# SENG1005 – C PROGRAMMING

## ASSIGNMENT 3 - ARRAY AND POINTER TECHNIQUES WITH BITWISE OPERATIONS

## OVERVIEW

In this assignment, you will create a Cryptocurrency Wallet Manager in C programming. This program simulates a basic cryptocurrency wallet where users can manage transactions, analyze wallet data, and perform essential operations. The tasks are designed to enhance your understanding of dynamic memory management, bitwise operations, array manipulation, and error handling.

## GENERAL COURSE OBJECTIVES ADDRESSED IN THIS ASSIGNMENT

- Apply advanced C programming techniques and tools used in industry for software development.
- Analyze and diagnose common C programming errors using effective debugging techniques.
- Create efficient software solutions that leverage data structures, arrays, and pointers.
- Implement robust user input validation to enhance application security and prevent vulnerabilities.
- Use function pointers to create flexible and modular code.

## ACADEMIC INTEGRITY AND LATE PENALTIES

- Use of Generative AI
  - ➢ You are prohibited from using generative AI tools to create part or all of the solution to this assignment.
  - ➢ You are permitted to use generative AI tools for personal use to learn skills and gain knowledge related to the course content.
  - ➢ If you incorporate any techniques, patterns, or code into your assignment that have not been directly taught in this course, you must cite the source of that information explicitly. Failure to do so will be considered an academic integrity violation. This is especially important for patterns generated by AI when producing code that is not covered in our course material.
  - ➢ Please refer to the Generative AI Toolkit for Students (Home - Gen AI for Students - Library & Learning Services at Conestoga College).
- Link to Academic Integrity Information
- Link to Late Policy

## EVALUATION

- TBD

## PREPARATION

- Familiarize yourself with error handling and debugging techniques.
- Ensure familiarity with memory management techniques and error handling in C.
- Familiarize yourself with pointers and bitwise operations
- Review lectures and assignments from this week and previous weeks

## REQUIREMENTS

### MAIN PROGRAM

### Core Functionalities

#### *Adding Transactions*

- ➢ Allow the user to add transactions by inputting amounts.

- Use -1 as a sentinel value to stop adding transactions.
- Initialize each transaction's status flags (Processed and Refunded) to No (unset).

### *Displaying Transactions*

- Display the full list of transactions with:
  - Transaction amount.
  - Flags indicating if the transaction is processed or refunded.
- Format monetary values as $XX.XX.

### *Adding Transactions*

- Calculate and display the following:
  - Total transaction amount.
  - Average transaction value.
- Handle the case where there are no transactions gracefully.

### *Applying Transaction Fees*

- Deduct a user-specified percentage fee from each transaction.
- Update the transaction amounts to reflect the fee deduction.
- Display the updated amounts after applying the fee.

### *Finding the Highest Transaction*

- Identify and display the highest transaction amount.
- Handle the case where there are no transactions gracefully.

### *Swapping Transactions*

- Allow the user to specify two transaction indices to swap their values.
- Perform the swap using bitwise XOR operations.

### *Swapping Transactions*

- Provide a menu for managing flags with the following options:
  - Set a flag: Mark a transaction as processed.
  - Clear a flag: Unmark a transaction as processed.
  - Toggle a flag: Change the refunded status of a transaction.
  - Display all flags: Show the status of each transaction.
- Use bitwise operations (AND, OR, XOR) to manipulate flags.

## Program Constraints

### *Dynamic Memory*

- Use malloc and realloc for all dynamic arrays.
- Ensure proper memory deallocation with free before the program exits.

### *Bitwise Operations*

- Use bitwise operators (AND, OR, XOR, NOT) for:
  - Swapping transaction values.
  - Managing flags.

*Formatting*

- ➢ Format all monetary values to two decimal places.
- ➢ Display flags as Yes or No.

## Error Handling and Validation

- Ensure that the program handles common file errors, such as file not found, permission denied, or read/write errors, gracefully.
- Ensure all user inputs are validated.
- Use appropriate error messages to guide the user.

## Memory Management

- Ensure that all dynamically allocated memory is properly freed before the program exits.
- Dynamically allocate memory to store an array of transaction amounts (in USD).
- Ensure memory is expanded automatically as new transactions are added.
- Free all allocated memory when the program exits.

## COMMENTING AND INDENTATION:

- You must have a header comment similar to that found in the SET Coding Standards or the Course Notes. This requirement is the same as in Focused Assignment 2.
- You must have function comments for all functions except main().
- Indent bodies as previously discussed in lecture.
- Use the SET Coding Standards (found on eConestoga) that are relevant.
- The program must not use ‘goto’ (this requirement holds for all subsequent assignments).
- All variables must be declared within functions (i.e. it must not use global variables (covered later in the Scope and Style lecture)) except for constants.
- Appropriate programming style as discussed in lecture and in the course notes must be used.
- You must only use C language features that have been covered or mentioned in the course up to the assignment due date in completing this assignment.

## FILE NAMING REQUIREMENTS

- ➢ Assignment3.cpp and assignment3.h

## SUBMISSION REQUIREMENTS

- ➢ Follow the SET Submission Standards for naming conventions and file organization.
- ➢ Ensure the program compiles and runs without errors on the specified development environment.

## ADDITIONAL INFORMATION

- Use the provided lecture materials on error logging and test harness development as reference.

- Include additional error-handling mechanisms as needed.
- Be prepared to demonstrate and explain your solution if requested.