ICS2O Summative

You are required to design and create a program of your choice that will provide a learning outcome/positive message during/at the completion of the program. The positive message should be something explored during our previous unit: Computers and Society.

Prior to beginning your project you must submit a typed proposal and a flow chart of the final product

You will be evaluated on degree of difficulty, code efficiency and organization, project execution and documentation.

Remember this is your final project and it is a representation of what you have learned. Start small, get it working and then add more.

Step 1: Project Proposal, Flow chart

- Brainstorm ideas
- When you have selected a final topic obtain approval from the teacher for that project
- Submit a project proposal that includes project description, minimum outcomes, advanced outcomes and timelines
- Include details of plan (written steps, storyboards, flowcharts, pseudo code, etc)

Step 2: Programming

- You should be working each day to advance your code/program
- You should take 10 mins at the end of each class to make a daily report of your progress. The reports could include:
  - A daily explanation of what you accomplished during the class
  - An evaluation of your project to date
  - Explanation of problems/expectations/changes encountered

Step 3: Testing and Maintaining

- In House test – try to locate errors and problems in your hardware/software and document fixes
- Field Tests (3 separate) – 1 classmate and 2 outside sources should test your program. Document the issues encountered, and the fixes you made
- Report on each set of tests

Step 4: Final Submission (Due Thursday, June 20th 2019 before 12:00pm)

- A User Manual or User Guide (External documentation)
- Project Proposal
- Optional: Daily Reports
- Testing Reports
- Final project complete and documented (internal documentation)

Note: Reports/code are/is due even if you are away! If away, you will submit them via google drive. Share them with me: nathan.soini@gapps.yrdsb.ca

In addition to marks, during the project, each day I will be using the following criteria to evaluate your learning skills:

- Time management
- Use and understanding of content previously taught
- Self learning (review/research stuff!)
- Reliance on teacher and others
- Daily reports
- Classroom etiquette

In order to assist you, you may use the six steps of the design process:

Analysis:

- Define the problem
- State the requirements and expectations
- Create a reasonable timeline for your projected progress

Design:

- Explain how the program will work
- Provide step by step details of your plan including diagrams (storyboards, flow charts, pseudo code, etc)

Programming:

- Create the program making sure to incorporate the concepts taught in class (procedures, arrays, loops, etc)
- Note; you may need to do your own research on topics not yet covered in class however you can create a great project using all that has been taught.

Documenting:

- Create both internal AND external documentation
- Provide extensive detail explaining the use of your program
- All code should be organized using comment separators
- All code should be explained using comments

Testing:

- Your program should be error proof
- Create a report of the errors/problems encountered and how you modified your program to accommodate to them

Maintaining:

- Look back at your initial plan. Did you complete the expectations?
- Have others run your program and make appropriate changes based on their recommendations

Name: **ICS2O Course Culminating Rubric** Date:

	Level 4	Level 3	Level 2	Level 1	Level R
Planning Program was planned out well. All required checkpoints were met as expected. Program was developed with little to no help.	Student clearly expressed their goals, followed a plan closely, and independently resolved issues along the way with minimal assistance. All deadlines were met.	Student had a fairly clear picture of goals but struggled to implement their plan. Most deadlines were met.	Student brain stormed but struggled to implement their plan without considerable teacher guidance. Only some deadlines were met.	Student spent little time brainstorming and was often unclear on how to continue without constant teacher assistance. Most deadlines weren't met.	No evidence that planning was undertaken.
Code Organization Clear internal documentation used including header, variable details, purpose of loops and conditional statements. Organized into appropriate sections with comment separators. External documentation describes the functions of the program and user input expectations.	Code is easy to follow with descriptive variables, extensive commenting and well organized. External documentation provides extensive details on usage and requirements of program.	Good degree of commenting and organization used to explain essential portions of the program. External documentation explains usage of program clearly.	Some commenting used to explain some code sections. Code is understandable but not well organized. External documentation explains some parts of the program.	Code is hard to follow with limited use of commenting to explain code. Slapped together with no organization. External documentation does not fully explain program.	No evidence that coding was completed.
Code Content Proper use of coding structure including input/output statements, variables, loops and if-then statements. Code is logical and well thought out.	Code is well thought out, and logical. All components are used appropriately.	Mostly well thought out logic, but not optimized. Most components used correctly.	Some thought put into program but doesn't flow properly. Many components are used incorrectly.	Code is full of programming errors with few components used properly.	No evidence that coding was completed.
Testing In-house (your own), 1 internal (classmate), and 2 external (friends/family) tests sessions completed. Reports submitted stating issues discovered and steps taken to resolve any problems.	All testing completed and fully documented.	Some testing completed with basic documentations.	Some testing complete with minimal documentation.	No testing done beyond typical daily error resolution.	No evidence that testing was done.
Code Functionality Program compiles and executes as expected. The interface is easy to follow and user friendly. The final product functions as expected and as according to the initial plan.	No flaws, functions as expected, and very user friendly.	Some minor flaws, but still quite functional and easy to use.	Some major flaws, but code still executes and some parts work.	Major flaws, code won't execute, but is present.	No evidence that coding was completed.

Comments: