CNG140 C Programming Project 2

Date handed out: 14 August 2024, Wed
Date submission due: 22 August 2024, Thursday, 23:55, **Cyprus Time**

CNG140 Programming Assignment 4: Simple Banking System (150 points)

Purpose:

The main purpose of this programming assignment is to revise the topics we have covered so far, including arrays, pointers, functions, repetitive statements, conditional statements, and C programming fundamentals.

In this assignment, you are tasked with creating a simple banking system in C. The program will simulate basic banking operations such as account creation, deposits, withdrawals, and checking balances. The program must use arrays to store account information, pointers for passing data between functions, loops for repetitive tasks, and conditional statements for decision-making processes.

1. **Main Menu:**
   - The program should display a menu with the following options:
     - Create New Account
     - Deposit Money
     - Withdraw Money
     - Check Balance
     - Quit
   - Use a loop to continuously display the menu until the user decides to quit.
2. **Account Creation:**
   - When the user selects "Create New Account," the program should prompt for the user's name and initial deposit amount.
   - Each account should be assigned a unique account number, starting from 1001.
   - The account details (name, account number, balance) should be stored in an array of structures. Each structure should contain a string (char array) for the name, an integer for the account number, and a float for the balance.
   - Use pointers to pass the array to the function handling account creation.
3. **Deposit Money:**
   - The user should be able to deposit money into an existing account by entering the account number and the deposit amount.
   - The program should validate the account number using a loop and conditional statements.
   - If the account exists, update the balance using the pointer to the structure.
4. **Withdraw Money:**
   - The user should be able to withdraw money from an existing account by entering the account number and the withdrawal amount.
   - The program should validate the account number and ensure sufficient funds using conditional statements before processing the withdrawal.

    - Update the balance accordingly, making use of pointers.
5. **Check Balance:**
    - The user should be able to check the balance of any account by entering the account number.
    - Use a loop to find the account and display the current balance.
6. **Quit:**
    - The program should terminate gracefully when the user selects the "Quit" option.

Programming Requirements:

To implement this game, you will need to write **at least** the following functions, but if you need more functions, please add them.

Function	***Description***
main	Handles the main menu loop and calls the appropriate functions based on user input. Ensures that the array of accounts is managed properly.
displayMenu	Displays the main menu using repetitive statements and returns the user's choice.
createAccount	Handles the logic for creating a new account. It should update the array of accounts with the new account details using pointers and manage the assignment of unique account numbers.
depositMoney	Handles the logic for depositing money into an account. It should validate the account number, use loops to find the account and update the balance using pointers.
withdrawMoney	Handles the logic for withdrawing money from an account. It should check for sufficient funds using conditional statements and update the balance accordingly.
checkBalance	Displays the current balance of a specific account using a loop and conditional statements to locate the account.
getUserInput	Reads and validates user input for different operations, ensuring the program handles errors gracefully.

Sample Output:

```
Welcome to the Simple Banking System Designed by METU-NCC Student!

Menu:
1. Create New Account
2. Deposit Money
3. Withdraw Money
4. Check Balance
5. Quit
```

```
Enter your choice: 1

Enter your name: Mehmet
Enter initial deposit: 1000
Account created successfully!
 Account Holder Name is Mehmet
 Account number is 1001.

Menu:
1. Create New Account
2. Deposit Money
3. Withdraw Money
4. Check Balance
5. Quit

Enter your choice: 2

Enter account number: 1001
Enter deposit amount: 500
Deposit successful! New balance: 1500

Menu:
1. Create New Account
2. Deposit Money
3. Withdraw Money
4. Check Balance
5. Quit

Enter your choice: 3

Enter account number: 1001
Enter withdrawal amount: 300
Withdrawal successful! New balance: 1200

Menu:
1. Create New Account
2. Deposit Money
3. Withdraw Money
4. Check Balance
5. Quit

Enter your choice: 4

Enter account number: 1001
Current balance: 1200

Menu:
1. Create New Account
2. Deposit Money
3. Withdraw Money
4. Check Balance
5. Quit

Enter your choice: 5

Thank you for using our Simple Banking System
Goodbye!
```

Grading Policy:

If your code does not compile, you will automatically get zero. If your code compiles, you will then be graded based on the following criteria:

- **main**: Properly initializes the program, handles menu navigation using loops, and calls the correct functions with pointers to the data structures (20 points).
- **displayMenu**: Displays the menu correctly and uses loops to manage user input (10 points).
- **createAccount**: Correctly implements account creation using arrays and pointers, updates the account array, and assigns unique account numbers (15 points).
- **depositMoney**: Accurately handles deposits, including account number validation using loops and conditionals, and balance updates using pointers (15 points).
- **withdrawMoney**: Correctly implements withdrawals, ensuring sufficient funds using conditional statements, and accurately updates the balance using pointers (15 points).
- **checkBalance**: Properly displays the balance for a given account number using loops and conditionals (15 points).
- **getUserInput**: Correctly handles and validates user input throughout the program (10 points).
- **Use of Arrays**: Effective and correct use of arrays to store account information, with proper memory management (10 points).
- **Use of Pointers**: Proper implementation of pointers for passing data between functions and managing arrays (10 points).
- **Use of Repetitive Statements**: Appropriate use of loops for menu navigation, account management, and user input handling (10 points).
- **Use of Conditional Statements**: Correct and efficient use of conditional statements for validating inputs and making decisions in the program (10 points).
- **Code Quality**: Includes appropriate comments, meaningful variable names, and a clean, readable structure. Also, demonstrates reusability and extensibility (10 points).

Please make sure that you follow the restrictions for the assignment as follows:

- Strictly obey the input-output format. Do not print extra things.
- You are not allowed to use Global and Boolean variables.
- Add your name/surname and ID at the top of your code as comments, and name your source file “**Name-Surname-StudentID.c**”
- Only submit your C code.