeeting(
          minutesToTime(
            matchingAvailabilities[i].
            start),
          minutesToTime(
            matchingAvailabilities[i].
            end)
          ));
    }
    return matchingAvailabilitiesInHours;
  }

  public static int timeToMinutes(string time) {
    int delimiterPos = time.IndexOf(":");
    int hours = Int32.Parse(time.Substring(0, delimiterPos));
    int minutes = Int32.Parse(time.Substring(delimiterPos + 1));
    return hours * 60 + minutes;
  }
```

```
116
117     public static string minutesToTime(int minutes) {
```