

SEEM3460 Tutorial

Assignment 2

2025.10.27

Pls ensure the followings:

- ❑ **Use CUHK network if you are using your own computers**

Otherwise you won't be able to connect to the servers

- ❑ **Connect to our remote servers**

linux03.se.cuhk.edu.hk

linux04.se.cuhk.edu.hk

Description: Assignment 2

Question 1

- (a) Implement the **multi-module C program** design for the calculator project
- (b) Modify the design with an **additional module for subtraction**.

Question 2

Consider the multi-module design as described in the Lecture notes on “Multi-Module Design Example: Matrix Operations in C”, and assume that the parts for Gauss elimination and Linear regression **are not yet completed**.

Description: Assignment 2

For both Q1 and Q2:

- You should verify the correct operations of functions within **both Question 1 and Question 2** with appropriate data.
- Show **at least 3 successful running examples** for each question. Provide the screenshots that include both input and output. You can also report functions' processing results for **wrong input data** to show robustness.

Techniques for Assignment 2

Multi-Module Design

Modules: Typically consist of a pair—a header file (.h) for declarations and a source file (.c) for definitions and implementations, respectively.

Header Files: Declared functions, variables, structures, enums, and macros that can be shared across modules. They use include guards (e.g., `#ifndef HEADER_NAME_H`) to prevent multiple inclusions.

Source Files: Contain the actual code implementations. Functions or variables defined here can be made accessible to other modules via the header.

Scope Control: Use `extern` to declare variables or functions defined in another module. Use `static` to restrict functions or variables to the current module (file scope only).

You can refer to Lecture note: <Multi-Module Design in C Programming> & <Multi-Module Design Example: Matrix Operations in C>

Techniques for Assignment 2

Makefile

Target: What to build (e.g., executable or object file).

Prerequisites: Files needed to build the target (e.g., source files or headers).

Recipe: Commands to execute (indented with a tab, not spaces).

Variables: Defined like `CC = gcc` for reusability.

Phony Targets: Special targets like `clean` that aren't files (declared with `.PHONY: clean`).

Pattern Rules: For generic handling, e.g., `%.o: %.c` to compile any `.c` to `.o`.

Common variables:

`CC`: Compiler (e.g., `gcc`).

`CFLAGS`: Compiler flags (e.g., `-Wall -O2` for warnings and optimization).

`LDFLAGS`: Linker flags (e.g., `-lm` for math library).

You can refer to Lecture note: <Makefiles for C Projects>

Notice

Do not plagiarize !!

If a student plagiarizes, there will be a heavy punishment. The definition of plagiarism includes copying all or part of programming assignments from other's work or the Web. **The penalty will apply to both the one who copies the work and the one whose work has been copied.**

Submission

For Question 1:

6 files to submit in total.

main.c: Entry point to test everything.

add.h + add.c: Conduct add operation.

subtraction.h + subtraction.c: Conduct subtraction operation.

makefile: For compiling source files, linking object files, and handling dependencies

Submission

For Question 2:

6 files to submit in total.

main.c: Entry point to test everything.

matrix_utils.h + matrix_utils.c: Shared helpers (create, free, transpose, print matrices).

matrix_mult.h + matrix_mult.c: Matrix multiplication

makefile: For compiling source files, linking object files, and handling dependencies

Moreover:

You should provide your **testing results** for both Q1 and Q2 in one PDF file.

Submission

PDF file: <Student ID>-assign2-main.pdf;

Programming files for Q1 and Q2: All the *.c and *.h, and the makefile files.

Put all files in a folder and zip this folder: Name it as <Student ID>-

<Student Name>-assign2.zip. Submit **only** this zip file.