UNIVERSITY OF SOUTHERN QUEENSLAND
CSC1401 - Foundation Programming (Semester 2, 2018)
Assignment II Specification

Personal GPA Calculator

Due date: 13 Sep 2018

Weight: 12%

Type: Individual

Goals and Topics

The assignment problem is straightforward. All necessary details have been supplied. The solution of the problem will use the programming concepts and strategies covered in Workshops 1-7. The subgoals are:

- Obtaining advanced understanding of values, variables and arrays;
- Understanding program input and output, functions and expressions;
- Understanding simple strategies like iteration, validation, sum, count, minimum, and maximum plans;
- Translating simple design into JavaScript code
- The mechanics of editing, interpreting, building, running and testing a program
- Commenting source code
- Becoming confident and comfortable with programming in small problems

Background

Grading in education is the process of applying standardised measurements of varying levels of achievement in a course. Another way GPA (Grade Point Average) can be determined is through extra curricular activities. Grades can be assigned as letters (generally A through F), as a range (for example 1 to 6), as a percentage of a total number of questions answered correctly, or as a number out of a possible total (for example out of 20 or 100).

In some countries, grades from all current classes are averaged to create a grade point average (GPA) for the marking period. The GPA is calculated by taking the number of grade points a student earned in a given period of time of middle school through high school. GPAs are also calculated for undergraduate and graduate students in most universities. The GPA can be used by potential employers or educational institutions to assess and compare applicants. A cumulative grade point average is a calculation of the average of all of a student's grades for all of his or her complete education career.

In USQ, a Grade Point Average (GPA) is defined as "the average of all your final grades for courses within a program, weighted by the unit value of each of those courses.". The numerical value assigned to each final grade to calculate your GPA for a program are shown as the following:

Numerical value	7	6	5	4	3	1.5
Semester 2, 2007 onwards	HD	A	B	C	D	F

Note that for the purposes of calculating GPA's, all Failing grades, such as F - Fail, FNP - Fail (Did not participate), FNS - Fail (Did not sit), FNC - Fail (Did not complete) and FLW - Fail (Late withdrawal), have a numerical value of 1.5. Also, courses for which exemptions have been granted are not included in the calculation of a GPA, but courses transferred for credit are included.

Mathematically, the GPA is calculated as:

$$\text{GPA} = \frac{\sum(\textit{Unit value} \times \textit{Grade value})}{\sum \textit{Unit value}} \tag{1}$$

where $\sum$ is "summed over the relevant courses", and the calculation results in a number (GPA) between 0 and 7.

In this assignment, we assume all courses have the equal *Unit value* as 1.

Source from:

- *Wikipedia. Grading (education), accessed on 12 Feb 2017.*
- *University of Southern Queensland. Your grade point average (GPA), accessed on 12 Feb 2017.*
- *University of Southern Queensland. Grade Point Average (GPA) Calculation for Medals and for With Distinction Procedure, accessed on 12 Feb 2017.*
- *University of Southern Queensland. Results legend and glossary, accessed on 12 Feb 2017.*

Your Task

USQ is a university with a considerable number and proportion of international students. Some of the international students are not familiar with the grading system in USQ and sometimes confused with how their Grade Point Average (GPA) is calculated. Aiming at helping these students, the USQ Student Guild decided to develop a program as students' personal GPA calculator. In this assignment, you'll give the USQ Student Guild a hand by designing and implementing the personal GPA calculator program that allows students to input the completed courses and awarded grades and then calculate the GPA accordingly.

Functional Requirements

The program should be implemented in *JavaScript* and running on *Firefox*, a web browser independent to operating systems. The USQ Student Guild has specified the following requirements for the functionality of the program:

1. The program should be running without errors throughout two Phases: *Information Gathering* and *Information Presenting*.
2. *Information Gathering* is to gather the information such as course codes and awarded grades, for calculation of GPA;
3. The program should first confirm with the student (user) for willingness of entering a new course before proceeding to gather information of the course code and grade value for calculation.
4. When receiving a new entry for a course, the program should first prompt and ask the user to enter the course code. If the user enters nothing or an invalid course code, the program should alert an error message on screen and then prompt the user to re-enter. The process should iterate until a valid course code is entered.
5. If the entered course code is valid, the program should then prompt the user to input the grade value for the course. Again, if nothing or an invalid value is entered, the program should display an error message then iterate until receive a valid grade value.
6. After valid input of course code and grade value, the program should loop back to seek user confirmation for either proceeding to add one more course or moving to the *Information Presenting* phase to calculate and display the results;
7. If the user confirms no more courses to enter, the *Information Gathering* phase is completed and the program then moves to *Information Presenting*.
8. In the *Information Presenting* phase, the program prints on the web page a table containing all entered courses, including information such as course code and awarded grade value.
9. To make the GPA calculator user-friendly, the USQ Student Guild also expects the program to display some statistic information:
   - The number of courses that the user has completed and entered;
   - The course(s) with the highest grade value;
   - The course(s) with the lowest grade value;
   - The Grade Point Average (GPA).

Respectively, Figure 1 and 2 illustrate the dataflow in *Information Gathering* and a sample result presented to the web browser in *Information Presenting*.

Implementation

Task 1 - A Validation Plan for Course Codes

You need to implement a validation plan to get a valid input from the user for course code. A valid course code needs to satisfy ALL the following criteria:

- The length of course code is seven;
  - The first three characters are alphabetic letters in uppercases; •
    The last four characters are numbers in range of 1000 – 9999;
- No space nor symbols should be included in the course code.

You can refer to the "Timetable Finder" page (https://www.usq.edu.au/timetables/Sem22018twmba/) on USQ website for sample course codes in practice.

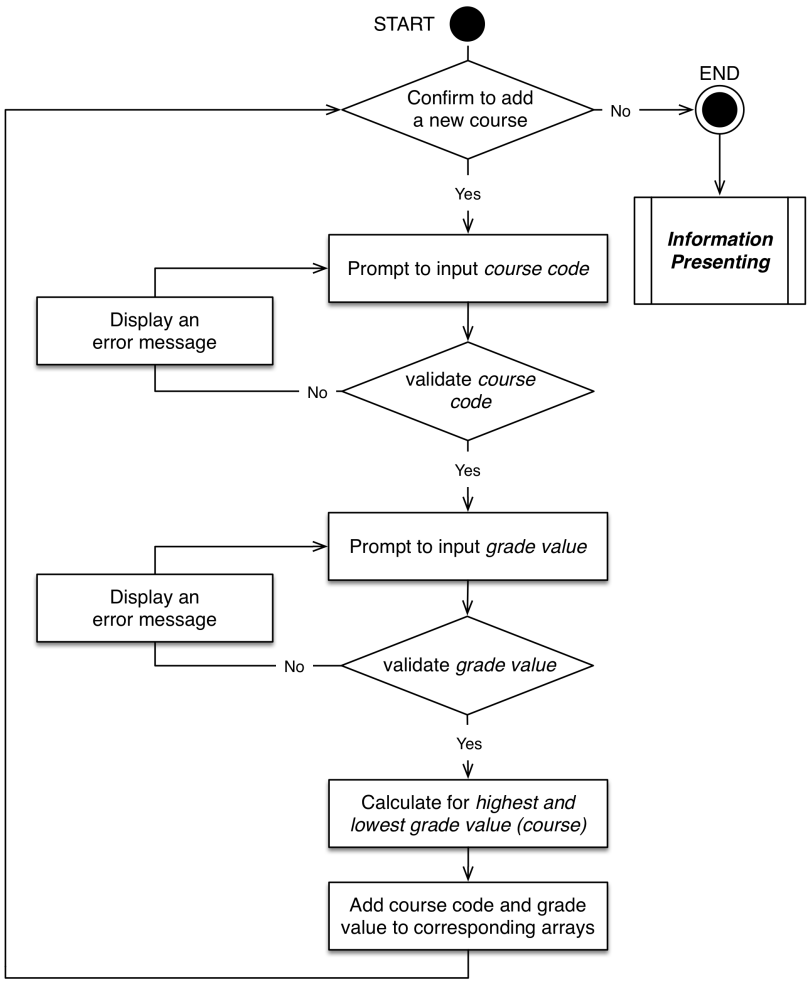

Figure 1: Dataflow in Information Gathering

Course Code	Grade
CSC1401	7
CSC1402	5
CSC1403	1.5
MAT1101	6
STA2300	5

Statistics

- There are 5 courses on the list;
- Course with the highest grade value: CSC1401;
- Course with the lowest grade value: CSC1403;
- Grade Point Average (GPA): 4.9

Figure 2: Sample Output of the Program

Task 2 - A Validation Plan for Grade Values

You need to implement another validation plan to get a valid input from the user for grade value. A valid grade value needs to satisfy ALL the following criteria:

- it is a number;
- it is an integer value between 3 and 7, or 1.5, a floating point value.

Task 3 - An Iteration Plan for Information Gathering Phase

You need to design an iteration plan to implement the *Information Gathering Phase*. Refer to Functional Requirements and Fig. 1 for the detail of data flow in iteration. Clearly, this task should incorporate the works in Task 1 and 2.

Task 4 - A Maximum Plan to Find the Course with the Highest Grade

Your program needs to find the course completed with the highest academic grade. If you have multiple courses with the same highest academic grade, you need to find and list only one (anyone) of them for "Course with the highest grade".

***Optional Challenge with no extra marks gained**: If you have multiple courses with the same highest academic grade, list them all together for "Course with the highest grade".*

Task 5 - A Minimum Plan to Find the Course with the Lowest Grade

Your program needs to find the course completed with the lowest academic grade. If you have multiple courses with the same lowest academic grade, you need to find and list only one (anyone) of them for "Course with the lowest grade".

Optional Challenge with no extra marks gained*: If you have multiple courses with the same lowest academic grade, list them all together for "Course with the lowest hrade".*

Task 6 - Sum, Count, Average Plans to Calculate the Grade Point Average

Your program needs to be able to calculate the grade point average for all entered courses. The task should be completed following Eq. 1 on Page 2 with a fixed parameter $Unit\ value = 1$, including

- a sum plan following $\sum(Unit\ value \times Grade\ value)$ to accumulate the grades of all courses;
- a count plan following $\sum Unit\ value$ to count the number of courses being entered;
- an average plan to calculate the grade point average;
- handling the "Division by Zero" exception when calculating average.

Task 7 - Presenting the Detailed Course Information

Print to a table the detailed information of entered courses including the course codes and grade values.

Task 8 - Presenting the Statistics

Print to an unordered list the statistic information of

- how many courses having been entered;
- the course with the highest grade value;
- the course with the lowest grade value;
- the Grade Point Average (GPA) with the precision of only one digit after decimal point (round up if the second digit is equal to or above 0.05).

The table and the list should be formatted like the screenshot in Figure 2.

Task 9 - Duplicate Course Detection (optional challenge with no extra marks gained)

The USQ Student Guild will appreciate it if an extra feature can be delivered – to detect duplicate course entries. If a course code has already been entered, the system should detect it and then ask for user confirmation for updating the corresponding grade value or not. If the user gives a positive confirmation, the system will proceed to prompt for grade value and then replace the stored grade by the newly entered value; otherwise, the program terminates the current course-adding process and iterates to ask user confirmation for adding a new course or not. Note that the user is not allowed to remove a course after entered.

Program Integration Test

You need to test the program for all functionality thoroughly before delivering the program to USQ Student Guild. The program should be running appropriately without any syntax or logic errors.

Non-Functional Requirements

- All code should appear in the script section in the head of the HTML document. Do not write any code in the HTML body. All functionality are delivered by JavaScript.
- In the script arrange your code as in the following order:
  (a) Constants;
  (b) Variables and objects (declared and initialised);
  (c) Other statements.
- Variable and constant identifiers should follow appropriate conventions.
- Code is indented appropriately and grouped in blocks according to the common tasks attempting to.
- Appropriate comments should be added to all blocks of code. Do not simply translate the syntax into English for comments, instead, describe the purpose of the code.

Submission

What You Need to Submit – Two Files

For a complete submission you need to submit two files as specified below. You can submit them individually or compress them and submit a common *.zip* (or *.rar*) file. The assignment submission system will accept only the files with extensions specified in this section.

1. ***Statement of Completeness*** in a file saved in *.doc* format with 200–300 of your own words describes:
   - **The state of your assignment**, such as, any known functionality that has not been implemented and delivered, etc. (It is expected that most people will implement all of the functionality required by this assignment.)
   - **Problem encountered**, such as, any technical problems that you encountered during the assignment work and how you dealt with them;
   - **Reflection**, such as, any lessons learnt in doing the assignment and suggestions to future programming study or work.
2. ***The program*** in a file saved with an *.html* extension contains the source code implemented following the functional and non-functional requirements.

Late Submission and Extension Request

Please refer to *USQ Policy Library - Assessment Procedure* for information on the late submission policy and *USQ Policy Library - Assessment of Compassionate and Compelling Circumstances Procedure* for considerable special circumstances in extension request.

The Extension Request Form is available on the course's StudyDesk. Should you need to request an extension please fill the form and email it to the Course Examiner with supportive documents (e.g., medical certificate or endorsement letter from supervisor in workplace) prior to the due date . **Please note that any requests without supportive documents will be rejected directly**.

Suggested Strategy

Plan to complete the assignment on time. Do not write all of the code in one sitting and expect that everything will be working smoothly like a magic.

First step Read assignment specification carefully; clarify anything unclear by putting a post on the assignment forum; think about how to do it, how to test it, devise high-level algorithms for each independent part of the assignment. Begin to write program (with comments), in incremental stages. Seek help on the assignment forum if needed.

Second step Re-read the specification carefully. Try to implement the function in one task a time, test it and make sure it works as expected before move to next task. Bring up any problems to the assignment forum if necessary. Integrate all functions and finish initial coding..

Third step Fully test the program; have another review on the source code; re-read the specification (especially marking criteria) to make sure you have done exactly what is required. Complete the "Statement of Completeness".

Marking Criteria

The assignment will be marked out of 24 and scaled down to a grade out of 12. Table 1 presents the marking criteria. If all criteria are satisfied you will receive 24 marks. If not all criteria are met, part marks may be given. Check your own submission against these criteria before you submit it.

Table 1: Marking Criteria

ID	**REQUIREMENTS**	MARK
	Statement of Completeness	
1	The statement is in appropriate length of 200-300 of student's own words	1
2	The "State of assignment" reflects the true state of completeness	1
3	The "Problems encountered" discusses problems, difficulties and their dealing strategies	1
4	The "Reflection" discusses learnt lessons and reasonable suggestions	1
	Subtotal	*4*
	Functional Requirements	
5	The program is running without any errors	1
6	Task 1 (a) - User input for course code is obtained appropriately	1
7	Task 1 (b) - The validation plan validates course code input as required	1
8	Task 2 (a) - User input for grade value is obtained and parsed to appropriate type if necessary	1
9	Task 2 (b) - The validation plan validates grade value input as required	1
10	Task 3 (a) - Information Gathering Phase in iteration takes multiple courses	1
11	Task 3 (b) - User confirmation is obtained using correct function and used as the sentinel for iteration plan	1
12	Task 4 - The maximum plan finds the course with the highest grade value	1
13	Task 5 - The minimum plan finds the course with the lowest grade value	1
14	Task 6 (a) - The count plan calculates the number of entered courses correctly	1
15	Task 6 (b) - The sum plan calculates the total grade values correctly	1
16	Task 6 (c) - The average plan calculate the GPA correctly	1
17	Task 6 (d) - The GPA is displayed at right precision level	1
18	Task 6 (e) - "Division by Zero" exception is handled appropriately	1
19	Task 7 - The entered courses and corresponding grade values are presented correctly in table form	1
20	Task 8 - The statistics are presented completely in an unordered list	1
	Subtotal	*16*
	Non-functional Requirements	
21	The program is implemented in JavaScript. No code goes outside of the script section	1
22	Identifiers of variables and constants are following professional conventions	1
23	Constants and variables are used in calculation and expression instead of explicit values	1
24	At least five comments are added to describe the purpose of blocks of code	1
	Subtotal	*4*
	TOTAL	**24**