Create – Applications From Ideas
Written Response Submission Template

Please see Assessment Overview and Performance Task Directions for Student for the task directions and recommended word counts.

Program Purpose and Development

2a)

Parents of teenagers often worry about their children's safety on the road. The cause of car accidents involving teenagers is typically due to speeding and distracted driving because of their lack of experience. Also, teenagers with uninvolved parents are more prone to get into accidents. Therefore, for the safety of young drivers, parents should put in the effort to take on a more authoritative role in their kids' lives.

This program analyzes the statistics of fatal crashes involving teens related to parenting styles. Written in Google Apps Script, my program analyzes data collected from NHTSA's Teen Crash Statistics [1], Wiki [2] and a Pediatric Journal [3]. My video demonstrates:

a) How to create a merged data tab from two Google sheets
b) MATH calculation of Maximum and Minimum number of crashes

Analyzing this data, I can conclude that parents in Texas would benefit the most by utilizing more authoritative parenting.

2b)

I encountered several difficulties while writing the code for this program. The first difficulty was converting strings to numbers (16 years 6 months to 16.5) that were used for plotting a map on the ArcGIS website. I resolved this by creating my own Google App Script function which reads the string, splits it based on blank space, and divides the month's portion by 12 to get the decimal value. The second difficulty was calculating the minimum and maximum teen fatalities. I was not sure if I had to write my own function or not, but after some research, my mentor suggested using the Google Apps Script API to call the math functions. This difficulty was solved by iterating through the values of the spreadsheet for teen fatalities, and applying a math formula to calculate the minimum and the maximum of the data set. I also used a lot of logging to debug

issues that came up while iterating through the sheets, one of them being that I was not sure if my variables were getting the actual values. The logger.log function helped me by printing the values I needed into the log file, which allowed me to debug.

2c)

First Algorithm

```
          //function to get max value from a given column of values
   function getmaxval(sheetName,colindex){
    var ss = SpreadsheetApp.getActiveSpreadsheet().getSheetByName(sheetName);
    var numRows = ss.getLastRow() -1;
    var max1 = 0;
    for(var i=3; i<numRows; i++) {      //loop through each row to get the max value
     val2 = ss.getRange(i, colindex).getValue();    //get actual value from the cell
     max1 = Math.max(max1, val2);       //use the MATH function to get the max value
//    Logger.log(max1);
      }
       return(max1)                                 // return the max value
     }
```

Second Algorithm

```
        //function to get min value from a given column of values
  function getminval(sheetName,colindex){
    var ss = SpreadsheetApp.getActiveSpreadsheet().getSheetByName(sheetName);
    var numRows = ss.getLastRow() -1;
    var min1 = 1000000;

  for(var i=3; i<numRows; i++) {          loop through each row to get the min value
    val2 = ss.getRange(i, colindex).getValue(); //get the actual value from the cell
    min1 = Math.min(min1, val2);        //use the MATH function to get the min value
    // Logger.log(min1);
  }
  return(min1)                                  // return the min value
 }
```

Third Algorithm

```
          //function to convert[illegible]e from String to Decimal format
  function ConvertAge(label){       //Uses min age requirements as parameter
    var spltext = label.split(' ');  //Splits the String using blank as a separator
    var age = Number(spltext[0]);    //Obtains the years in whole numbers

   if(spltext[2]!= null){           //if there is months then converts into decimal
     var months = Number(spltext[2]/12); //divide months by 12 to get decimal format
     var newage = age + months;  //Adds the years with the months to create a number
     // Logger.log(spltext[2])
     Logger.log(newage)
     return(newage);                  //returns values
   } else {
     Logger.log(age)
     return(age);
   }
```

The first algorithm getmaxval() that I came up was to get the maximum number of teen crashes by iterating through each row of the spreadsheet. I do this by determining the maximum number of rows, and using a for loop to iterate through each value. I then use the Math.max function which compares two values and keeps track of the highest value. At the end of the method, I use a return statement to return the maximum value saved. Similarly, the second algorithm getminval() was created to find the minimum number of teen crashes by iterating through each value of each row in the spreadsheet. However, I used a different math function, Math.min, to compare the values with each other to

find the minimum value. At the end of the method, I use a return statement to return the minimum value. Finally, the last algorithm ConvertAge() converts a string value to a float value by locating the white space within the strings. Then, I integrated logic by using a condition to check if the variable value contained months. If it did, I would use the standard of 12 months to convert the number of months into a decimal value.

2d)

```
function TeenStatsFunction() {

  //declare variables section
  var ss = SpreadsheetApp.getActiveSpreadsheet().getSheetByName("Min Age by
      State");  //Obtains Spreadsheet for minimum age requirements

  var ss2 = SpreadsheetApp.getActiveSpreadsheet().getSheetByName("Teen
      Crashes");  //Obtains Spreadsheet for teen crashes

  var ss_new =
      SpreadsheetApp.getActiveSpreadsheet().getSheetByName("Merged");  //Obtains
      Spreadsheet for merged spreadsheets

  var colindex = columnIndex("State","Min Age by State");  //Getting the states from
       the  minimum age requirements spreadsheet
  var numRows = ss.getLastRow() -1;  //Obtaining the last row of the states
  var colValues = ss.getRange(3, colindex, numRows, 1).getValues() //Sequentially
       going through each state, and saves the values
       Logger.log(colValues) //print to debug

  var colindex1 = columnIndex("Minimum Age for Restricted License","Min Age by
       State");  //Gets the Minimum Age Column from the spreadsheet

  var colindex2 = columnIndex("Total Fatalities","Teen Crashes");  //Gets the number
       of teen crashes column from the spreadsheet

  var colValues1 = ss.getRange(3, colindex1, numRows, 1).getValues()  //Sequentially
       going through each state, and saves the values
  var colValues2 = ss2.getRange(3, colindex2, numRows, 1).getValues() //Sequentially
       going through each state, and saves the values


... ...
... ..

//function to get the column index inside a sheet
function columnIndex(label,sheetName){
 var ss = SpreadsheetApp.getActiveSpreadsheet().getSheetByName(sheetName);
 var lc = ss.getLastColumn();
 var lookupRangeValues = ss.getRange(2, 1, 1, lc).getValues()[0];
 var index = lookupRangeValues.indexOf(label) + 1
 return(index);
}
```

This code implements abstraction by calculating Teen driver statistics from different data sources. This includes the minimum age requirements to obtain a license and the amount of teen crashes categorized by state. Inclusion of abstraction has made my code more efficient and compact by creating a single method that can be used at various points in the code. The complexity lies in the iteration of the columnIndex() method to obtain the index of the column in the spreadsheet that contains the values defined by the labels. This process is reused multiple times to obtain data from various sources to increase efficiency. My code includes mathematical concepts to determine the minimum and maximum of the data sets, as well as mathematical formulas to convert months into decimals for the age requirement information. My code implements logic by using conditions when determining if the age requirement contains months or not. If it does, the ConvertAge() method runs and converts the months into years using mathematical functions. Additionally, my code implements logic by using for loops when trying to obtain information from certain columns in the data set.