Multi-Paradigm Problem Solving built with C/OCaml/Python

Samrat Baral

Date: April 25, 2025

- **Term:** Spring 2025 Second Bi-term
- **Class:** Advanced Programming Languages -MSCS-632-B01
- **Instructor:** Dr. Dax Bradley
- **University of the Cumberlands**

The GitHub repository, https://github.com/baralsamrat/MSCS632_Assignment_7

REPORT : - 📒 **PDF**

Screenshots

Capture-1.PNG 1

Overview 📊 Mean, Median, and Mode Calculator

This project implements a simple statistics calculator to compute the **mean**, **median**, and **mode** from a list of integers. It is implemented in three different paradigms:

Language	Paradigm	File
C	Procedural	`math.c`
OCaml	Functional	`math.ml`
Python	Object-Oriented	`math.py`

✅ Output

```
====================================
Running C Program (Procedural Approach)
====================================
Data: 5 3 8 3 9 1
Mean: 4.83
Median: 4.00
Mode: 3

====================================
Running OCaml Program (Functional Approach)
====================================
Data: 5 3 8 3 9 1
Mean: 4.83
```

```
Median: 4.00
Mode: 3

====================================
Running Python Program (OOP Approach)
====================================
Data: [5, 3, 8, 3, 9, 1]
Mean: 4.83
Median: 4.00
Mode: 3
```

🚀 How to Run

Prerequisites:
Install the following:

Software	Install Command
`gcc`	`sudo apt install gcc`
`ocaml`	`sudo apt install ocaml`
`python3`	`sudo apt install python3`

Steps:

```
git clone https://github.com/baralsamrat/MSCS632_Assignment_7
cd MSCS632_Assignment_7/src
chmod +x run.sh
./run.sh
```

The script compiles and runs C, OCaml, and Python programs automatically.

🗂 Project Structure

```
.
├── math.c      # C implementation (Procedural)
├── math.ml     # OCaml implementation (Functional)
├── math.py     # Python implementation (OOP)
├── run.sh      # Script to run all programs
└── README.md   # Documentation
```

🔎 Brief Report: Paradigm Comparison

- **Procedural (C)**: Manual memory management and array operations; prone to bugs if not handled carefully but fast and efficient.
- **Functional (OCaml)**: Immutability and pure functions make code concise and safer, though managing state inside folds can be tricky.
- **Object-Oriented (Python)**: Encapsulation makes the code clean, modular, and easy to extend using classes and built-in libraries.

Aspect	C (Procedural)	OCaml (Functional)	Python (OOP)
Programming Style	Step-by-step instructions	Function chaining, recursion	Class-based encapsulation
Data Handling	Manual arrays	Immutable lists	Lists, dynamic typing
Development Ease	Moderate, careful coding needed	Moderate, requires functional thinking	Easy, intuitive
Key Challenges	Memory management, sorting logic	State tracking inside folds	Keeping methods modular
Error Risk	High	Low	Low

✨ Conclusion

Each paradigm brings its strengths:

- **C** is ideal for fine control and performance.
- **OCaml** ensures correctness through pure functional logic.
- **Python** excels in ease of development and scalability.
