***DSA PBL ALGORITHUM***

**STEP NO 01:**

* Make a class with the name “crash” in which all the data of the csv file will be stored.
* It contains a constructor with all the necessary parameters to initialize them and getter methods will be made to get the respective values.
* Make an other class with the name “main\_Queue” in which all the required methods will be implemented.
* Make a main method in this class and import csv file in the IDE which you are using like intellij. Make object of the main class.
* Now import the header file io.\* and use bufferreader (in the try catch block) to read the csv file by providing the path of csv file.
* Now for reading the file line by line in while loop use readline(); function till its not null and store it in the String array(by splitting it with ‘,’), by using the crash class object providing string array indexes in the constructor and adding it in main queue.

**STEP NO 02:**

* + Now make three sub queue for weather conditions (clear, rainy, snow) by using for each loop traverse the main queue and check
  + for (crash obj: Main\_Queue){  
     if (obj.getWeather\_condition().equalsIgnoreCase("clear"))
  + if it satisfy the condition then add it in the sub queue by using add(); method
* clearWeather.add(obj);
* do the same thing with remaining 2 queues

**STEP NO 03:**

* + now make 7 sub queues for days of the week (Monday, Tuesday, Wednesday, Thursday, Friday , Saturday, Sunday ) by using for each loop traverse the main queue and check
  + for (crash obj: Main\_Queue){  
     if (obj.getCrash\_dayOfWeek().equalsIgnoreCase("monday"))
  + do the same thing we did for the above queues.

**STEP NO 04:**

* + Make BST class (which will contain methods like add(); grow(); toString(); rightmost(); leftmost(); etc) and two binary Search trees one will store hit and run crashes by using add method and other will store not hit and run crashes by using add method(for each month).
  + Firstly make two methods for counting hitAndRun and notHitAndRun crashes in each month, use 12 counters to count for each month, as given below:
  + if (c.getCrash\_month().equalsIgnoreCase("january")&&c.getCrash\_hitAndRun().equalsIgnoreCase("hit-and-run")) cJan++;
  + and then add each counter in the Binary tree.

**PROBLEM NO 01:**

* + Deadliest day of the week
  + Compare the size of all the 7 sub queues with each other and print the day which is deadliest of all. Just like
* if (Monday().size()>Tuesday().size() &&(Monday().size()>Wednesday().size()) &&(Monday().size()>Thursday().size())&&(Monday().size()>Friday().size()) &&(Monday().size()>Saturday().size()) &&(Monday().size()>Sunday().size())){  
   System.*out*.println("Monday deadliest: "+Monday().size());
  + repeat it for all the days.
  + Which month has the deadliest Friday
  + Make 12 counters, and compare the month’s name with Friday with help of getter function, if its true counter++; just like
* if (c.getCrash\_month().equalsIgnoreCase("january")&&c.getCrash\_dayOfWeek().equalsIgnoreCase("friday")) cJan++;
  + then at last compare all counters with each other, the one month’s counter which have maximum value is the deadliest Friday container.

**PROBLEM NO 02:**

* + Make 12 counters, and compare the month’s name with weather condition(rain) with help of getter function, if its true counter++; just like
  + if (c.getCrash\_month().equalsIgnoreCase("january")&&c.getWeather\_condition().equalsIgnoreCase("rain")) cJan++;
  + then at last print the counters with month’s name.

**PROBLEM NO 03:**

**for finding highest and lowest Hit and Run**

* + Compare leftMost() method to all 12 counter of hit and run.
  + Use if else to check which month counter is equal to the left most node key that will give us the required month with lowest value.
  + Do this step with rightmost() to find highest value.

**For finding highest and lowest Not Hit and Run**

* + Compare leftMost() method to all 12 counter of not hit and run.
  + Use if else to check which month counter is equal to the left most node key that will give us the required month with lowest value.
  + Similarly doing with rightmost() to find highest value.

**POSTULATE:**

* Make a Boolean method with name “light\_condition”, in which traverse the main queue using for each loop.
* Declare 2 counters.
* And check in which days crashes were max :
* if ((c.getCrash\_dayOfWeek().equalsIgnoreCase("saturday")||c.getCrash\_dayOfWeek().equalsIgnoreCase ("sunday"))&&c.getLight\_condition().equalsIgnoreCase("dark lighted")) weekEnds++;
* do same for Mondays and Tuesdays
* compare both counters if weekends counters is greater than weekdays then its true otherwise false

***CODE OF THE PBL***

import java.io.\*;  
import java.util.LinkedList;  
import java.util.Queue;  
import java.lang.\*;  
  
//Step 1: Make a queue for the car crashes based on their time line from January to December (Day 1 to  
//Day 31) and store day of week, crash type, light conditions, weather conditions and other necessary data.  
class mainQueue {  
  
 Queue <crash> Main\_Queue=new LinkedList<>();  
 //hit-and-run counters  
 int cJan=0, cFeb=0, cMarch=0, cApril=0, cMay=0, cJune=0, cJuly=0, cAugust=0, cSep=0, cOct=0, cNov=0, cDec=0;  
 //not hit-and-run counters  
 int c1Jan=0, c2Feb=0, c3March=0, c4April=0, c5May=0, c6June=0, c7July=0, c8August=0, c9Sep=0, c10Oct=0, c11Nov=0,  
 c12Dec=0;  
 //dark lighted condition counter  
 int weekDays=0, weekEnds=0;  
  
 public static class crash {  
 int crash\_day, crash\_year;  
 String crash\_month, crash\_timeOfDay, crash\_dayOfWeek, crash\_intersection, crash\_hitAndRun, light\_condition, weather\_condition;  
//constructor  
 public crash(int crash\_year,String crash\_month, int crash\_day, String crash\_timeOfDay, String crash\_dayOfWeek, String crash\_intersection,  
 String crash\_hitAndRun, String light\_condition, String weather\_condition ){  
 this.crash\_year=crash\_year;  
 this.crash\_month=crash\_month;  
 this.crash\_day=crash\_day;  
 this.crash\_timeOfDay=crash\_timeOfDay;  
 this.crash\_dayOfWeek=crash\_dayOfWeek;  
 this.crash\_intersection=crash\_intersection;  
 this.crash\_hitAndRun=crash\_hitAndRun;  
 this.light\_condition=light\_condition;  
 this.weather\_condition=weather\_condition;  
 }  
   
 //getter methods   
 public String getWeather\_condition(){  
 return weather\_condition;  
 }  
 public String getCrash\_dayOfWeek(){  
 return crash\_dayOfWeek;  
 }  
 public String getCrash\_month(){  
 return crash\_month;  
 }  
 public String getCrash\_hitAndRun(){  
 return crash\_hitAndRun;  
 }  
 public String getLight\_condition(){return light\_condition;}  
  
 }  
  
 // Step 2: Make three sub queues from the original queue such as sub queues storing only clear weather  
 //crashes, snow weather crashes and rain weather crashes.  
 public Queue CWcrash(){  
 Queue<crash> clearWeather=new LinkedList<>();  
 for (crash obj: Main\_Queue){  
 if (obj.getWeather\_condition().equalsIgnoreCase("clear")){  
 clearWeather.add(obj);  
 }}  
  
 return clearWeather;  
 }  
  
 public Queue SWcrash(){  
 Queue<crash> snowWeather=new LinkedList<>();  
 for (crash obj: Main\_Queue){  
 if (obj.getWeather\_condition().equalsIgnoreCase("snow")){  
 snowWeather.add(obj);  
 }  
 }  
 return snowWeather;  
 }  
 public Queue RWcrash(){  
 Queue<crash> rainyWeather=new LinkedList<>();  
 for (crash obj: Main\_Queue){  
 if (obj.getWeather\_condition().equalsIgnoreCase("rain")){  
 rainyWeather.add(obj);  
 }  
 }  
 return rainyWeather;  
 }  
  
 //Step 3: Make seven sub queues (one for each day of the week) for all crashes.  
  
 public Queue Monday(){  
 Queue<crash> monday=new LinkedList<>();  
 for (crash obj: Main\_Queue){  
 if (obj.getCrash\_dayOfWeek().equalsIgnoreCase("monday")){  
 monday.add(obj);  
 }  
 }return monday;  
 }  
  
 public Queue Tuesday(){  
 Queue<crash> tuesday=new LinkedList<>();  
 for (crash obj: Main\_Queue){  
 if (obj.getCrash\_dayOfWeek().equalsIgnoreCase("tuesday")){  
 tuesday.add(obj);  
 }  
 }return tuesday;  
 }  
  
 public Queue Wednesday(){  
 Queue<crash> wednesday=new LinkedList<>();  
 for (crash obj: Main\_Queue){  
 if (obj.getCrash\_dayOfWeek().equalsIgnoreCase("wednesday")){  
 wednesday.add(obj);  
 }  
 }return wednesday;  
 }  
  
 public Queue Thursday(){  
 Queue<crash> thursday=new LinkedList<>();  
 for (crash obj: Main\_Queue){  
 if (obj.getCrash\_dayOfWeek().equalsIgnoreCase("thursday")){  
 thursday.add(obj);  
 }  
 }return thursday;  
 }  
  
 public Queue Friday(){  
 Queue<crash> friday=new LinkedList<>();  
 for (crash obj: Main\_Queue){  
 if (obj.getCrash\_dayOfWeek().equalsIgnoreCase("friday")){  
 friday.add(obj);  
 }  
 }return friday;  
 }  
  
 public Queue Saturday(){  
 Queue<crash> saturday=new LinkedList<>();  
 for (crash obj: Main\_Queue){  
 if (obj.getCrash\_dayOfWeek().equalsIgnoreCase("saturday")){  
 saturday.add(obj);  
 }  
 }return saturday;  
 }  
  
 public Queue Sunday(){  
 Queue<crash> sunday=new LinkedList<>();  
 for (crash obj: Main\_Queue){  
 if (obj.getCrash\_dayOfWeek().equalsIgnoreCase("sunday")){  
 sunday.add(obj);  
 }  
 }return sunday;  
 }  
  
// =========problem 1==========  
   
//A: How to find out which day (from Monday to Sunday) was the deadliest of all days as well as  
//which months’ Friday was the deadliest among all Fridays.  
  
 public void deadliestDay(){  
 if (Monday().size()>Tuesday().size() &&(Monday().size()>Wednesday().size()) &&(Monday().size()>Thursday().size())  
 &&(Monday().size()>Friday().size()) &&(Monday().size()>Saturday().size()) &&(Monday().size()>Sunday().size())){  
 System.*out*.println("Monday deadliest: "+Monday().size());  
 }  
 if (Tuesday().size()>Monday().size()&&(Tuesday().size()>Wednesday().size())&&(Tuesday().size()>Thursday().size())  
 &&(Tuesday().size()>Friday().size()) &&(Tuesday().size()>Saturday().size())&&(Tuesday().size()>Sunday().size())){  
 System.*out*.println("Tuesday deadliest: "+Tuesday().size());  
 }  
  
 if (Wednesday().size()>Monday().size()&&(Wednesday().size()>Tuesday().size())&&(Wednesday().size()>Thursday().size())  
 &&(Wednesday().size()>Friday().size()) &&(Wednesday().size()>Saturday().size())&&(Wednesday().size()>Sunday().size())){  
 System.*out*.println("Wednesday deadliest: "+Wednesday().size());  
 }  
  
 if (Thursday().size()>Monday().size()&&(Thursday().size()>Tuesday().size())&&(Thursday().size()>Wednesday().size())  
 &&(Thursday().size()>Friday().size()) &&(Thursday().size()>Saturday().size())&&(Thursday().size()>Sunday().size())){  
 System.*out*.println("Thursday deadliest: "+Thursday().size());  
 }  
  
 if (Friday().size()>Monday().size()&&(Friday().size()>Tuesday().size())&&(Friday().size()>Wednesday().size())  
 &&(Friday().size()>Thursday().size()) &&(Friday().size()>Saturday().size())&&(Friday().size()>Sunday().size())){  
 System.*out*.println("Friday deadliest: "+Friday().size());  
 }  
  
 if (Saturday().size()>Monday().size()&&(Saturday().size()>Tuesday().size())&&(Saturday().size()>Wednesday().size())  
 &&(Saturday().size()>Thursday().size()) &&(Saturday().size()>Friday().size())&&(Saturday().size()>Sunday().size())){  
 System.*out*.println("Saturday deadliest: "+Saturday().size());  
 }  
  
 if (Sunday().size()>Monday().size()&&(Sunday().size()>Tuesday().size())&&(Sunday().size()>Wednesday().size())  
 &&(Sunday().size()>Thursday().size()) &&(Sunday().size()>Friday().size())&&(Sunday().size()>Saturday().size())){  
 System.*out*.println("Sunday deadliest: "+Sunday().size());  
 }  
 }  
  
 // which months’ Friday was the deadliest among all Fridays.  
 public void deadliestFridayOfMonth(){  
 int cJan=0, cFeb=0, cMarch=0, cApril=0, cMay=0, cJune=0, cJuly=0, cAugust=0, cSep=0, cOct=0, cNov=0, cDec=0;  
 for (crash c:Main\_Queue){  
  
 if (c.getCrash\_month().equalsIgnoreCase("january")&&c.getCrash\_dayOfWeek().equalsIgnoreCase("friday")) cJan++;  
 if (c.getCrash\_month().equalsIgnoreCase("february")&&c.getCrash\_dayOfWeek().equalsIgnoreCase("friday"))cFeb++;  
 if (c.getCrash\_month().equalsIgnoreCase("march")&&c.getCrash\_dayOfWeek().equalsIgnoreCase("friday"))cMarch++;  
 if (c.getCrash\_month().equalsIgnoreCase("april")&&c.getCrash\_dayOfWeek().equalsIgnoreCase("friday"))cApril++;  
 if (c.getCrash\_month().equalsIgnoreCase("may")&&c.getCrash\_dayOfWeek().equalsIgnoreCase("friday"))cMay++;  
 if (c.getCrash\_month().equalsIgnoreCase("june")&&c.getCrash\_dayOfWeek().equalsIgnoreCase("friday"))cJune++;  
 if (c.getCrash\_month().equalsIgnoreCase("july")&&c.getCrash\_dayOfWeek().equalsIgnoreCase("friday"))cJuly++;  
 if (c.getCrash\_month().equalsIgnoreCase("august")&&c.getCrash\_dayOfWeek().equalsIgnoreCase("friday"))cAugust++;  
 if (c.getCrash\_month().equalsIgnoreCase("september")&&c.getCrash\_dayOfWeek().equalsIgnoreCase("friday"))cSep++;  
 if (c.getCrash\_month().equalsIgnoreCase("october")&&c.getCrash\_dayOfWeek().equalsIgnoreCase("friday"))cOct++;  
 if (c.getCrash\_month().equalsIgnoreCase("november")&&c.getCrash\_dayOfWeek().equalsIgnoreCase("friday"))cNov++;  
 if (c.getCrash\_month().equalsIgnoreCase("december")&&c.getCrash\_dayOfWeek().equalsIgnoreCase("friday"))cDec++;}  
  
 if ((cJan>cFeb)&&(cJan>cMarch)&&(cJan>cApril)&&(cJan>cMay)&&(cJan>cJune)&&(cJan>cJuly)&&(cJan>cAugust)&&(cJan>cSep)&&(cJan>cOct)&&(cJan>cNov)  
 &&(cJan>cDec)){System.*out*.println("january's friday is deadliest ");}  
 if (cFeb>cJan&&cFeb>cMarch&&cFeb>cApril&&cFeb>cMay&&cFeb>cJune&&cFeb>cJuly&&cFeb>cAugust&&cFeb>cSep&&cFeb>cOct&&cFeb>cNov  
 &&cFeb>cDec){System.*out*.println("february's friday is deadliest ");}  
 if (cMarch>cJan&&cMarch>cFeb&&cMarch>cApril&&cMarch>cMay&&cMarch>cJune&&cMarch>cJuly&&cMarch>cAugust&&cMarch>cSep&&cMarch>cOct&&cMarch>cNov  
 &&cJan>cDec){System.*out*.println("march's friday is deadliest ");}  
 if (cApril>cJan&&cApril>cFeb&&cApril>cMarch&&cApril>cMay&&cApril>cJune&&cApril>cJuly&&cApril>cAugust&&cApril>cSep&&cApril>cOct&&cApril>cNov  
 &&cJan>cDec){System.*out*.println("april's friday is deadliest ");}  
 if (cMay>cJan&&cMay>cFeb&&cMay>cMarch&&cMay>cApril&&cMay>cJune&&cMay>cJuly&&cMay>cAugust&&cMay>cSep&&cMay>cOct&&cMay>cNov  
 &&cMay>cDec){System.*out*.println("may's friday is deadliest ");}  
 if (cJune>cJan&&cJune>cFeb&&cJune>cMarch&&cJune>cApril&&cJune>cMay&&cJune>cJuly&&cJune>cAugust&&cJune>cSep&&cJune>cOct&&cJune>cNov  
 &&cJune>cDec){System.*out*.println("june's friday is deadliest ");}  
 if (cJuly>cJan&&cJuly>cFeb&&cJuly>cMarch&&cJuly>cApril&&cJuly>cMay&&cJuly>cJune&&cJuly>cAugust&&cJuly>cSep&&cJuly>cOct&&cJuly>cNov  
 &&cJuly>cDec){System.*out*.println("july's friday is deadliest ");}  
 if (cAugust>cJan&&cAugust>cFeb&&cAugust>cMarch&&cAugust>cApril&&cAugust>cMay&&cAugust>cJune&&cAugust>cJuly&&cAugust>cSep&&cAugust>cOct  
 &&cAugust>cNov&&cAugust>cDec){System.*out*.println("august's friday is deadliest ");}  
 if (cSep>cJan&&cSep>cFeb&&cSep>cMarch&&cSep>cApril&&cSep>cMay&&cSep>cJune&&cSep>cJuly&&cSep>cAugust&&cSep>cOct&&cSep>cNov  
 &&cSep>cDec){System.*out*.println("september's friday is deadliest ");}  
 if (cOct>cJan&&cOct>cFeb&&cOct>cMarch&&cOct>cApril&&cOct>cMay&&cOct>cJune&&cOct>cJuly&&cOct>cAugust&&cOct>cSep&&cOct>cNov  
 &&cOct>cDec){System.*out*.println("october's friday is deadliest ");}  
 if (cNov>cJan&&cNov>cFeb&&cNov>cMarch&&cNov>cApril&&cNov>cMay&&cNov>cJune&&cNov>cJuly&&cNov>cAugust&&cNov>cSep&&cNov>cOct  
 &&cNov>cDec){System.*out*.println("november's friday is deadliest ");}  
 if (cDec>cJan&&cDec>cFeb&&cDec>cMarch&&cDec>cApril&&cDec>cMay&&cDec>cJune&&cDec>cJuly&&cDec>cAugust&&cDec>cSep&&cDec>cOct  
 &&cDec>cNov){System.*out*.println("december's friday is deadliest ");}  
  
 }  
  
 // ========Problem 2=========  
   
 // How to find the number of rainy day crashes in every month.  
public void rainyDayCrashes(){  
 int cJan=0, cFeb=0, cMarch=0, cApril=0, cMay=0, cJune=0, cJuly=0, cAugust=0, cSep=0, cOct=0, cNov=0, cDec=0;  
 for (crash c:Main\_Queue){  
  
 if (c.getCrash\_month().equalsIgnoreCase("january")&&c.getWeather\_condition().equalsIgnoreCase("rain")) cJan++;  
 if (c.getCrash\_month().equalsIgnoreCase("february")&&c.getWeather\_condition().equalsIgnoreCase("rain"))cFeb++;  
 if (c.getCrash\_month().equalsIgnoreCase("march")&&c.getWeather\_condition().equalsIgnoreCase("rain"))cMarch++;  
 if (c.getCrash\_month().equalsIgnoreCase("april")&&c.getWeather\_condition().equalsIgnoreCase("rain"))cApril++;  
 if (c.getCrash\_month().equalsIgnoreCase("may")&&c.getWeather\_condition().equalsIgnoreCase("rain"))cMay++;  
 if (c.getCrash\_month().equalsIgnoreCase("june")&&c.getWeather\_condition().equalsIgnoreCase("rain"))cJune++;  
 if (c.getCrash\_month().equalsIgnoreCase("july")&&c.getWeather\_condition().equalsIgnoreCase("rain"))cJuly++;  
 if (c.getCrash\_month().equalsIgnoreCase("august")&&c.getWeather\_condition().equalsIgnoreCase("rain"))cAugust++;  
 if (c.getCrash\_month().equalsIgnoreCase("september")&&c.getWeather\_condition().equalsIgnoreCase("rain"))cSep++;  
 if (c.getCrash\_month().equalsIgnoreCase("october")&&c.getWeather\_condition().equalsIgnoreCase("rain"))cOct++;  
 if (c.getCrash\_month().equalsIgnoreCase("november")&&c.getWeather\_condition().equalsIgnoreCase("rain"))cNov++;  
 if (c.getCrash\_month().equalsIgnoreCase("december")&&c.getWeather\_condition().equalsIgnoreCase("rain"))cDec++;}  
  
  
 {System.*out*.println("no of rainy day crashes in january: "+cJan);}  
  
 {System.*out*.println("no of rainy day crashes in february: "+cFeb);}  
  
 {System.*out*.println("no of rainy day crashes in march:"+cMarch);}  
  
 {System.*out*.println("no of rainy day crashes in april: "+cApril);}  
  
 {System.*out*.println("no of rainy day crashes in may: "+cMay);}  
  
 {System.*out*.println("no of rainy day crashes in june: "+cJune);}  
  
 {System.*out*.println("no of rainy day crashes in july: "+cJuly);}  
  
 {System.*out*.println("no of rainy day crashes in august: "+cAugust);}  
  
 {System.*out*.println("no of rainy day crashes in september: "+cSep);}  
  
 {System.*out*.println("no of rainy day crashes in october: "+cOct);}  
  
 {System.*out*.println("no of rainy day crashes in november: "+cNov);}  
  
 {System.*out*.println("no of rainy day crashes in december: "+cDec);}  
  
}  
  
//calculation of hit-and-run crashes in each month  
public void hitAndRun(){  
 for (crash c:Main\_Queue){  
  
 if (c.getCrash\_month().equalsIgnoreCase("january")&&c.getCrash\_hitAndRun().equalsIgnoreCase("hit-and-run")) cJan++;  
 if (c.getCrash\_month().equalsIgnoreCase("february")&&c.getCrash\_hitAndRun().equalsIgnoreCase("hit-and-run"))cFeb++;  
 if (c.getCrash\_month().equalsIgnoreCase("march")&&c.getCrash\_hitAndRun().equalsIgnoreCase("hit-and-run"))cMarch++;  
 if (c.getCrash\_month().equalsIgnoreCase("april")&&c.getCrash\_hitAndRun().equalsIgnoreCase("hit-and-run"))cApril++;  
 if (c.getCrash\_month().equalsIgnoreCase("may")&&c.getCrash\_hitAndRun().equalsIgnoreCase("hit-and-run"))cMay++;  
 if (c.getCrash\_month().equalsIgnoreCase("june")&&c.getCrash\_hitAndRun().equalsIgnoreCase("hit-and-run"))cJune++;  
 if (c.getCrash\_month().equalsIgnoreCase("july")&&c.getCrash\_hitAndRun().equalsIgnoreCase("hit-and-run"))cJuly++;  
 if (c.getCrash\_month().equalsIgnoreCase("august")&&c.getCrash\_hitAndRun().equalsIgnoreCase("hit-and-run"))cAugust++;  
 if (c.getCrash\_month().equalsIgnoreCase("september")&&c.getCrash\_hitAndRun().equalsIgnoreCase("hit-and-run"))cSep++;  
 if (c.getCrash\_month().equalsIgnoreCase("october")&&c.getCrash\_hitAndRun().equalsIgnoreCase("hit-and-run"))cOct++;  
 if (c.getCrash\_month().equalsIgnoreCase("november")&&c.getCrash\_hitAndRun().equalsIgnoreCase("hit-and-run"))cNov++;  
 if (c.getCrash\_month().equalsIgnoreCase("december")&&c.getCrash\_hitAndRun().equalsIgnoreCase("hit-and-run"))cDec++;}  
 {System.*out*.println("no of hit-and-run crashes in january: "+cJan);}  
  
 {System.*out*.println("no of hit-and-run day crashes in february: "+cFeb);}  
  
 {System.*out*.println("no of hit-and-run day crashes in march:"+cMarch);}  
  
 {System.*out*.println("no of hit-and-run day crashes in april: "+cApril);}  
  
 {System.*out*.println("no of hit-and-run day crashes in may: "+cMay);}  
  
 {System.*out*.println("no of hit-and-run day crashes in june: "+cJune);}  
  
 {System.*out*.println("no of hit-and-run day crashes in july: "+cJuly);}  
  
 {System.*out*.println("no of hit-and-run day crashes in august: "+cAugust);}  
  
 {System.*out*.println("no of hit-and-run day crashes in september: "+cSep);}  
  
 {System.*out*.println("no of hit-and-run day crashes in october: "+cOct);}  
  
 {System.*out*.println("no of hit-and-run day crashes in november: "+cNov);}  
  
 {System.*out*.println("no of hit-and-run day crashes in december: "+cDec);}  
  
 }  
  
 //calculation of not hit-and-run crashes in each month  
 public void notHitAndRun(){  
 for (crash c:Main\_Queue){  
  
 if (c.getCrash\_month().equalsIgnoreCase("january")&&c.getCrash\_hitAndRun().equalsIgnoreCase("not hit-and-run")) c1Jan++;  
 if (c.getCrash\_month().equalsIgnoreCase("february")&&c.getCrash\_hitAndRun().equalsIgnoreCase("not hit-and-run"))c2Feb++;  
 if (c.getCrash\_month().equalsIgnoreCase("march")&&c.getCrash\_hitAndRun().equalsIgnoreCase("not hit-and-run"))c3March++;  
 if (c.getCrash\_month().equalsIgnoreCase("april")&&c.getCrash\_hitAndRun().equalsIgnoreCase("not hit-and-run"))c4April++;  
 if (c.getCrash\_month().equalsIgnoreCase("may")&&c.getCrash\_hitAndRun().equalsIgnoreCase("not hit-and-run"))c5May++;  
 if (c.getCrash\_month().equalsIgnoreCase("june")&&c.getCrash\_hitAndRun().equalsIgnoreCase("not hit-and-run"))c6June++;  
 if (c.getCrash\_month().equalsIgnoreCase("july")&&c.getCrash\_hitAndRun().equalsIgnoreCase("not hit-and-run"))c7July++;  
 if (c.getCrash\_month().equalsIgnoreCase("august")&&c.getCrash\_hitAndRun().equalsIgnoreCase("not hit-and-run"))c8August++;  
 if (c.getCrash\_month().equalsIgnoreCase("september")&&c.getCrash\_hitAndRun().equalsIgnoreCase("not hit-and-run"))c9Sep++;  
 if (c.getCrash\_month().equalsIgnoreCase("october")&&c.getCrash\_hitAndRun().equalsIgnoreCase("not hit-and-run"))c10Oct++;  
 if (c.getCrash\_month().equalsIgnoreCase("november")&&c.getCrash\_hitAndRun().equalsIgnoreCase("not hit-and-run"))c11Nov++;  
 if (c.getCrash\_month().equalsIgnoreCase("december")&&c.getCrash\_hitAndRun().equalsIgnoreCase("not hit-and-run"))c12Dec++;}  
 {System.*out*.println("no of not hit-and-run crashes in january: "+cJan);}  
  
 {System.*out*.println("no of not hit-and-run day crashes in february: "+cFeb);}  
  
 {System.*out*.println("no of not hit-and-run day crashes in march:"+cMarch);}  
  
 {System.*out*.println("no of not hit-and-run day crashes in april: "+cApril);}  
  
 {System.*out*.println("no of not hit-and-run day crashes in may: "+cMay);}  
  
 {System.*out*.println("no of not hit-and-run day crashes in june: "+cJune);}  
  
 {System.*out*.println("no of not hit-and-run day crashes in july: "+cJuly);}  
  
 {System.*out*.println("no of not hit-and-run day crashes in august: "+cAugust);}  
  
 {System.*out*.println("no of not hit-and-run day crashes in september: "+cSep);}  
  
 {System.*out*.println("no of not hit-and-run day crashes in october: "+cOct);}  
  
 {System.*out*.println("no of not hit-and-run day crashes in november: "+cNov);}  
  
 {System.*out*.println("no of not hit-and-run day crashes in december: "+cDec);}  
  
 }  
  
 // ========Postulate 1=========  
   
 // Is it true that the weekends (Saturdays and Sundays) when the light conditions were dark,  
 //caused more crashes than Mondays and Tuesdays.  
public boolean darkLightCondition(){  
 for (crash c: Main\_Queue){  
 if ((c.getCrash\_dayOfWeek().equalsIgnoreCase("saturday")||c.getCrash\_dayOfWeek().equalsIgnoreCase  
 ("sunday"))&&c.getLight\_condition().equalsIgnoreCase("dark lighted")) weekEnds++;  
  
 if ((c.getCrash\_dayOfWeek().equalsIgnoreCase("monday")||c.getCrash\_dayOfWeek().equalsIgnoreCase  
 ("tuesday"))&&c.getLight\_condition().equalsIgnoreCase("dark lighted")) weekDays++;  
 }  
 System.*out*.println("week days: "+weekDays);  
 System.*out*.println("week ends: "+weekEnds);  
 if (weekEnds>weekDays){  
 return true; //System.out.println("saturdays and sundays caused more crashes than mondays and tuesdays");  
 }  
 else return false;//System.out.println("mondays and tuesdays caused more crashes than saturdays and sunday");  
}  
  
  
 //main method  
  
 public static void main(String[] args) {  
 mainQueue q=new mainQueue();  
  
 String file="C:\\Users\\hp\\IdeaProjects\\Crashes\\src\\ped\_crash.csv";  
// BufferedReader reader=null;  
 String line="";  
 try(BufferedReader reader =new BufferedReader(new FileReader(file))){  
// File f=new File(file);  
// reader =new BufferedReader(new FileReader(f));  
 reader.readLine();  
 while((line= reader.readLine())!=null){  
 String[] values=line.split(",");  
 crash details=new crash(Integer.*parseInt*(values[0]),values[1], Integer.*parseInt*(values[2]), values[3],  
 values[4], values[5], values[6],values[7], values[8] );  
 q.Main\_Queue.add(details);  
  
 }  
  
 }catch(FileNotFoundException e){}  
 catch (IOException e) {  
 throw new RuntimeException(e);  
 }  
 System.*out*.println("size of the csv file: "+q.Main\_Queue.size());  
 System.*out*.println("size of monday in the main queue: "+q.Monday().size());  
 System.*out*.println("size of tuesday in the main queue: "+q.Tuesday().size());  
 System.*out*.println("size of wednesday in the main queue: "+q.Wednesday().size());  
 System.*out*.println("size of thursday in the main queue: "+q.Thursday().size());  
 System.*out*.println("size of friday in the main queue: "+q.Friday().size());  
 System.*out*.println("size of saturday in the main queue: "+q.Saturday().size());  
 System.*out*.println("size of sunday in the main queue: "+q.Sunday().size());  
 System.*out*.println();  
  
 System.*out*.println(" ==========PROBLEM NO 01(a)==========");  
 System.*out*.println(" which day of the week is the deadliest?");  
 System.*out*.print(" deadliest day of the week is: ");  
 q.deadliestDay();  
 System.*out*.println();  
  
 System.*out*.println(" ==========PROBLEM NO 01(b)==========");  
 System.*out*.println(" which month's friday is the deadliest?");  
 System.*out*.print(" deadliest friday is of the month : ");  
 q.deadliestFridayOfMonth();  
 System.*out*.println();  
  
 System.*out*.println(" ============PROBLEM NO 02===========");  
 System.*out*.println(" how to find the rainy day crashes in every month?");  
 System.*out*.println(" rainy day the crashes in each month are : ");  
 q.rainyDayCrashes();  
 System.*out*.println();  
  
 System.*out*.println(" ==========PROBLEM NO 03==========");  
 System.*out*.println();  
 System.*out*.println(" ------no of hit and run crashes every month:--------- ");  
 q.hitAndRun();  
  
 System.*out*.println();  
 System.*out*.println(" ------no of not hit and run crashes every month:-------- ");  
 q.notHitAndRun();  
  
  
 BST bst1=new BST(q.cJan);  
 bst1.add(q.cFeb);  
 bst1.add(q.cMarch);  
 bst1.add(q.cApril);  
 bst1.add(q.cMay);  
 bst1.add(q.cJune);  
 bst1.add(q.cJuly);  
 bst1.add(q.cAugust);  
 bst1.add(q.cSep);  
 bst1.add(q.cOct);  
 bst1.add(q.cNov);  
 bst1.add(q.cDec);  
  
 BST bst2=new BST(q.c1Jan);  
 bst2.add(q.c2Feb);  
 bst2.add(q.c3March);  
 bst2.add(q.c4April);  
 bst2.add(q.c5May);  
 bst2.add(q.c6June);  
 bst2.add(q.c7July);  
 bst2.add(q.c8August);  
 bst2.add(q.c9Sep);  
 bst2.add(q.c10Oct);  
 bst2.add(q.c11Nov);  
 bst2.add(q.c12Dec);  
  
 System.*out*.println();  
 System.*out*.println(" ------data stored in the binary tree: for hit and run: ------");  
 System.*out*.println("bst 1: "+bst1.toString());  
 System.*out*.println();  
 System.*out*.println(" -------data stored in the binary tree: for not hit and run:------- ");  
 System.*out*.println("bst 2: "+bst2.toString());  
  
 System.*out*.println();  
 System.*out*.print("rightmost leaf of the hit and run tree is:");  
 System.*out*.println(bst1.rightMost(bst1));  
 System.*out*.print("leftmost leaf of the hit and run tree is:");  
 System.*out*.println(bst1.leftMost(bst1));  
  
 System.*out*.println();  
 System.*out*.print("rightmost leaf of the not hit and run tree is:");  
 System.*out*.println(bst2.rightMost(bst2));  
 System.*out*.print("leftmost leaf of the not hit and run tree is:");  
 System.*out*.println(bst2.leftMost(bst2));  
  
  
  
 System.*out*.println();  
 System.*out*.print(" highest hit and run crashes are happened in: ");  
 int right1=bst1.rightMost(bst1);  
 if (right1==q.cJan) System.*out*.println("january");  
 if (right1==q.cFeb) System.*out*.println("february");  
 if (right1==q.cMarch) System.*out*.println("march");  
 if (right1==q.cApril) System.*out*.println("april");  
 if (right1==q.cMay) System.*out*.println("may");  
 if (right1==q.cJune) System.*out*.println("june");  
 if (right1==q.cJuly) System.*out*.println("july");  
 if (right1==q.cAugust) System.*out*.println("august");  
 if (right1==q.cSep) System.*out*.println("september");  
 if (right1==q.cOct) System.*out*.println("october");  
 if (right1==q.cNov) System.*out*.println("november");  
 if (right1==q.cDec) System.*out*.println("december");  
  
  
 System.*out*.print(" lowest hit and run crashes are happened in: ");  
 int left1=bst1.leftMost(bst1);  
 if (left1==q.cJan) System.*out*.println("january");  
 if (left1==q.cFeb) System.*out*.println("february");  
 if (left1==q.cMarch) System.*out*.println("march");  
 if (left1==q.cApril) System.*out*.println("april");  
 if (left1==q.cMay) System.*out*.println("may");  
 if (left1==q.cJune) System.*out*.println("june");  
 if (left1==q.cJuly) System.*out*.println("july");  
 if (left1==q.cAugust) System.*out*.println("august");  
 if (left1==q.cSep) System.*out*.println("september");  
 if (left1==q.cOct) System.*out*.println("october");  
 if (left1==q.cNov) System.*out*.println("november");  
 if (left1==q.cDec) System.*out*.println("december");  
  
  
 System.*out*.print(" highest not hit and run crashes are happened in: ");  
 int right2=bst2.rightMost(bst2);  
 if (right2==q.c1Jan) System.*out*.println("january");  
 if (right2==q.c2Feb) System.*out*.println("february");  
 if (right2==q.c3March) System.*out*.println("march");  
 if (right2==q.c4April) System.*out*.println("april");  
 if (right2==q.c5May) System.*out*.println("may");  
 if (right2==q.c6June) System.*out*.println("june");  
 if (right2==q.c7July) System.*out*.println("july");  
 if (right2==q.c8August) System.*out*.println("august");  
 if (right2==q.c9Sep) System.*out*.println("september");  
 if (right2==q.c10Oct) System.*out*.println("october");  
 if (right2==q.c11Nov) System.*out*.println("november");  
 if (right2==q.c12Dec) System.*out*.println("december");  
  
  
 System.*out*.print(" lowest not hit and run crashes are happened in: ");  
 int left2=bst2.leftMost(bst2);  
 if (left2==q.c1Jan) System.*out*.println("january");  
 if (left2==q.c2Feb) System.*out*.println("february");  
 if (left2==q.c3March) System.*out*.println("march");  
 if (left2==q.c4April) System.*out*.println("april");  
 if (left2==q.c5May) System.*out*.println("may");  
 if (left2==q.c6June) System.*out*.println("june");  
 if (left2==q.c7July) System.*out*.println("july");  
 if (left2==q.c8August) System.*out*.println("august");  
 if (left2==q.c9Sep) System.*out*.println("september");  
 if (left2==q.c10Oct) System.*out*.println("october");  
 if (left2==q.c11Nov) System.*out*.println("november");  
 if (left2==q.c12Dec) System.*out*.println("december");  
  
 System.*out*.println();  
 System.*out*.println(" ============ postulate no 1============ ");  
 System.*out*.println(" Is it true that the weekends (Saturdays and Sundays) when the light conditions were dark, \n" +  
 "caused more crashes than Mondays and Tuesdays. ");  
 System.*out*.println(q.darkLightCondition());  
  
  
  
  
 }}

**BST Class**

//Step 4: Make two binary trees to store total number of crashes for each month(use crash number as key)  
//for both ‘hit and run’ and ‘not hit and run’ crash types.  
public class BST {  
 private int key, height;  
 private BST left, right;  
 BST(){  
 this.left=this;  
 this.right=this;  
 height=-1;  
 }  
// private BST(int key, BST left, BST right){  
// this.key=key;  
// this.left=left;  
// this.right=right;  
// height=1+Math.max(left.height, right.height);  
// }  
 public static final BST *nil*=new BST();  
 public BST(int key){  
 this.key=key;  
 left=right=*nil*;  
 }  
 public boolean add(int key){  
 int oldsize=size();  
 grow(key);  
 return size()>oldsize;  
 }  
 public int size(){  
 if (this==*nil*){return 0;}  
 else{ return 1+ left.size()+ right.size();}  
 }  
 public BST grow(int key){  
 if (this==*nil*){return new BST(key);}  
 if (key==this.key) {return this;}  
 if (key<this.key){ left=left.grow(key);}  
 else {right=right.grow(key);}  
 height=1+Math.*max*(left.height, right.height);  
 return this;}  
  
 public String toString(){  
 if (this==*nil*) return " ";  
 return left.toString()+" "+key+" "+right.toString();  
 }  
  
 public int rightMost(BST b){  
 if(b.right==*nil*) {return b.key;}  
 return b.rightMost(b.right);  
  
  
 }  
  
 public int leftMost(BST b1){  
 if(b1.left==*nil*) {return b1.key;}  
 return b1.leftMost(b1.left);  
 }  
  
}

**Output**

"C:\Program Files\Java\jdk-17.0.4\bin\java.exe" "-javaagent:C:\Program Files\JetBrains\IntelliJ IDEA Community Edition 2022.2\lib\idea\_rt.jar=63690:C:\Program Files\JetBrains\IntelliJ IDEA Community Edition 2022.2\bin" -Dfile.encoding=UTF-8 -classpath C:\Users\hp\IdeaProjects\Crashes\out\production\Crashes mainQueue  
 size of the csv file: 6809  
 size of monday in the main queue: 959  
 size of tuesday in the main queue: 1021  
 size of wednesday in the main queue: 977  
 size of thursday in the main queue: 1028  
 size of friday in the main queue: 1078  
 size of saturday in the main queue: 990  
 size of sunday in the main queue: 756  
  
 ==========PROBLEM NO 01(a)==========  
 which day of the week is the deadliest?  
 deadliest day of the week is: Friday deadliest: 1078  
  
 ==========PROBLEM NO 01(b)==========  
 which month's friday is the deadliest?  
 deadliest friday is of the month : october's friday is deadliest   
  
 ============PROBLEM NO 02===========  
 how to find the rainy day crashes in every month?  
 rainy day the crashes in each month are :  
 no of rainy day crashes in january: 23  
 no of rainy day crashes in february: 15  
 no of rainy day crashes in march:27  
 no of rainy day crashes in april: 20  
 no of rainy day crashes in may: 17  
 no of rainy day crashes in june: 8  
 no of rainy day crashes in july: 5  
 no of rainy day crashes in august: 12  
 no of rainy day crashes in september: 22  
 no of rainy day crashes in october: 48  
 no of rainy day crashes in november: 36  
 no of rainy day crashes in december: 20  
  
 ==========PROBLEM NO 03==========  
  
 ------no of hit and run crashes every month:---------  
 no of hit-and-run crashes in january: 213  
 no of hit-and-run day crashes in february: 171  
 no of hit-and-run day crashes in march:214  
 no of hit-and-run day crashes in april: 170  
 no of hit-and-run day crashes in may: 195  
 no of hit-and-run day crashes in june: 220  
 no of hit-and-run day crashes in july: 208  
 no of hit-and-run day crashes in august: 224  
 no of hit-and-run day crashes in september: 229  
 no of hit-and-run day crashes in october: 222  
 no of hit-and-run day crashes in november: 238  
 no of hit-and-run day crashes in december: 205  
  
 ------no of not hit and run crashes every month:--------  
 no of not hit-and-run crashes in january: 213  
 no of not hit-and-run day crashes in february: 171  
 no of not hit-and-run day crashes in march:214  
 no of not hit-and-run day crashes in april: 170  
 no of not hit-and-run day crashes in may: 195  
 no of not hit-and-run day crashes in june: 220  
 no of not hit-and-run day crashes in july: 208  
 no of not hit-and-run day crashes in august: 224  
 no of not hit-and-run day crashes in september: 229  
 no of not hit-and-run day crashes in october: 222  
 no of not hit-and-run day crashes in november: 238  
 no of not hit-and-run day crashes in december: 205  
  
 ------data stored in the binary tree: for hit and run: ------  
 bst 1: 170 171 195 205 208 213 214 220 222 224 229 238  
  
 -------data stored in the binary tree: for not hit and run:-------  
 bst 2: 276 288 311 323 328 346 366 385 387 399 418 473  
  
 rightmost leaf of the hit and run tree is:238  
 leftmost leaf of the hit and run tree is:170  
  
 rightmost leaf of the not hit and run tree is:473  
 leftmost leaf of the not hit and run tree is:276  
  
 highest hit and run crashes are happened in: november  
 lowest hit and run crashes are happened in: april  
 highest not hit and run crashes are happened in: october  
 lowest not hit and run crashes are happened in: april  
  
 ============ postulate no 1============  
 Is it true that the weekends (Saturdays and Sundays) when the light conditions were dark,  
 caused more crashes than Mondays and Tuesdays.  
 week days: 542  
 week ends: 690  
 true  
  
 Process finished with exit code 0