405701 Programming 1 / 735318 Programming for Engineering Applications

Week 11, Homework 09	Weight: 6%	Due: Monday week 14, 09:00am (via `sync`)

Pre-homework Preparation:

- **Labs: Weeks 01, 02, 03, 04, 05, 06, 07, 08, 09, 10, 11**
- **Lectures: Weeks 01, 02, 03, 04, 05, 06, 07, 08, 09, 10, 11**

Homework Activities:

Obtain the homework files using **`sync`**. For each question's source file, be sure to fill in the file comment header accurately!

Authenticity:
Remember, it is unacceptable to hand in any code which has previously been submitted for assessment (for any paper, including Programming 1), and all work submitted must be unique and your own; do not collaborate with any other party on this assignment!

Homework 09: Project/Assignment

Individually develop your own unique C program. You have the freedom to choose what your program will do; it may automate some task, be a tool to be used in a business or engineering scenario, or solve an interesting problem, it could even be your own idea for a computer game! You choose!

You must utilise the following C language features in your program:

- Sequence, Selection, Repetition, Functions
- Variable Types, Arrays, Structs
- Dynamic Memory
- Console Input/Output , File Input/Output
- Multiple **`.c`** and **`.h`** files

Ensure your idea for the program has enough scope to satisfy each of the feature areas listed above. You may utilise the P1 Turtle World, Robot World, or Image World in your program, or a combination of; but you are not required to use these. Pick what will benefit your program.

Design Phase

In the **`design.txt`** file, complete the following descriptions using English:

- Overview: A short paragraph about the program's purpose.
- Features: A list of the core features your program will provide to the user.
- Algorithms: An overview description of any important algorithms your program will need to implement to realise the features.
- Interface: A description of the proposed user interface; how will the user interact with your program?

Ensure the document is proofread, and it is free of spelling and grammar errors.

Implementation Phase

After deciding on what your program will do, and writing the **`design.txt`** document, begin programming.

As you program, start small… add a feature, compile, run and test the feature. Incrementally build your program with short iterations. Start simple, and build up to more complex scenarios. You will need to keep a development log as you progress.

Development Log

Keep track of your progress for this assignment in a development log. This must be recorded in the electronic document **`log.txt`** file. Each time work is undertaken on Homework 9, create a new entry in the Development Log. Each entry must detail the Date and Time work was undertaken, the duration of the work, what tasks were undertaken, and any issues that arose.

Ensure the document is proofread, and it is free of spelling and grammar errors.

Testing Phase

During development, regularly test your program's functionality. Identify and correct any bugs that are in your program.

Document bugs encountered in testing the program in the **`bugs.txt`** file. Each bug entry must include:

- Unique ID: (for example: BUG-1.)
- Bug Description: (for example: When the user is prompted to enter their age, entering a negative value causes the program to crash with a segmentation fault.)
- Bug Status: (for example: Fixed/In-Progress/Not-Fixed.)

Homework Submission:

Run the **`sync`** command to submit your completed homework.

Marking Criteria:

Criteria:	Weight:	A Grade Range: 100 ≥ x ≥ 80%	B Grade Range: 80 > x ≥ 65%	C Grade Range: 65 > x ≥ 50%	D Grade Range: 50 > x ≥ 0%
Design -design.txt -Details -Spelling -Grammar	**10%**	`design.txt` has a clear overview of the program's purpose. Key features are listed. Required algorithms are documented. The user interface is described. Program matches `design.txt` requirements. `design.txt` is free of spelling and grammer issues.	`design.txt` has an overview of the program's purpose. Key features are listed. Required algorithms are documented. The user interface is described.	`design.txt` has an overview of the program's purpose. Key features are listed.	`design.txt` is not present or is empty.
Bug Tracking: -log.txt -Accuracy	**10%**	`bugs.txt` is comprehensive. Multiple entries have been detailed, and status have accurately been recorded and updated during development.	Multiple entries have been detailed in `bugs.txt`, and status have accurately been recorded.	Some bugs detailed in the `bugs.txt`.	`bugs.txt` is not present or is empty.
Development Log: -Spelling -Grammar -Accurate -Descriptive -Honest	**10%**	Comprehensive development `log.txt`. `log.txt` is free of spelling and grammer issues.	Development log `log.txt` file is present, and contains some development entries.	Development log `log.txt` file is present, and contains some development entries.	Development log `log.txt` file is not present or does not contain any useful detail.
C Programming: -Sequence -Selection -Repetition -Functions	**15%**	Functions are modular and reusable. Sequence, selection, and repetition are all demonstrated in a variety of unique functions in the program.	Sequence, selection, and repetition are all demonstrated in a variety of unique functions in the program.	Unique functions are defined and called.	Program does not define any functions.
C Programming: -Variables -Arrays -Structs -File I/O	**15%**	Variables and arrays are used meaningfully. Multiple arrays are used and have purpose in the program. Meaningful structs are created and used. File input and output is meaningfully used.	Variables and an array is used.	Variables are used.	No variables used.
C Programming: -Dynamic Memory	**10%**	Malloc and free are used meaningfully, for a dynamic purpose in the program.	Malloc and free are used.	Malloc is used.	No malloc or free used.
Runtime Execution: -Interactive -Meaningful -Non-trivial -Bug Free	**15%**	The program is interactive, meaningful and has a non-trivial purpose that will engage the user. The program is bug free.	The program is interactive, meaningful and has a non-trivial purpose that will engage the user.	The program is interactive, and has some purpose.	There is no program.
Code Quality: -Whitespace -Naming -Reuse -Modularity	**15%**	Whitespace is perfect, code is indented flawlessly. Functions and variables are well named, their purpose is clear from their given name. Reusable code is modularised into functions and multiple source files, with associated header files, are present.	Whitespace is almost perfect, only minor whitespace problems present. Functions and variables are well named, their purpose is clear from their given name.	Code is readable. Most variable and function naming is good, however there may be room for improvement.	Code is difficult to read. Good programming practice has not been followed.