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)

I wrote this program in JavaScript using the Code.org's App Lab programming environment. The purpose of my program is to calculate GPA, both weighted and unweighted. The instructions page gives the user clear directions on how to input their data and how to use the functions of the app. The default number of subjects is 8. The user can add up to 10 subjects. The number of subjects can be added or deleted by pressing the buttons Add/Delete Subject. My video illustrates how the user can input the subject name and select the level and grade from the dropdown menus. All fields must be filled in or else an error message is displayed. When the user presses the Calculate GPA button, it displays his/her weighted and unweighted GPA. From that page, the user can either go back and edit their input information or start over and enter new data.

2b)

I completed this project independently. However, I collaborated with my classmate for testing. I developed this program with a series of iterations. I began developing my program by using a default of eight subjects. Then, I focused on implementing the core functionality, which was to read the user inputs and calculate GPA. In the second iteration, I added a function to my program for data validation. Over a few more iterations, I resolved several issues that I noticed, such as rounding GPA to three decimals and fixing the calculations when the back button was used. After a peer tested my program, she gave me feedback that the start over button did not reset the values of the user inputs. I overcame this difficulty by creating a function that reset all the input controls and also changed the number of subjects back to eight, which is the default. One opportunity I identified by testing an early version of my program was to allow flexibility to add or remove the number of subjects as per the user needs. To address this, I incorporated the buttons Add Subject and Delete Subject into my program. Now, the user can have one to ten subjects.

2c)

If your captured code segment is an image, click the Picture icon to browse to the location of your saved image. If your captured code segment is text, include it with the rest of your 2c written response below.

Main Algorithm:

```
function calculateGPA() {
//calculateGPA function calculates the GPA based on the data entered
//It calls convertGradeToPoints and convertGradePointsToWeight functions
//Main algorithm (Parent)

for (var i = 1; i < numberOfSubjects+1; i++)
{
 var gradeDropdownId = "gradeDropdown"+i.toString();
 var gradex = "grade"+i.toString();
 var levelDropdownId = "levelDropdown"+i.toString();
 var levelx = "level"+i.toString();
 var gpaxWeighted = "gpa"+i.toString();
 var gpaxUnweighted = "gpa"+i.toString();

 gradex = getText(gradeDropdownId);
 levelx = getText(levelDropdownId);

 setScreen("displayScreen");
 //Calcution of GPA
 convertGradeToPoints(gradex);
 convertGradePointsToWeight(levelx);
 gpaxWeighted = gpaWeighted;
```

```
  gpaxUnweighted = gpaUnweighted;
  sumGPAWeighted = sumGPAWeighted+gpaxWeighted;
  sumGPAUnweighted = sumGPAUnweighted+gpaxUnweighted;
}

finalGPAWeighted = (sumGPAWeighted/numberOfSubjects).toFixed(3);
setText("textAreaGPAWeighted", finalGPAWeighted);

finalGPAUnweighted = (sumGPAUnweighted/numberOfSubjects).toFixed(3);
setText("textAreaGPAUnweighted", finalGPAUnweighted);

//console.log ("number of Subjects is " + numberOfSubjects );

sumGPAWeighted = 0;
sumGPAUnweighted = 0;
}

function convertGradeToPoints(grade) {
//convertGradeToPoints function assigns the GPA value based on the Grade
//Child Algorithm 1
    if (grade == "A+"){
    gpaUnweighted = 4.333;
    }
    else if (grade == "A"){
    gpaUnweighted = 4.0;
    }
    else if (grade == "A-"){
    gpaUnweighted = 3.666;
    }
    else if (grade == "B+"){
    gpaUnweighted = 3.333;
    }
    else if (grade == "B"){
    gpaUnweighted = 3.0;
    }
    else if (grade == "B-"){
    gpaUnweighted = 2.666;
    }
    else if (grade == "C+"){
    gpaUnweighted = 2.333;
    }
    else if (grade == "C"){
    gpaUnweighted = 2.0;
    }
    else if (grade == "C-"){
    gpaUnweighted = 1.666;
    }
    else if (grade == "D+"){
```

```
    gpaUnweighted = 1.333;
    }
    else if (grade == "D"){
    gpaUnweighted = 1.0;
    }
    else if (grade == "D-"){
    gpaUnweighted = 0.666;
    }
    else if (grade == "F"){
    gpaUnweighted = 0;
    }
}

function convertGradePointsToWeight(level){
//convertGradePointsToWeight function is used to calculate the weighted GPA
//Child Algorithm 2
  if (level == "Honors"){
    gpaWeighted = gpaUnweighted + 0.5;}
    else if (level == "AP"){
    gpaWeighted = gpaUnweighted + 1.0;}
    else if (level == "Regular"){
    gpaWeighted = gpaUnweighted;}
}
```

Written Response 2c:

The main algorithm that I selected is calculateGPA(). This algorithm has two key parts, convertGradeToPoints(grade) and convertGradePointsToWeight(level). The convertGradeToPoints(grade) algorithm is designed to calculate GPA, based on the grade entered. This function takes grade as a parameter and assigns a corresponding GPA value based on some predefined values. Several selection statements are used to cover all the grades from A+ to F. The convertGradePointsToWeight(level) algorithm uses mathematical and logical concepts and is designed to modify the GPA value, based on the level entered. The GPA value is incremented for Honors (+0.5) and AP (+1.0) subjects. Together these algorithms help achieve the purpose of my program, which is to calculate both weighted and unweighted GPA. The main algorithm, calculateGPA, reads the value of inputs, which are levelx and gradex for each subject. Then, it passes these as parameters to the sub-algorithms, convertGradeToPoints(grade) and convertGradePointsToWeight(level) to calculate GPA. The GPA is calculated for all the subjects in the same way, using a loop. The final GPA is calculated by adding the GPA of all subjects and dividing by the number of subjects, using the global variable, numberOfSubjects. Finally, the GPA value is rounded to three decimal places.

2d)

If your captured code segment is an image, click the Picture icon to browse to the location of your saved image. If your captured code segment is text, include it with the rest of your 2d written response below.

Abstraction:

```
function dataValidation(colName){
//dataValidation function validates the data for any empty values
  for (var i = 0; i<numberOfSubjects; i++)
  {
     if (data[i][colName]===""){
     isDataValid="No";
  }
 }
}
```

Written Response 2d:

One abstraction that I developed myself was a function, called dataValidation(colName). This function uses the column name as a parameter. This abstraction helps manage complexity in my code since it allows me to put this behavior, that I need to use multiple times, in a single function. I would need to include this code at least three places in my program, which would make it messy and harder to understand and maintain. For instance, I would have to write almost the exact lines of code when validating the data for user inputs subject name, level, and grade. Additionally, this action had to be checked for all the subjects, so I added a loop in this function itself to make my program less intricate. I used mathematical and logical concepts in this function, like incrementing the subject Name counter, checking the condition with less than operator. As this function is called at multiple places in the program by passing different values to the parameter, I was able to reuse it for all the input data validation. Without this function as my abstraction, the program would be more difficult to manage because it increases the complexity and lines of code.