Programming Languages Practical Assignment

Total Marks: 30
Language: Any programming language of your choice

Assignment Overview

This practical assignment explores programming language concepts through hands-on implementation and analysis. You will create a simple calculator program and analyze your chosen language's features.

Part A: Calculator Implementation (15 marks)

Task 1: Basic Calculator (10 marks)

Create a simple calculator program that can perform the following operations:

Required Features:

1. **Basic Operations:** Addition, Subtraction, Multiplication, Division
2. **Input Handling:** Accept two numbers and an operator from user
3. **Error Handling:** Handle division by zero and invalid inputs
4. **Menu System:** Allow users to perform multiple calculations
5. **Exit Option:** Provide a way to quit the program

Sample Program Flow:

=== Simple Calculator ===

1. Addition (+)

2. Subtraction (-)

3. Multiplication (*)

4. Division (/)

5. Exit

Choose operation (1-5): 1

Enter first number: 15

Enter second number: 25

Result: 15 + 25 = 40

Continue? (y/n): y

Choose operation (1-5): 4

Enter first number: 10

Enter second number: 0

Error: Division by zero is not allowed!

Continue? (y/n): n

Thank you for using the calculator!

Implementation Requirements:

- Use functions/methods for each operation
- Include input validation
- Provide clear user interface
- Handle errors gracefully

Task 2: Enhanced Features (5 marks)

Add TWO of the following features to your calculator:

Choose 2 from these options:

1. **Memory Functions:**
    - Store result in memory (M+)
    - Recall from memory (MR)
    - Clear memory (MC)

2. **Advanced Operations:**
    - Power (x^y)
    - Square root
    - Percentage calculations
3. **History Feature:**
    - Keep track of last 5 calculations
    - Display calculation history
    - Clear history option
4. **File Operations:**
    - Save calculations to a text file
    - Load and display previous calculations

Example with Memory Functions:

Choose operation (1-6): 1

Enter first number: 10

Enter second number: 5

Result: 10 + 5 = 15

Save to memory? (y/n): y

Memory saved: 15

Choose operation (1-6): 2

Enter first number: M (recall from memory)

Using memory value: 15

Enter second number: 3

Result: 15 - 3 = 12

Part B: Language Analysis Report (10 marks)

Task 3: Language Feature Analysis (10 marks)

Write a report analyzing your chosen programming language based on the calculator implementation.

Report Structure: Create a document (Word/PDF/Markdown) covering the following sections:

1. Language Choice and Justification (2 marks)

- Which programming language did you choose and why?
- What influenced your decision?
- Is this language suitable for this type of application?

2. Language Evaluation Criteria Analysis (4 marks)

Based on your implementation experience, evaluate your chosen language on:

Readability (1 mark):

- How easy is your code to read and understand?
- Provide a code example that demonstrates good/poor readability
- What language features help or hinder readability?

Writability (1 mark):

- How easy was it to write the calculator in this language?
- What features made coding easier or harder?
- Did you need many lines of code or could you be concise?

Reliability (1 mark):

- How does the language help prevent errors?
- Does it have strong type checking?
- How did you handle runtime errors?

Cost (1 mark):

- How much time did you spend learning language-specific features?
- Is the development environment free and accessible?
- Would this language be cost-effective for a business project?

3. Implementation Challenges and Solutions (2 marks)

- What was the most challenging part of the implementation?
- How did your language's features help or hinder you?
- What would you do differently if you used another language?

4. Language Comparison (2 marks)

- Compare your chosen language with ONE other language you know
- Which would be better for this calculator project and why?
- Give specific examples of how the implementation might differ

---

Part C: Code Quality and Documentation (5 marks)

Task 4: Professional Code Standards (5 marks)

Code Quality Requirements:

- **Clear Naming:** Use meaningful variable and function names
- **Comments:** Add comments explaining complex logic
- **Structure:** Organize code into logical sections/functions
- **Formatting:** Consistent indentation and spacing
- **Error Handling:** Proper exception/error management

Documentation Requirements:

- **README File:** Include instructions on how to run your program
- **Code Comments:** Explain what each major function does
- **User Guide:** Brief explanation of how to use the calculator

Example Code Quality:

python

```
# Good example (Python)
def divide_numbers(dividend, divisor):
    """
```

```
    Performs division operation with error handling
    Args: dividend (float), divisor (float)
    Returns: result (float) or error message (string)
    """
    try:
        if divisor == 0:
            return "Error: Division by zero"
        return dividend / divisor
    except ValueError:
        return "Error: Invalid input"

# Poor example
def d(a,b):
    return a/b  # No error handling, unclear names
```

Submission Requirements

Files to Submit:

1. **Source Code:** Your calculator program file(s)
2. **Analysis Report:** Document file (Word/PDF/Markdown)
3. **README.txt:** Instructions for running your program
4. **Sample Output:** Screenshots or text file showing program execution

File Naming Convention:

- Calculator: calculator_[your_name].[extension]
- Report: language_analysis_[your_name].[extension]
- README: README_[your_name].txt