You have been hired to test a new calendar app! Congratulations!(?).

This program allows users to book meetings, adding those meetings to calendars maintained for rooms and employees. It will actively prevent multiple bookings, and will manage the busy and open status for employees and rooms. The system enables the following high-level functions:

- Booking a meeting
- Booking vacation time
- Checking availability for a room
- Checking availability for a person
- Printing the agenda for a room
- Printing the agenda for a person.

Normally, actions are conducted through a command line user interface provided by the main method in the ***PlannerInterface*** class. As a tester, you - of course - have full access to the source code to employ in testing the system. The code is available at:
https://drive.google.com/drive/folders/1Dh14HFzdxZ8sMbW1RHWvSmP4Hnhovbx-?usp=sharing
This is an Intellij Idea project. But you can use any IDE you want. Just keep the java files and set up project configuration for your own environment.

**You are to do the following:**

1. **Form groups of at most five members**
    a. Team up with your classmates and distribute the work evenly.

2. **Formulate an informal test plan**
    a. Given the above features and the code documentation, plan out a series of unit tests to ensure that these features can be performed without error.
        i. Make sure you think about both the normal execution and illegal Inputs and actions that could be performed. Think of as many things that could go wrong as you can! For instance, you will probably be able to add a normal meeting, but can you add a meeting for February 35th? Try it out.

3. **Write tests in the jUnit framework**
    a. If a test is supposed to cause an exception to be thrown. Make sure you check for that exception.
    b. Make sure that your expected output is detailed enough to ensure that - if something is supposed to fail - that it fails for the correct reasons. Use appropriate assertions.

4. **IDE Configuration:**
    If you need to configure JUnit in your IDE of choice follow this guide -
    https://junit.org/junit5/docs/current/user-guide/#running-tests

**<u>Some Useful Hints(For Now):</u>**

- Write all your unit tests inside the test package. The classes corresponding to each unit have already been created.
- Check out the ***CalendarTest.java*** class to find a sample unit test already written. The test method is named `testAddMeeting_holiday()` As the name suggests this unit test checks whether you can book a meeting on a holiday. Which should be occupied in the calendar.
- To write better tests you can analyse the source code for faults and potential vulnerabilities. Some of which are listed below.

**<u>Can you expose these faults :</u>**

- ***getMeeting()*** and ***removeMeeting()*** perform no error checking on dates.

```
public Meeting getMeeting(int month, int day, int index){
   return occupied.get(month).get(day).get(index);
}
public void removeMeeting(int month, int day, int index){
   occupied.get(month).get(day).remove(index);
}
```

- Used a >= in checking for illegal times. December no longer exists.

```
if(mMonth < 1 || mMonth >= 12){
   throw new TimeConflictException("Month does not exist.");
}
```

- Find more faults like these on your own and write test cases to expose these faults.
- Remember there will be several unit tests corresponding to each method in each class.