|  |  |
| --- | --- |
| **Pre-AP® Computer Science** | **Lab 11A**  **Practice/Perform Major Java Assignment** |
| **The “Mean, Median and Mode” Program** | **80, 90, 100 & 110 Point Versions** |
| **Assignment Purpose:**  This program requires knowledge of creating and manipulating arrays.  Additionally, it requires an understanding of the three types of "averages." | |

Mathematics recognizes three levels of "centrality" or averages. They are called the *mean*, the *median* and the *mode*. The *average* most familiar to student is the *mean*.

The *mean* is computed by adding up the sum of all the numbers and then dividing by the count of the numbers. This type of average is normally used to find the class average for a test.

The *median* is the middle number in a list of numbers. This type of average is frequently used to find the average home value in a city or neighborhood. Using the *mean* can distort the average. It is possible that an extremely expensive home is included. With the mean computation the expensive home will raise the average to an unrealistic average.

The *mode* is the number that occurs the most frequently in a list of numbers.

|  |  |
| --- | --- |
| **Lab 11A Student Version** | **Do not copy this file, which is provided.** |
| // Lab11Ast.java  // The "Mean, Median and Mode" Program  // This is the student, starting version of Lab 11A.  import java.util.Arrays;  import java.util.Random;  public class Lab11Ast  {  public static void main(String args[])  {  System.out.println("\nLab 11A\n");  System.out.print("Enter the quantity of random numbers ===>> ");  int listSize = Expo.enterInt();  System.out.println();  Statistics intList = new Statistics(listSize);  intList.randomize();  intList.computeMean();  intList.computeMedian();  intList.computeMode();  intList.displayStats();  System.out.println();  }  }  class Statistics  {  private int list[]; // the actual array of integers  private int size; // user-entered number of integers in the array  private double mean; // used for the 80, 100 and 110 point versions  private double median; // used for the 100 and 110 point versions  private int mode; // used for the 110 point version only  public Statistics(int s)  {  size = s;  list = new int[size];  mean = median = mode = 0;  }  public void randomize()  {  **// This provided method creates the same exact list of "random" numbers for every execution.**  **// You will learn more about this in Chapter 14. For now just use the provided method.**  Random rand = new Random(12345);  for (int k = 0; k < size; k++)  list[k] = rand.nextInt(31) + 1; // range of 1..31  }  public void computeMean()  {  }  public void computeMedian()  {  }  public void computeMode()  {  // precondition: The list array has exactly 1 mode.  }  public void displayStats()  {  System.out.println(Arrays.toString(list));  System.out.println();  System.out.println("Mean: " + mean);  System.out.println("Median: " + median);  System.out.println("Mode: " + mode);  }  } | |

**80-Point Version Specifics**

For the 80-point version your program will only compute the **mean**. This means that you need to complete the **computeMean** method. The provided **displayStats** method will display the **mean**, the **median**, and the **mode**. For the 80-point version, a value of **0** will be displayed for the **median** and the **mode**. Run the program twice, first with **20** numbers, and then with **21**.

**80-Point Version Output #1**

|  |
| --- |
| ÏÏ«Ï ----jGRASP exec: java Lab11Av80 ÏÏ§Ï ÏÏ§ÏLab 11A ÏÏ§Ï ¼¼§ÏEnter the quantity of random numbers ===>> 20 ÏÏ§Ï ÏÏ§Ï[15, 26, 24, 22, 4, 21, 13, 14, 26, 9, 18, 12, 11, 28, 30, 24, 26, 2, 19, 3] ÏÏ§Ï ÏÏ§ÏMean: 17.35 ÏÏ§ÏMedian: 0.0 ÏÏ§ÏMode: 0 ÏÏ§Ï ÏÏ©Ï ----jGRASP: operation complete. |

**80-Point Version Output #2**

|  |
| --- |
| ÏÏ«Ï ----jGRASP exec: java Lab11Av80 ÏÏ§Ï ÏÏ§Ï ÏÏ§ÏLab 11A ÏÏ§Ï ¼¼§ÏEnter the quantity of random numbers ===>> 21 ÏÏ§Ï ÏÏ§Ï[15, 26, 24, 22, 4, 21, 13, 14, 26, 9, 18, 12, 11, 28, 30, 24, 26, 2, 19, 3, 9] ÏÏ§Ï ÏÏ§ÏMean: 16.952380952380953 ÏÏ§ÏMedian: 0.0 ÏÏ§ÏMode: 0 ÏÏ§Ï ÏÏ§Ï ÏÏ©Ï ----jGRASP: operation complete. |

**90 & 100-Point Version Specifics**

The 100-point version displays the **mean** and the **median**. This means that you only need to complete both the **computeMean** and the **computeMedian** methods. The median is the middle number in a sorted list. The list is currently random, so it will need to be sorted before you can do anything else. Use the **Arrays** class for this. After the list is sorted, the computer needs to determine if this is an *odd* list or an *even* list. This is necessary because computing the **median** is actually done differently in each situation. If the number of integers in the list is *odd*, the **median**is simply the integer in the exact middle. If the number of integers in the list is *even*, computing the **median**becomes more challenging. An *even* list has 2 integers in the middle. To find the **median** of an *even* list, you need to find the **mean** of these 2 middle numbers.

**90 &100-Point Version Output #1**

|  |
| --- |
| ÏÏ«Ï ----jGRASP exec: java Lab11Av100 ÏÏ§Ï ÏÏ§Ï ÏÏ§ÏLab 11A ÏÏ§Ï ¼¼§ÏEnter the quantity of random numbers ===>> 20 ÏÏ§Ï ÏÏ§Ï[2, 3, 4, 9, 11, 12, 13, 14, 15, 18, 19, 21, 22, 24, 24, 26, 26, 26, 28, 30] ÏÏ§Ï ÏÏ§ÏMean: 17.35 ÏÏ§ÏMedian: 18.5 ÏÏ§ÏMode: 0 ÏÏ§Ï ÏÏ§Ï ÏÏ©Ï ----jGRASP: operation complete. |

**90 &100-Point Version Output #2**

|  |
| --- |
| ÏÏ«Ï ----jGRASP exec: java Lab11Av100 ÏÏ§Ï ÏÏ§Ï ÏÏ§ÏLab 11A ÏÏ§Ï ¼¼§ÏEnter the quantity of random numbers ===>> 21 ÏÏ§Ï ÏÏ§Ï[2, 3, 4, 9, 9, 11, 12, 13, 14, 15, 18, 19, 21, 22, 24, 24, 26, 26, 26, 28, 30] ÏÏ§Ï ÏÏ§ÏMean: 16.952380952380953 ÏÏ§ÏMedian: 18.0 ÏÏ§ÏMode: 0 ÏÏ§Ï ÏÏ§Ï ÏÏ©Ï ----jGRASP: operation complete. |

NOTE: To earn a grade of **100** your program must work for both of the outputs above.

If your program only works for one of the outputs, you will be assigned a grade of **90**.

This means your program works for *even* lists only or *odd* lists only, but not both.

**110-Point Version Specifics**

The 110-point version adds the **mode** computation. This process is easily stated. Find the number that occurs most frequently. This is definitely not a trivial task. There is a reason why this is for extra credit.

NOTE: You can take advantage of the fact that the list is now sorted.

All like numbers are grouped together.

**110-Point Version Output #1**

|  |
| --- |
| ÏÏ«Ï ----jGRASP exec: java Lab11Av110 ÏÏ§Ï ÏÏ§Ï ÏÏ§ÏLab 11A ÏÏ§Ï ¼¼§ÏEnter the quantity of random numbers ===>> 20 ÏÏ§Ï ÏÏ§Ï[2, 3, 4, 9, 11, 12, 13, 14, 15, 18, 19, 21, 22, 24, 24, 26, 26, 26, 28, 30] ÏÏ§Ï ÏÏ§ÏMean: 17.35 ÏÏ§ÏMedian: 18.5 ÏÏ§ÏMode: 26 ÏÏ§Ï ÏÏ§Ï ÏÏ©Ï ----jGRASP: operation complete. |

**110-Point Version Output #2**

|  |
| --- |
| ÏÏ«Ï ----jGRASP exec: java Lab11Av110 ÏÏ§Ï ÏÏ§Ï ÏÏ§ÏLab 11A ÏÏ§Ï ¼¼§ÏEnter the quantity of random numbers ===>> 21 ÏÏ§Ï ÏÏ§Ï[2, 3, 4, 9, 9, 11, 12, 13, 14, 15, 18, 19, 21, 22, 24, 24, 26, 26, 26, 28, 30] ÏÏ§Ï ÏÏ§ÏMean: 16.952380952380953 ÏÏ§ÏMedian: 18.0 ÏÏ§ÏMode: 26 ÏÏ§Ï ÏÏ§Ï ÏÏ©Ï ----jGRASP: operation complete. |

NOTE: It is technically possible for a list of numbers to have “multiple modes” or “no mode”.

For this assignment you are not concerned with either of these cases.