**PBL Code**

**Roll No :21sw062**

**Section: II**

import org.w3c.dom.Node;  
  
import java.io.\*;  
import java.util.LinkedList;  
import java.util.Queue;  
  
class LinkedQueue {  
  
 String fileLocation="C:\\Users\\hp\\IdeaProjects\\Crashes\_Details\\ped\_crashes.csv";  
 String line="";  
 Queue<Crashes> queue=new LinkedList<>();  
// Queue<String> monday=new LinkedList<>();  
// Queue<String> tuesday=new LinkedList<>();  
// Queue<String> wednesday=new LinkedList<>();  
// Queue<String> thursday=new LinkedList<>();  
// Queue<String> friday=new LinkedList<>();  
// Queue<String> saturday=new LinkedList<>();  
// Queue<String> sunday=new LinkedList<>();  
 String[] data;Crashes newData;  
 public void formMainQueue() throws IOException {  
 try (BufferedReader br = new BufferedReader(new FileReader(fileLocation))) {  
 br.readLine();  
 while ((line = br.readLine()) != null) {  
 data = line.split(",");  
 newData= new Crashes(Integer.parseInt(data[0]),data[1],Integer.parseInt(data[2]),data[3],data[4],data[5],data[6],  
 data[7],data[8]);  
 queue.add(newData);  
 }  
 } catch (IOException e) {  
 e.printStackTrace();  
 }  
 } //form main queue  
  
  
 //-->Form 03 Queues of Weather rainy,Snowy and Clear  
 public Queue SnowQueue(){  
 Queue<Crashes> snow=new LinkedList<>();  
 for (Crashes crashes : queue) {  
 if (crashes.getWeatherCondition().equalsIgnoreCase("Snow")){  
 snow.add(crashes);  
 }}  
 return snow;  
 }  
  
 public Queue rainQueue(){  
 Queue<Crashes> rain=new LinkedList<>();  
  
 for (Crashes crashes : queue) {  
 if (crashes.getWeatherCondition().equalsIgnoreCase("Rain")){  
 rain.add(crashes);  
 }  
 }  
 return rain;  
  
 }  
  
 public Queue clearQueue(){  
 Queue<Crashes> clear=new LinkedList<>();  
  
 for (Crashes crashes : queue) {  
 if (crashes.getWeatherCondition().equalsIgnoreCase("Clear")){  
 clear.add(crashes);  
 }  
 // weekDays.add(crashes.getDayOfWeek());  
  
 }  
 return clear;}  
  
  
 //--> form Queue of the Days  
 public Queue thursdayQueue(){  
 Queue<Crashes> thursday=new LinkedList<>();  
  
 for (Crashes crashes : queue) {  
 if (crashes.getDayOfWeek().equalsIgnoreCase("thursday")){  
 thursday.add(crashes);  
 }  
 }  
 return thursday;  
 }  
 public Queue fridayQueue(){  
 Queue<Crashes> friday=new LinkedList<>();  
  
 for (Crashes crashes : queue) {  
 if (crashes.getDayOfWeek().equalsIgnoreCase("friday")) {  
 friday.add(crashes);  
 }}  
// System.out.println("Friday Crashes: "+fridayQueue().size()+"\n");  
 return friday;}  
 public Queue saturdayQueue(){  
 Queue<Crashes> saturday=new LinkedList<>();  
  
 for (Crashes crashes : queue) {  
 if (crashes.getDayOfWeek().equalsIgnoreCase("saturday")){  
 saturday.add(crashes);}  
 }  
// System.out.println("Saturday Crashes: "+saturdayQueue().size()+"\n");  
 return saturday;  
 }  
 public Queue sundayQueue(){  
 Queue<Crashes> sunday=new LinkedList<>();  
  
 for (Crashes crashes : queue) {  
 if (crashes.getDayOfWeek().equalsIgnoreCase("sunday")){  
 sunday.add(crashes);  
 }  
 }  
// System.out.println("Sunday Crashes: "+sundayQueue().size()+"\n");  
 return sunday;  
 }  
 public Queue mondayQueue(){  
 Queue<Crashes> monday=new LinkedList<>();  
  
 for (Crashes crashes : queue) {  
 if (crashes.getDayOfWeek().equalsIgnoreCase("monday")){  
 monday.add(crashes);  
 }  
 }  
// System.out.println("Monday Crashes"+monday.size()+"\n");  
 return monday;}  
 public Queue tuesdayQueue(){  
 Queue<Crashes> tuesday=new LinkedList<>();  
  
 for (Crashes crashes : queue) {  
 if (crashes.getDayOfWeek().equalsIgnoreCase("tuesday")){  
 tuesday.add(crashes);  
 }}  
// System.out.println("Tuesday Crashes: "+tuesdayQueue().size()+"\n");  
 return tuesday;  
 }  
 public Queue wedQueue(){  
 Queue<Crashes> wednesday=new LinkedList<>();  
  
 for (Crashes crashes : queue) {  
 if (crashes.getDayOfWeek().equalsIgnoreCase("Wednesday")){  
 wednesday.add(crashes);  
 }  
 }  
// System.out.println("Wednesday Crashes: "+wedQueue().size()+"\n");  
 return wednesday;  
 }  
  
 public void deadLeastOfAllDay(){  
 Queue thursday=thursdayQueue();  
 Queue friday=fridayQueue();  
 Queue saturday=saturdayQueue();  
 Queue sunday=sundayQueue();  
 Queue monday=mondayQueue();  
 Queue tuesday=tuesdayQueue();  
 Queue wednesday=wedQueue();  
  
// int max=Math.max(thursdayQueue().size(),fridayQueue().size(),saturdayQueue().size(),sundayQueue().size(),mondayQueue().size(),  
// tuesdayQueue().size(),wedQueue().size());  
 if(thursday.size()>friday.size() && thursday.size()>saturday.size() && thursday.size()>sunday.size()  
 &&thursday.size()>monday.size()&&thursday.size()>tuesday.size()&&thursday.size()>wednesday.size()) {  
 System.out.println("DeadLeast day is Thursday");  
 }  
 else if(friday.size()>thursday.size() && friday.size()>saturday.size() && friday.size()>sunday.size()  
 &&friday.size()>monday.size()&&friday.size()>tuesday.size()&&friday.size()>wednesday.size()) {  
 System.out.println("DeadLeast day is Friday");  
 }  
 else if(saturday.size()>thursday.size() && saturday.size()>friday.size() && saturday.size()>sunday.size()  
 &&saturday.size()>monday.size()&&saturday.size()>tuesday.size()&&saturday.size()>wednesday.size()) {  
 System.out.println("DeadLeast day is Saturday");  
 }  
 else if(sunday.size()>thursday.size() && sunday.size()>saturday.size() && sunday.size()>friday.size()  
 &&sunday.size()>monday.size()&&sunday.size()>tuesday.size()&&sunday.size()>wednesday.size()) {  
 System.out.println("DeadLeast day is Sunday");  
 }  
 else if(monday.size()>thursday.size() && monday.size()>saturday.size() && monday.size()>sunday.size()  
 &&monday.size()>friday.size()&&monday.size()>tuesday.size()&&monday.size()>wednesday.size()) {  
 System.out.println("DeadLeast day is Monday");  
 }  
 else if(tuesday.size()>thursday.size() && tuesday.size()>saturday.size() && tuesday.size()>sunday.size()  
 &&tuesday.size()>monday.size()&&tuesday.size()>friday.size()&&tuesday.size()>wednesday.size()) {  
 System.out.println("DeadLeast day is Tuesday");  
 }  
 else {  
 System.out.println("DeadLeast day is Wednesday");  
 }  
  
  
 }  
  
  
  
 // Find Dead least Month of the friday  
 public void deadLiestMonth(){  
 int jan=0,feb=0,march=0,april=0,may=0,june=0,july=0,august=0,sep=0,oct=0,nov=0,dec=0;  
 for (Crashes c:queue){  
 if (c.getDayOfWeek().equalsIgnoreCase("friday") && c.getCrashMonth().equalsIgnoreCase("January")){  
 jan++;  
 }  
 if (c.getDayOfWeek().equalsIgnoreCase("friday") && c.getCrashMonth().equalsIgnoreCase("February")){  
 feb++;  
 }  
 if (c.getDayOfWeek().equalsIgnoreCase("friday") && c.getCrashMonth().equalsIgnoreCase("March")){  
 march++;  
 }  
 if (c.getDayOfWeek().equalsIgnoreCase("friday") && c.getCrashMonth().equalsIgnoreCase("April")){  
 april++;  
 }  
 if (c.getDayOfWeek().equalsIgnoreCase("friday") && c.getCrashMonth().equalsIgnoreCase("may")){  
 may++;  
 }  
 if (c.getDayOfWeek().equalsIgnoreCase("friday") && c.getCrashMonth().equalsIgnoreCase("june")){  
 june++;  
 }  
 if (c.getDayOfWeek().equalsIgnoreCase("friday") && c.getCrashMonth().equalsIgnoreCase("july")){  
 july++;  
 }  
 if (c.getDayOfWeek().equalsIgnoreCase("friday") && c.getCrashMonth().equalsIgnoreCase("august")){  
 august++;  
 }  
 if (c.getDayOfWeek().equalsIgnoreCase("friday") && c.getCrashMonth().equalsIgnoreCase("September")){  
 sep++;  
 }  
 if (c.getDayOfWeek().equalsIgnoreCase("friday") && c.getCrashMonth().equalsIgnoreCase("October")){  
 oct++;  
 }  
 if (c.getDayOfWeek().equalsIgnoreCase("friday") && c.getCrashMonth().equalsIgnoreCase("November")){  
 nov++;  
 }  
 if (c.getDayOfWeek().equalsIgnoreCase("friday") && c.getCrashMonth().equalsIgnoreCase("December")){  
 dec++;  
 }  
}// end for each loop  
 // System.out.println(jan+" "+feb+" "+march+" "+april+" "+may+" "+june+" "+july+" "+august+" "+sep+" "+oct+" "+nov+" "+dec);  
  
  
//--> checking which month of friday is dead least  
if (jan>feb&&jan>march&&jan>april&&jan>may&&jan>june&&jan>july&&jan>august&&jan>sep&&jan>oct&&jan>nov&&jan>dec){  
 System.out.println("it's \"january\" on which the friday is deadliest");  
} else if (feb>jan&&feb>march&&feb>april&&feb>may&&feb>june&&feb>july&&feb>august&&feb>sep&&feb>oct&&feb>nov&&feb>dec){  
 System.out.println("it's \"February\" on which the friday is deadliest");  
}  
else if (march>feb&&march>jan&&march>april&&march>may&&march>june&&march>july&&march>august&&march>sep&&march>oct&&march>nov&&march>dec){  
 System.out.println("it's \"March\" on which the friday is deadliest");  
}  
else if (april>feb&&april>march&&april>jan&&april>may&&april>june&&april>july&&april>august&&april>sep&&april>oct&&april>nov&&april>dec){  
 System.out.println("it's \"April\" on which the friday is deadliest");  
 }  
else if (may>feb&&may>march&&may>april&&may>jan&&may>june&&may>july&&may>august&&may>sep&&may>oct&&may>nov&&may>dec){  
 System.out.println("it's \"May\" on which the friday is deadliest");  
}  
else if (june>feb&&june>march&&june>april&&june>may&&june>jan&&june>july&&june>august&&june>sep&&june>oct&&june>nov&&june>dec){  
 System.out.println("it's \"June\" on which the friday is deadliest");}  
else if (july>feb&&july>march&&july>april&&july>may&&july>june&&july>jan&&july>august&&july>sep&&july>oct&&july>nov&&july>dec){  
 System.out.println("it's \"July\" on which the friday is deadliest");}  
else if(august>feb&&august>march&&august>april&&august>may&&august>june&&august>july&&august>jan&&august>sep&&august>oct&&august>nov&&august>dec){  
 System.out.println("it's \"August\" on which the friday is deadliest");}  
else if (sep>feb&&sep>march&&sep>april&&sep>may&&sep>june&&sep>july&&sep>august&&sep>jan&&sep>oct&&sep>nov&&sep>dec) {  
 System.out.println("it's \"September\" on which the friday is deadliest");  
}  
else if (oct>feb&&oct>march&&oct>april&&oct>may&&oct>june&&oct>july&&oct>august&&oct>sep&&oct>jan&&oct>nov&&oct>dec) {  
 System.out.println("it's \"October\" on which the friday is deadliest");  
}  
else if (nov>feb&&nov>march&&nov>april&&nov>may&&nov>june&&nov>july&&nov>august&&nov>sep&&nov>oct&&nov>jan&&nov>dec) {  
 System.out.println("it's \"November\" on which the friday is deadliest");  
}else {  
 System.out.println("it's \"December\" on which the friday is deadliest");  
}  
  
}// deadliest month  
  
  
  
  
  
 public void rainyDayCrashes(){  
 int jan=0,feb=0,march=0,april=0,may=0,june=0,july=0,august=0,sep=0,oct=0,nov=0,dec=0;  
 System.out.println("Rain Day Crashes in Every Month: ");  
 for (Crashes c:queue){  
 if (c.getWeatherCondition().equalsIgnoreCase("rain") && c.getCrashMonth().equalsIgnoreCase("January")){  
 jan++;  
 }  
 if (c.getWeatherCondition().equalsIgnoreCase("rain") && c.getCrashMonth().equalsIgnoreCase("February")){  
 feb++;  
 }  
 if (c.getWeatherCondition().equalsIgnoreCase("rain") && c.getCrashMonth().equalsIgnoreCase("March")){  
 march++;  
 }  
 if (c.getWeatherCondition().equalsIgnoreCase("rain") && c.getCrashMonth().equalsIgnoreCase("April")){  
 april++;  
 }  
 if (c.getWeatherCondition().equalsIgnoreCase("rain") && c.getCrashMonth().equalsIgnoreCase("may")){  
 may++;  
 }  
 if (c.getWeatherCondition().equalsIgnoreCase("rain") && c.getCrashMonth().equalsIgnoreCase("june")){  
 june++;  
 }  
 if (c.getWeatherCondition().equalsIgnoreCase("rain") && c.getCrashMonth().equalsIgnoreCase("july")){  
 july++;  
 }  
 if (c.getWeatherCondition().equalsIgnoreCase("rain") && c.getCrashMonth().equalsIgnoreCase("august")){  
 august++;  
 }  
 if (c.getWeatherCondition().equalsIgnoreCase("rain") && c.getCrashMonth().equalsIgnoreCase("September")){  
 sep++;  
 }  
 if (c.getWeatherCondition().equalsIgnoreCase("rain") && c.getCrashMonth().equalsIgnoreCase("October")){  
 oct++;  
 }  
 if (c.getWeatherCondition().equalsIgnoreCase("rain") && c.getCrashMonth().equalsIgnoreCase("November")){  
 nov++;  
 }  
 if (c.getWeatherCondition().equalsIgnoreCase("rain") && c.getCrashMonth().equalsIgnoreCase("December")){  
 dec++;  
 }  
 }// end for each loop  
 System.out.println("January: "+jan+"\nFebruary: "+feb+"\nMarch: "+march+"\nApril: "+april+"\nMay: "+may+  
 "\nJune: "+june+"\nJuly: "+july+"\nAugust: "+august+"\nSeptember: "+sep+"\nOctober: "+oct+"\nNovember: "+nov+"\nDecember: "+dec);  
 }//end rainDayMonths  
  
  
  
  
  
 public void darkCondition(){  
// Is it true that the weekends (Saturdays and Sundays) when the light conditions were dark,  
// caused more crashes than Mondays and Tuesdays.  
  
 int satAndSun=0, monAndTues=0;  
 for (Crashes traverse: queue){  
 if ((traverse.dayOfWeek.equalsIgnoreCase("saturday") || traverse.dayOfWeek.equalsIgnoreCase("Sunday"))&&  
 traverse.getLightCondition().equalsIgnoreCase("dark lighted")){  
 satAndSun++;  
 }  
 if ((traverse.dayOfWeek.equalsIgnoreCase("Monday") || traverse.dayOfWeek.equalsIgnoreCase("Tuesday"))&&  
 traverse.getLightCondition().equalsIgnoreCase("dark lighted")){  
 monAndTues++;  
 }  
 }  
 System.out.println("Crashes when Dark Lighted on Monday and Tuesday: "+monAndTues);  
 System.out.println("Crashes when Dark Lighted on Saturday and Sunday: "+satAndSun);  
 if (monAndTues<satAndSun){  
 System.out.println("\"It is true that the weekends (Saturday and Sunday) caused more crashes\"");  
 } else System.out.println("\"It is false that the weekends (Saturday and Sunday ) caused more crashes\"");  
  
  
 }// end darkCondition  
  
  
  
  
  
  
// Make the tree to store the hit in data or not hit on data inside the tree and find the total data of each month an dthen find the least  
// and highest hitAndRun and not hitAndRun  
  
 int jan=0,feb,mar,apr,may,jun,july,aug,sep,oct,nov,dec;  
 int Njan,Nfeb,Nmar,Napr,Nmay,Njun,Njuly,Naug,Nsep,Noct,Nnov,Ndec;  
public void countHitAndRun(){  
 for (Crashes c: queue) {  
 // Calculate Hit and Run Counter  
 if (c.getCrashHitAndRun().equalsIgnoreCase("Hit-and-run") && c.getCrashMonth().equalsIgnoreCase("january")) jan++;  
 if (c.getCrashHitAndRun().equalsIgnoreCase("Hit-and-run") && c.getCrashMonth().equalsIgnoreCase("february")) feb++;  
 if (c.getCrashHitAndRun().equalsIgnoreCase("Hit-and-run") && c.getCrashMonth().equalsIgnoreCase("march")) mar++;  
 if (c.getCrashHitAndRun().equalsIgnoreCase("Hit-and-run") && c.getCrashMonth().equalsIgnoreCase("april")) apr++;  
 if (c.getCrashHitAndRun().equalsIgnoreCase("Hit-and-run") && c.getCrashMonth().equalsIgnoreCase("may")) may++;  
 if (c.getCrashHitAndRun().equalsIgnoreCase("Hit-and-run") && c.getCrashMonth().equalsIgnoreCase("june")) jun++;  
 if (c.getCrashHitAndRun().equalsIgnoreCase("Hit-and-run") && c.getCrashMonth().equalsIgnoreCase("july")) july++;  
 if (c.getCrashHitAndRun().equalsIgnoreCase("Hit-and-run") && c.getCrashMonth().equalsIgnoreCase("august")) aug++;  
 if (c.getCrashHitAndRun().equalsIgnoreCase("Hit-and-run") && c.getCrashMonth().equalsIgnoreCase("september")) sep++;  
 if (c.getCrashHitAndRun().equalsIgnoreCase("Hit-and-run") && c.getCrashMonth().equalsIgnoreCase("october")) oct++;  
 if (c.getCrashHitAndRun().equalsIgnoreCase("Hit-and-run") && c.getCrashMonth().equalsIgnoreCase("november")) nov++;  
 if (c.getCrashHitAndRun().equalsIgnoreCase("Hit-and-run") && c.getCrashMonth().equalsIgnoreCase("december")) dec++;  
 // System.out.println(jan+" "+feb+" "+mar+" "+apr+" "+may+" "+jun+" "+july+" "+aug+" "+sep+" "+oct+" "+nov+" "+dec);  
  
  
  
 // Calculate Not Hit And run Counters  
 if (c.getCrashHitAndRun().equalsIgnoreCase("Not Hit-and-run") && c.getCrashMonth().equalsIgnoreCase("january")) Njan++;  
 if (c.getCrashHitAndRun().equalsIgnoreCase("not Hit-and-run") && c.getCrashMonth().equalsIgnoreCase("february")) Nfeb++;  
 if (c.getCrashHitAndRun().equalsIgnoreCase("not Hit-and-run") && c.getCrashMonth().equalsIgnoreCase("march")) Nmar++;  
 if (c.getCrashHitAndRun().equalsIgnoreCase("not Hit-and-run") && c.getCrashMonth().equalsIgnoreCase("april")) Napr++;  
 if (c.getCrashHitAndRun().equalsIgnoreCase("not Hit-and-run") && c.getCrashMonth().equalsIgnoreCase("may")) Nmay++;  
 if (c.getCrashHitAndRun().equalsIgnoreCase("not Hit-and-run") && c.getCrashMonth().equalsIgnoreCase("june")) Njun++;  
 if (c.getCrashHitAndRun().equalsIgnoreCase("not Hit-and-run") && c.getCrashMonth().equalsIgnoreCase("july")) Njuly++;  
 if (c.getCrashHitAndRun().equalsIgnoreCase("not Hit-and-run") && c.getCrashMonth().equalsIgnoreCase("august")) Naug++;  
 if (c.getCrashHitAndRun().equalsIgnoreCase("not Hit-and-run") && c.getCrashMonth().equalsIgnoreCase("september"))Nsep++;  
 if (c.getCrashHitAndRun().equalsIgnoreCase("not Hit-and-run") && c.getCrashMonth().equalsIgnoreCase("october"))Noct++;  
 if (c.getCrashHitAndRun().equalsIgnoreCase("not Hit-and-run") && c.getCrashMonth().equalsIgnoreCase("november")) Nnov++;  
 if (c.getCrashHitAndRun().equalsIgnoreCase("not Hit-and-run") && c.getCrashMonth().equalsIgnoreCase("december")) Ndec++;  
  
 }  
 // System.out.println(jan+" "+feb+" "+mar+" "+apr+" "+may+" "+jun+" "+july+" "+aug+" "+sep+" "+oct+" "+nov+" "+dec);  
 // System.out.println(Njan+" "+Nfeb+" "+Nmar+" "+Napr+" "+Nmay+" "+Njun+" "+Njuly+" "+Naug+" "+Nsep+" "+Noct+" "+Nnov+" "+Ndec);  
}  
  
  
  
  
public BST createHitAndRunTree(){  
 countHitAndRun();  
 BST hitAndRun=new BST(jan);  
 hitAndRun.add(feb);  
 hitAndRun.add(mar);  
 hitAndRun.add(apr);  
 hitAndRun.add(may);  
 hitAndRun.add(jun);  
 hitAndRun.add(july);  
 hitAndRun.add(aug);  
 hitAndRun.add(sep);  
 hitAndRun.add(oct);  
 hitAndRun.add(nov);  
 hitAndRun.add(dec);  
 return hitAndRun;  
}  
  
  
public BST createNotHitAndRun(){  
 BST notHitAndRun=new BST(Njan);  
 notHitAndRun.add(Nfeb);  
 notHitAndRun.add(Nmar);  
 notHitAndRun.add(Napr);  
 notHitAndRun.add(Nmay);  
 notHitAndRun.add(Njun);  
 notHitAndRun.add(Njuly);  
 notHitAndRun.add(Naug);  
 notHitAndRun.add(Nsep);  
 notHitAndRun.add(Noct);  
 notHitAndRun.add(Nnov);  
 notHitAndRun.add(Ndec);  
 return notHitAndRun;  
}  
  
  
  
 public static void main(String[] args) throws IOException {  
 LinkedQueue q=new LinkedQueue();  
 q.formMainQueue();  
  
// q.subQueues();  
 System.out.println(" \*\*\*\*\*\*\*Problem No:01\*\*\*\*\*\*\*\n\n");  
 q.deadLeastOfAllDay();  
 q.deadLiestMonth();  
// System.out.println(q.monday.size());System.out.println(q.tuesday.size());System.out.println(q.wednesday.size());  
  
  
 System.out.println(" \*\*\*\*\*\*\*Problem No:02\*\*\*\*\*\*\*\n\n");  
 q.rainyDayCrashes();  
  
  
  
  
  
 System.out.println(" \*\*\*\*\*\*\*Problem No:03\*\*\*\*\*\*\*\n\n");  
  
  
 // For hit and run;  
 BST w=q.createHitAndRunTree();  
 // System.out.println(w);  
 int max=w.rightMost(w);  
 //Most hit and run  
 if (max==q.jan) System.out.println("Most Hit and Run occur on January= "+q.jan);  
 else if (max==q.feb) System.out.println("Most Hit and Run occur on February= "+q.feb);  
 else if (max==q.mar) System.out.println("Most Hit and Run occur on March= "+q.mar);  
 else if (max==q.apr) System.out.println("Most Hit and Run occur on April= "+q.apr);  
 else if (max==q.may) System.out.println("Most Hit and Run occur on February= "+q.may);  
 else if (max==q.jun) System.out.println("Most Hit and Run occur on February= "+q.jun);  
 else if (max==q.july) System.out.println("Most Hit and Run occur on February= "+q.july);  
 else if (max==q.aug) System.out.println("Most Hit and Run occur on February= "+q.aug);  
 else if (max==q.sep) System.out.println("Most Hit and Run occur on september= "+q.sep);  
 else if (max==q.oct) System.out.println("Most Hit and Run occur on october= "+q.oct);  
 else if (max==q.nov) System.out.println("Most Hit and Run occur on November= "+q.nov);  
 else if (max==q.dec) System.out.println("Most Hit and Run occur on December= "+q.dec);  
  
 System.out.println();  
 //Minimum hit and run  
 int min=w.leftMost(w);  
 if (min==q.jan) System.out.println("Minimum Hit and Run occur on January= "+q.jan);  
 else if (min==q.feb) System.out.println("Minimum Hit and Run occur on February= "+q.feb);  
 else if (min==q.mar) System.out.println("Minimum Hit and Run occur on March= "+q.mar);  
 else if (min==q.apr) System.out.println("Minimum Hit and Run occur on April= "+q.apr);  
 else if (min==q.may) System.out.println("Minimum Hit and Run occur on February= "+q.may);  
 else if (min==q.jun) System.out.println("Minimum Hit and Run occur on February= "+q.jun);  
 else if (min==q.july) System.out.println("Minimum Hit and Run occur on February= "+q.july);  
 else if (min==q.aug) System.out.println("Minimum Hit and Run occur on February= "+q.aug);  
 else if (min==q.sep) System.out.println("Minimum Hit and Run occur on september= "+q.sep);  
 else if (min==q.oct) System.out.println("Minimum Hit and Run occur on october= "+q.oct);  
 else if (min==q.nov) System.out.println("Minimum Hit and Run occur on November= "+q.nov);  
 else System.out.println("Minimum Hit and Run occur on December= "+q.dec);  
 System.out.println();  
  
 // For not hit and run  
 BST notHitAndRun=q.createNotHitAndRun();  
 //System.out.println(notHitAndRun);  
 int max1=notHitAndRun.rightMost(notHitAndRun);  
 if (max1==q.Njan) System.out.println("Most Not Hit and Run occur on January= "+q.Njan);  
 else if (max1==q.Nfeb) System.out.println("Most Not Hit and Run occur on February= "+q.Nfeb);  
 else if (max1==q.Nmar) System.out.println("Most Not Hit and Run occur on March= "+q.Nmar);  
 else if (max1==q.Napr) System.out.println("Most Not Hit and Run occur on April= "+q.Napr);  
 else if (max1==q.Nmay) System.out.println("Most Not Hit and Run occur on May= "+q.Nmay);  
 else if (max1==q.Njun) System.out.println("Most Not Hit and Run occur on June= "+q.Njun);  
 else if (max1==q.Njuly) System.out.println("Most Not Hit and Run occur on July= "+q.Njuly);  
 else if (max1==q.Naug) System.out.println("Most Not Hit and Run occur on August= "+q.Naug);  
 else if (max1==q.Nsep) System.out.println("Most Not Hit and Run occur on September= "+q.Nsep);  
 else if (max1==q.Noct) System.out.println("Most Not Hit and Run occur on October= "+q.Noct);  
 else if (max1==q.Nnov) System.out.println("Most Not Hit and Run occur on November= "+q.Nnov);  
 else System.out.println("Most Not Hit and Run occur on February= "+q.Ndec);  
  
 System.out.println();  
  
 int min1=notHitAndRun.leftMost(notHitAndRun);  
 if (min1==q.Njan) System.out.println("Minimum Not Hit and Run occur on January= "+q.Njan);  
 else if (min1==q.Nfeb) System.out.println("Minimum Not Hit and Run occur on February= "+q.Nfeb);  
 else if (min1==q.Nmar) System.out.println("Minimum Not Hit and Run occur on March= "+q.Nmar);  
 else if (min1==q.Napr) System.out.println("Minimum Not Hit and Run occur on April= "+q.Napr);  
 else if (min1==q.Nmay) System.out.println("Minimum Not Hit and Run occur on May= "+q.Nmay);  
 else if (min1==q.Njun) System.out.println("Minimum Not Hit and Run occur on June= "+q.Njun);  
 else if (min1==q.Njuly) System.out.println("Minimum Not Hit and Run occur on July= "+q.Njuly);  
 else if (min1==q.Naug) System.out.println("Minimum Not Hit and Run occur on August= "+q.Naug);  
 else if (min1==q.Nsep) System.out.println("Minimum Not Hit and Run occur on September= "+q.Nsep);  
 else if (min1==q.Noct) System.out.println("Minimum Not Hit and Run occur on October= "+q.Noct);  
 else if (min1==q.Nnov) System.out.println("Minimum Not Hit and Run occur on November= "+q.Nnov);  
 else System.*out*.println("Minimum Not Hit and Run occur on February= "+q.Ndec);  
  
 System.out.println("\n \*\*\*\*\*\*\*Postulate\*\*\*\*\*\*\*\n\n");  
 q.darkCondition();  
  
  
 }  
  
}  
  
class Crashes {  
 int crashYear, crashDay;  
 String timeOfDay, crashMonth, dayOfWeek, lightCondition, weatherCondition, crashIntersection, crashHitAndRun;  
 public Crashes(int crashYear, String crashMonth, int crashDay, String timeOfDay, String dayOfWeek, String crashIntersection,  
 String crashHitAndRun, String lightCondition, String weatherCondition) {  
 this.crashYear = crashYear;  
 this.crashDay = crashDay;  
 this.dayOfWeek = dayOfWeek;  
 this.timeOfDay = timeOfDay;  
 this.crashMonth = crashMonth;  
 this.lightCondition = lightCondition;  
 this.weatherCondition = weatherCondition;  
 this.crashIntersection = crashIntersection;  
 this.crashHitAndRun = crashHitAndRun;  
 }  
  
 public String toString(){  
 return crashYear+" "+crashMonth+" "+crashDay+" "+timeOfDay+" "+dayOfWeek+" "  
 +crashIntersection+" "+crashHitAndRun+" "+lightCondition+" "+weatherCondition+" ";  
 }  
  
 public String getDayOfWeek(){  
 return dayOfWeek;  
 }  
 public String getWeatherCondition(){  
 return weatherCondition;  
 }  
 public String getCrashMonth(){  
 return crashMonth;  
 }  
 public String getLightCondition(){  
 return lightCondition;  
 }  
 public String getCrashHitAndRun(){return crashHitAndRun;}  
  
  
}

class BST {  
 private int crashNo;  
 int height;  
 private BST left;  
 private BST right;  
 BST(){  
 this.right=this;  
 this.left=this;  
 this.height=-1;  
 }  
  
 public static final BST *Nil*=new BST();  
 public BST(int crashNo){  
 this.crashNo=crashNo;  
 right=left=*Nil*;  
 }  
  
public boolean add(int crashNo){  
 int old=size();  
 grow(crashNo);  
 return size() >old;  
}  
  
 public int size() {  
 if (this == *Nil*) {  
 return 0;  
 } else {  
 return 1 + left.size() + right.size();  
 }  
 }  
  
public BST grow(int crashNo){  
 if (this==*Nil*){return new BST(crashNo);}  
 if (crashNo==this.crashNo){return this;}  
  
if (crashNo<this.crashNo){  
 left=left.grow(crashNo);  
}  
else {  
 right=right.grow(crashNo);  
}  
  
  
height=1+Math.*max*(left.height, right.height);  
return this;  
}  
  
public static int rightMost(BST c){  
 if (c.right == *Nil*) {  
 return c.crashNo;  
 }  
 return c.*rightMost*(c.right);  
  
 // int rightmost;  
// Stack<BST> data=new Stack<>();  
// BST current=c;  
// while (c.right!=null || !data.isEmpty()){  
// if (current.right!=null) {  
// data.push(current);  
// current=current.right;  
// }else {  
// current=data.pop();  
// break;}  
// }// while  
//return current.crashNo;  
  
  
}  
  
public static int leftMost(BST c){  
 if (c.left == *Nil*) {  
 return c.crashNo;  
 }  
 return c.*leftMost*(c.left);  
}  
  
public String toString(){  
 if (this==*Nil*) return " ";  
 return left.toString()+" "+crashNo+" "+right.toString();  
}  
  
  
  
}

**OUTPUT:**

"C:\Program Files\Java\jdk-17.0.2\bin\java.exe" "-javaagent:C:\Program Files\JetBrains\IntelliJ IDEA Community Edition 2022.1\lib\idea\_rt.jar=50309:C:\Program Files\JetBrains\IntelliJ IDEA Community Edition 2022.1\bin" -Dfile.encoding=UTF-8 -classpath C:\Users\hp\IdeaProjects\Copy\_Crashes\out\production\Copy\_Crashes LinkedQueue  
 \*\*\*\*\*\*\*Problem No:01\*\*\*\*\*\*\*  
  
  
 DeadLeast day is Friday  
 it's "October" on which the friday is deadliest  
 \*\*\*\*\*\*\*Problem No:02\*\*\*\*\*\*\*  
  
  
 Rain Day Crashes in Every Month:  
 January: 23  
 February: 15  
 March: 27  
 April: 20  
 May: 17  
 June: 8  
 July: 5  
 August: 12  
 September: 22  
 October: 48  
 November: 36  
 December: 20  
 \*\*\*\*\*\*\*Problem No:03\*\*\*\*\*\*\*  
  
  
 Most Hit and Run occur on November= 238  
  
 Minimum Hit and Run occur on April= 170  
  
 Most Not Hit and Run occur on October= 473  
  
 Minimum Not Hit and Run occur on April= 276  
  
 \*\*\*\*\*\*\*Postulate\*\*\*\*\*\*\*  
  
  
 Crashes when Dark Lighted on Monday and Tuesday: 542  
 Crashes when Dark Lighted on Saturday and Sunday: 690  
 "It is true that the weekends (Saturday and Sunday) caused more crashes"  
  
 Process finished with exit code 0

**PBL ALGORITHUM**

1. Make a class of crashes which store the data of columns that are present in csv file.

* Make a constructor to initialize the data of columns that are present in data file.
* Getters are made to get the value of csv column.

**READ CSV FILE:**

Make another main class in which the method will be made to store data.

A main Queue is made in outer Main class.

Steps Of Reading data:

* Import the csv file on ide
* Use bufferedReader in the try catch block on which the location of file will be given
* Use bufferedObj.readLine() to read the file and insert the data separated by the comma in String array.
* Make an object of crash class and then stored the String array index into the constructor.
* Store this object of crash class in main CrashQueue.

**Making Sub Queue:**

Make the sub queue.

Make a for each loop to iterate the main CrashQueue and check:

If(carshes.getColumn().equal(“col value”))

Add that row in queue by add() mthod.

* All the subqueue of weathers and the 07 days of week will be making by the same manner.

**Problem 1:**

**Dead least Month:**

* Make the mothod of void type
* Compare all the size of of week days queue . The day queue having the greatest size will be the dead least of all days.

**Dead Friday of which Month:**

Make a method and declare the 07 counters of int type with the name of each month

Use the for each loop to traver main queue and check below condition

if (c.getDayOfWeek().equalsIgnoreCase("friday") && c.getCrashMonth().equalsIgnoreCase("January")){  
 jan++;  
 }

repeat the same step of all month. At the end we will have 12 counters of each month compare them and find which month has the greatest crashes on the day of Friday.

**Rainy day Crashes on each month:**

* Make a method of void type:
* Declare the 12 counters ;
* For each loop will be using to traverse the main Crashqueue

if (c.getWeatherCondition().equalsIgnoreCase("rain") && c.getCrashMonth().equalsIgnoreCase("January")){  
 jan++;  
}

This above condition will be check for all the month and then print each month counter to find the total number of rainy crashes in each month.

**Binary Search Tree:**

Make the Binary search tree class in which the total hit and not hit and run crashes will be stored.

**Method of LeftMost:**

Left most method inside BST class will be use to find the minimum hit and not hit and crashes

public static int leftMost(BST c){  
 if (c.left == *Nil*) {  
 return c.crashNo;  
 }  
 return c.*leftMost*(c.left);  
}

**Method of RightMost:**

Right most method inside BST class will be use to find the maximum hit and not hit and crashes

public static int rightMost(BST c){  
 if (c.right == *Nil*) {  
 return c.crashNo;  
 }  
 return c.*rightMost*(c.right);

**Make method to find the Hit and not hit and Run of all month:**

Use the 12 counters of each month to check the total hit and run in each month

* Traverse the main CrashQueue and check below condition:

if (c.getCrashHitAndRun().equalsIgnoreCase("Hit-and-run") && c.getCrashMonth().equalsIgnoreCase("january")) jan++;

the same condition will apply on all counters.

**Max and Min Hit and Run method:**

Compare the leftmost node by leftMost() method to all 12 counter

* Use if else to check which month counter is equal to the left most node key that will give us the required month in sop.
* Similarly doing with rightmost() to find Max value.

**Max and Min Not Hit and Run method:**

Compare the leftmost node by leftMost() method to all 12 counter of not hit and run months.

* Use if else to check which month counter is equal to the left most node key that will give us the required month in sop.
* Similarly doing with rightmost() to find Max value.

**Postulate:**

Declare the 02 counter 1st for the other week days and 2nd for the weekend days Saturday and Sunday.

Apply for each loop to travers the mainQueue and then apply condition:

if ((traverse.dayOfWeek.equalsIgnoreCase("saturday") || traverse.dayOfWeek.equalsIgnoreCase("Sunday"))&&  
 traverse.getLightCondition().equalsIgnoreCase("dark lighted")){  
 satAndSun++;  
}

if ((traverse.dayOfWeek.equalsIgnoreCase("saturday") || traverse.dayOfWeek.equalsIgnoreCase("Sunday"))&&  
 traverse.getLightCondition().equalsIgnoreCase("dark lighted")){  
 otherWeekDays++;  
}

compare both counters .

* If the weekend counter is greater than the week days counter the postulate will be true else false.