C Programming Major Project Guidelines

University of Petroleum and Energy Studies

Course Code: CSEG1032
Instructor: Dr. Srinivasan Ramachandran

1. Introduction

This document describes the complete set of rules, expectations, rubric criteria, and auto-evaluation mechanisms for the **C Programming Major Project**. The goal is to evaluate your ability to design, implement, and document a fully working C program that integrates concepts from all six units of the course.

The evaluation process is based entirely on:

- A standardized GitHub repository structure,
- A single text file submitted to the LMS,
- Automated scoring via a Python-based evaluator,
- Transparent and strictly enforced rules for originality and repository management.

An official example repository is provided for reference:

`https://github.com/aalavandhaann/Major-Project-Example.git`

You must follow the same structure, but you must not copy the code.

2. Team Size

- Maximum of **two students per team**.
- Each student must maintain their **own individual GitHub repository**.
- Both repositories must contain **identical content and commit history**.
- A shared repository for two students will result in **zero marks** for both.

3. Submission Requirement (Mandatory)

Each student will upload a single text file named:

`github_link.txt`

This file must contain **only one line**:

`https://github.com/<username>/<repository-name>.git`

Rules:

- The repository must be **public** until evaluation is completed.
- The report must be placed inside the GitHub repository (not uploaded to LMS).
- The repository link cannot be changed after submission.

4. Mandatory GitHub Repository Structure

Your repository must exactly follow the structure of the official example project.

```
/
|-- src/               (all .c files)
|-- include/           (all .h files)
|-- docs/              (project report, flowcharts, assets)
|-- assets/            (optional: screenshots, diagrams)
|-- README.md          (project description and instructions)
|__ sample_input.txt   (optional test inputs)
```

Important: Folder names are case-sensitive and must match exactly: src, include, docs. Any mismatch will result in automatic deduction during evaluation.

5. Project Report Structure (Inside /docs)

Your full project documentation must be included inside:

`/docs/ProjectReport.pdf`

The report must include:

1. Title Page
2. Abstract
3. Problem Definition
4. System Design (Flowcharts, Algorithms)
5. Implementation Details (with snippets)
6. Testing & Results
7. Conclusion & Future Work
8. References
9. Appendix (optional)

6. Project Ideas and Originality Clause

You may choose from the suggested problems or propose an original idea. However:

> Choosing an idea from the provided list caps your maximum originality score at **8/10**. Proposing an original topic allows you to earn the full **10/10**.

Suggested ideas include (but are not limited to):

- Student Record Management System
- Quiz Application
- Mini Library System
- Matrix Calculator with Dynamic Memory

7. Updated Rubrics (100 Marks)

The final project score is out of **100 marks**. Viva has been removed.

Criterion	Marks	Description
Problem Definition & Design	15	Clarity of objectives, logic, and design.
Implementation & Coding Style	20	Modularity, readability, comments, correctness.
Documentation Quality	15	Completeness of the report inside /docs.
GitHub Usage & Collaboration	20	Structure, commit activity, and correctness of repo.
Originality / Creativity	10	Novelty of idea, low similarity scores.
Execution & Output Validity	20	Correctness of output and runtime behavior.
Total	**100**	

8. Auto-Evaluation Logic (Python Script)

Your project will be evaluated using an automated Python grader. The following aspects will be analyzed:

8.1 Folder Structure

- Checks for required folders: `/src`, `/include`, `/docs`
- Any deviation results in point deduction

8.2 Code Quality and Implementation

Checks include:

- Successful compilation via GCC
- Number of user-defined functions
- Comment density
- No unused or uninitialized variables
- Clean modular separation of .c and .h files

8.3 Documentation Quality

The script scans the report for required sections:

- Abstract
- Problem Definition

- Algorithm/Flowchart
- Implementation
- Testing
- Conclusion

8.4 GitHub Activity Verification

Using GitHub API:

- Verifies number of commits
- Checks commit timestamps (prevents last-minute spamming)
- Ensures both team members contributed equally

8.5 Originality Check

Two comparisons:

- Against the example repository
- Against all other student submissions

Similarity over 40% results in severe deductions.

8.6 Execution & Output Testing

The script runs:

```
./main < sample_input.txt
```

Checks include:

- Correct output format
- Graceful handling of invalid inputs
- Absence of crashes or segmentation faults

9. Academic Integrity

- AI-generated or copied code is strictly prohibited.
- High similarity across unrelated submissions results in 0 marks.
- The example repository is for structure reference only; copying its logic or code will be detected.

— End of Guidelines Document —