Our Flounder

Computer Science Correspondence School

discere domi

Lecture 21

CISC 131
Introduction to Programming and Problem Solving
Spring 2020
Array Processing: Statistics

Due: **Friday, May 8, 2020, at start of class**
Points: **20**

Assignment

In this assignment you will write a program that calculates descriptive statistics based on user input that is done through the *window.prompt*. You must use some of the one dimension array functions from previous assignments and write a few additional functions. No *css* is needed and only the minimum amount of *html* necessary to launch your Javascript program.

You will need to use the *Math.pow* function. This function is used to raise a value to a power and is passed two numbers: *x* and *y*. The function returns x^y, For example:

```
var a;
var b;
a = 4.5;
b = 3;
window.alert( Math.pow(a,b) );
```

The dialog box would contain the value 91.125 which is $(4.5)^3$. There are more examples on the *W3School* site. Test as you develop the program. Testing as you develop is the quickest way to write and debug your program.

The program will need some of the array processing functions you have already written:

`arrayDisplay ( array )`	Lecture 8, April 7, 2020
`sum ( array )`	Lecture 9, April 8, 2020
`getDelimiterLocations ( line, delimiter )`	Lecture 17, April 28, 2020
`split ( line, delimiter )`	Lecture 17, April 28, 2020

and some new ones, as well.

Step 1

The first function is pretty straightforward.

```
function trimArray(array)
```

The function is passed an array and returns an array. The returned array contains only those elements in the parameter array that are

- not *null* and
- not zero length *Strings* after all leading and trailing *whitespace* has been removed

Each element in the returned array must have all leading and trailing whitespace[1] removed from it by using the built-in Javascript *String trim()*[2] method. You will need to use the *count-create-populate* approach in this function. Test it out to make sure it works correctly.

[1] Recall that whitespace is the general term used for characters that are not visible when displayed on the monitor or printed. The common whitespace characters are: space, tab, newline, return, and form feed. Other than the space

Step 2
The next function is:

```
function convertToNumber(array)
```

The function is passed an array and converts each element of the parameter array into its number equivalent, if it has one. The function returns a *String* that contains the comma separated values in the array that cannot be converted into numbers. The function returns a zero length *String* if all the elements of the array were successfully converted into numbers. This function modifies the contents of the parameter array. You will need to use the *isNumeric* and *toNumber* functions that you wrote for Lecture 19, May 1, 2020.

Step 3
Use this *window.onload* function:

```
window.onload = function()
{
 var array;
 var userData;

 userData = window.prompt('Enter the whitespace or comma separated values: ','');
 array    = split ( userData, ' \t\r\n\f,' );
 array    = trimArray ( array );

 userData = convertToNumber(array);

 if(userData.length > 0)
 {
  window.alert("Errors were found. These values are not numeric: " +
               userData);
 }
 else
 {
 // window.alert(calculateStatistics(array));
 }
};
```

The *window.onload* function uses a *window.prompt* to ask the user to enter any amount of numbers each separated by any number of whitespace or comma characters before or after a number. It will convert the data into a numeric array using the *split* and *convertToNumber* functions. As a temporary measure, display the array and verify that its contents are correct. Use your *arrayDisplay* function to help do this. When you are certain things are working correctly, remove the display code from the *window.onload* function.

Notice that the call to *calculateStatistics* is commented out. You will write that function in Step 5 at which time you should uncomment the call statement.

Step 4
Write the following descriptive statistics functions. Each is passed a one dimension array of numbers. The functions assume that the passed array contains numbers and the functions do not check to determine if they actually do contain numbers. This means that you do not need to write statements in these functions that check to see if the array contents are numeric. The statements

---

character, you must use escape codes to represent whitespace in your Javascript program. A *String* that contained all of the above whitespace characters (including the space) would look like this: " `\t\n\r\f`".
[2] See Lecture 3, March 20, 2020 for more information about the *String trim()* method.

you write assumes numeric content so make sure you pass only numeric arrays to these functions.

You will need to do some research to determine the algorithm and minimum number of values needed to calculate each statistic. For example, the array must contain at least one element in order to calculate the *mean* value. The other statistics will have similar requirements.

You may not use any built-in *Javascript* statistics functions. You must write your own.

The *getMaximum* and *getMinimum* functions return the element number of the maximum (or minimum) value and not the value itself. If a zero length array is sent to these functions, they return -1;

The *getMean* function returns the mean value of the array. If the mean value cannot be determined because the array contains no values, *NaN* is returned.

```
function getMaximum(array)
function getMean(array)
function getMinimum(array)
function getSampleStandardDeviation(array)
```

Nota bene, the last function listed above calculates the *sample* standard deviation and not the *population* standard deviation. You will need to use the *Math.pow* function. Square root is represented by using .5 as the power value. If the array does not contain sufficient values to calculate the sample standard deviation, the function returns *NaN*.

Make sure you document each function to describe the different values that it returns. Test each one thoroughly.

Step 4
After you write this function, uncomment the call to this function in the *window.onload*.

Write a function called *calculateStatistics* that is passed an array of numbers. The function returns a *String*. The function calls the four statistics functions described in Step 4. The *String* returned by the function contains one line of information describing the result of calling each of the statistics functions. Each line must contain either the statistic that was calculated or a message saying that it was not possible to calculate that statistic. In addition to the four lines of information obtained from calling the statistics functions, the first line of the *String* contains a message indicating how many (*n*) values were in the array. Here is some sample output when the numbers 3, 4, 5, and 6 were entered by the user:

```
n:   4
minimum: 3
maximum: 6
mean:    4.5
sample std dev:    1.2909944487358056
```

You can check your function results by using this online site:
http://www.easycalculation.com/statistics/standard-deviation.php
but note that it computes both the population standard deviation and the sample standard deviation. It is the sample standard deviation that is required by this assignment, so look at site's output carefully.

As usual, test early, test often, test thoroughly. Make sure your tests include zero length arrays, arrays that contain non-numbers, and arrays that contain zero length *Strings*.

What You Must Turn In

- Send me an email with this subject line:

  CISC131-Statistics
- Attach the ***YourLastName*-Statistics.js** file to the email message and send it to me.
- Do NOT attach any other files.
- Do not zip up the Javascript file.