2. Amelia

Program Name: Amelia.java **Input File: amelia.dat**

Amelia is now a TA for ECS 1100 at her college program. She wants your assistance creating a sorting program to determine who the best students are, so that we can find out who deserves extra credit. The sorting algorithm is outlined as follows:

- First, sort the students based on attendance percentage (classes attended/total classes), as the students who go to all the classes deserve the best treatment (the professor's words, not ours), with the higher attendance percentage coming first.
- Next, if attendance is the same between two students, sort by improvement average between exam 1 to exam 2, and exam 2 to exam 3 (((exam2 – exam1) + (exam3 – exam2))/2), with the higher improvement coming first.
- Next, if the improvement scores are the same between two students, sort by number of daily quizzes completed, with the most quizzes coming first.
- If all other requirements are the same, sort alphabetically by last name then first name.

Input: The input will begin with an integer, `num` (`0 < num <= 1000`), denoting the number of test cases to follow. Each test case will begin with one integer, `n` (`0 < n < 1000`), denoting the number of students in the class to be sorted. Each line will begin with two strings, separated by spaces, denoting the first and last names of the student in question, respectively. These strings will be followed by 6 integers, ca, tc, e1, e2, e3, and dq, denoting the classes attended, total classes, exam 1 score, exam 2 score, exam 3 score, and daily quizzes completed, respectively. Each of the student records will appear on its own line, and all values (string and integer both) will be separated by spaces.

Output: Output the names of the students, first name then last name separated by spaces, followed by a space, followed by the average of the 3 exam grades rounded to 2 decimal places, each on its own line, in sorted order. Every test case should be followed by a line of 10 asterisks.

Sample input:

```
2
3
Benjamin Armstrong 10 11 82 89 95 9
Derrick Martin 8 11 70 80 90 11
Matthew Sheldon 10 11 92 93 94 11
2
William Armstrong 8 9 80 90 100 7
AppleBottom Jeans 6 9 70 85 100 8
```

Sample output:

```
Benjamin Armstrong 88.67
Matthew Sheldon 93.00
Derrick Martin 80.00
**********
William Armstrong 90.00
AppleBottom Jeans 85.00
**********
```

Amelia: Suggestions:

1. For architecture, consider storing the data for each individual in an object. As you build the objects load them into an ArrayList. Then sort the list using comparators. This problem was designed to make the comparators a bit painful.

Suppose you want to use a comparator to rank a double called test average and you had stored the test average (double) for each object as an instance variable “ave”. You want a comparator to compare Amelia objects with

You could use:

Comparator<Amelia> average = (a,b)-> (int) Math.signum(b.ave-a.ave); // comparators need to return an int.

```
import java.io.*;
import java.util.*;

public class Amelia {
int[] data;
String firstName;
String lastName;
public Amelia(int[] data, String firstName, String lastName) {
   this.data = data;
   this.firstName = firstName;
   this.lastName = lastName;
}
static  double improvement(Amelia a){
   double x = ((1.0*a.data[3]-a.data[2])+(1.0*a.data[4]-a.data[3]))/2;
   return x;
}

static double average(Amelia a){return (a.data[2]+a.data[3]+a.data[4])/3.0;}
   public static void main(String[] args) throws Exception{
      Scanner scan = new Scanner(new File("amelia.dat"));
      int n = scan.nextInt();
      Comparator<Amelia> last = (a,b) -> a.lastName.compareTo(b.lastName);
      Comparator<Amelia> first = (a,b) -> a.firstName.compareTo(b.firstName);
      Comparator<Amelia> attendance = (a,b) -> (int) Math.signum(1.0*b.data[0]/b.data[1]-
1.0*a.data[0]/a.data[1]);
      Comparator<Amelia> improvement  = (a,b) -> improvement(b)-improvement(a)>0?1:-1;
      Comparator<Amelia> quizes  = (a,b) -> b.data[5]-a.data[5];
      while(n-->0){

         List<Amelia> list = new ArrayList<>();
         int individuals = scan.nextInt();
         while(individuals-->0) {
            int[] tdata = new int[6];
            String fN = scan.next();
            String lN = scan.next();
            for (int i = 0; i < 6; i++) {
               tdata[i] = scan.nextInt();
            }
```

```
            list.add(new Amelia(tdata, fN, lN));
        }

list.sort(attendance.thenComparing(improvement).thenComparing(quizes).thenComparin
g(last).thenComparing(first));
            for(Amelia li : list) {
               System.out.print(li.firstName + " " + li.lastName);
               System.out.printf(" %.2f\n", average(li));
            }
            System.out.println("**********");

               }
   }
}
```

1. Aspen

Program Name: Aspen.java **Input File: none**

Aspen's team is excited about representing their region here at the championships. Help Aspen show how excited they are by displaying the phrase `2025 UIL State Championships!` in the diagonal manner shown below.

Input: None

Output: Print the phrase "`2025 UIL State Championships!`" in the pattern shown below.

Sample input: `None`

Sample output:

```
2025 UIL State Championships!
!2025 UIL State Championships
s!2025 UIL State Championship
ps!2025 UIL State Championshi
ips!2025 UIL State Championsh
hips!2025 UIL State Champions
ships!2025 UIL State Champion
nships!2025 UIL State Champio
onships!2025 UIL State Champi
ionships!2025 UIL State Champ
pionships!2025 UIL State Cham
mpionships!2025 UIL State Cha
ampionships!2025 UIL State Ch
hampionships!2025 UIL State C
Championships!2025 UIL State
 Championships!2025 UIL State
e Championships!2025 UIL Stat
te Championships!2025 UIL Sta
ate Championships!2025 UIL St
tate Championships!2025 UIL S
State Championships!2025 UIL
 State Championships!2025 UIL
L State Championships!2025 UI
IL State Championships!2025 U
UIL State Championships!2025
 UIL State Championships!2025
5 UIL State Championships!202
25 UIL State Championships!20
025 UIL State Championships!2
2025 UIL State Championships!
```

Melina Problem Statement

UIL – Computer Science Programming Packet – Region - 2025

7. Melina

Program Name: Melina.java **Input File: melina.dat**

You have made a new friend, Melina. She has trouble counting sometimes, but she recently accepted a job as a cashier at a taco shop. You need to write a program to determine how many different ways she can make the change required by the purchase each customer makes, given the money denominations she has in the cash register.

Input: The input will begin with an integer, `n` (`0 < n <= 1000`), denoting the number of test cases to follow. Each test case will consist of two lines. The first will contain two space-separated floating-point numbers, `a` and `t`, denoting the amount the customer paid, and the amount that was due. The following line will contain an unspecified number of space-separated integers denoting all the possible denominations of money Melina can use to make change, it can be assumed for our purpose that you have access to an infinite number of each denomination. Denominations will never be smaller than `.01` (1 cent), and neither `a` or `t` will exceed two decimal places. Additionally, `a` is guaranteed to be larger than `t`, and the difference between the two will never exceed `10^9`.

Output: For each test case, output the number of possible combinations Melina can make to make the amount of change required by the purchase each customer made.

Sample input:

```
3
25 24.78
.01 .05 .10 .25 1
40 39.15
.01 .05 .10 .25 1
50.26 41.95
.01 .05 .10 .25 1 2 5 10
```

Sample output:

```
9
163
332365
```

Hint: Change everything to ints first. Use DP. This is the key algorithm

n is the amount of change to be made. arr is an array of the change denominations available. Feel free to Google this.

```
static int combs(int n, int[] arr){
   if(n==0) return 1;
   if(n<0) return 0;
   int[] arr1 = new int[n+1];
   arr1[0]=1;
  for(int i: arr){
     for(int k = i; k<=n; k++){
        arr1[k] +=arr1[k-i];

     }

   }
 return arr1[n];

}
```

One way of doing this.

```
import java.util.*;
import java.io.*;

public class Melina {
   public static void main(String[] args) throws Exception{
      Scanner scan = new Scanner(new File("melina.dat"));
      int n = scan.nextInt();
      while(n-->0){
         int price = (int) (scan.nextDouble()*100);
         int paid = (int) (scan.nextDouble()*100);
         scan.nextLine();
         double[] change =
Arrays.stream(scan.nextLine().split("\\s+")).mapToDouble(Double::parseDouble).toArray();
         int[] changeInt = Arrays.stream(change).mapToInt(a->(int)(a*100)).toArray();
         System.out.println(combs(price-paid,changeInt));
         //System.out.println(Arrays.toString(changeInt));
        // System.out.println(price-paid);
      }

   }
   static int combs(int n, int[] arr){
      if(n==0) return 1;
      if(n<0) return 0;
      int[] arr1 = new int[n+1];
      arr1[0]=1;
     for(int i: arr){
        for(int k = i; k<=n; k++){
           arr1[k] +=arr1[k-i];

        }

      }
    return arr1[n];

   }
```

9. Raymond

Program Name: Raymond.java **Input File: raymond.dat**

Your classmate Raymond is a bit of an idiot. You've explained to him countless times what LCM (Lowest Common Multiple) means and how to find it between two numbers but he just doesn't get it. Feeling sorry for him, you've decided to go ahead and just write a program that given a list of numbers will let Raymond know the LCM.

Input: The first input line will contain a value N ($1 \le N \le 100$) denoting the number of test cases. Each test case will start with a single value M ($1 \le M \le 100$) representing the number of values in the list. The next line will contain M integers A such that ($1 \le A \le 1000$), of which you are to find the LCM between them all.

Output: Output "`Lowest Common Multiple is` **x**" where **x** is the lowest common multiple. If the LCM is 1, then instead print "`LCM NUMBER 1!`" to give Raymond a little excitement.

Sample input:

```
2
5
1 1 1 1 1
3
159 261 932
```

Sample output:

```
LCM NUMBER 1!
Lowest Common Multiple is 12892356
```

Math note: Given x1,x2 and x3. The LCM of x1 and x2 = x1*x2/GCD(x1 and x2). Call this y1 i.e. y1 = x1*x2/GCD(x1 and x2).

Then the LCM of x1,x2,x3 = lcm of y1 and x3.