CSPP Week - 5 Exam

Section - II | Time: 3 hours | Max Score: 25 pts

Digital Wallet Programming

Overview:
In this assignment, you will be developing a simple text-based version of a Digital Wallet using Python. The system will enable a user to manage money in their wallet, make payments, view transaction history, and add funds. You'll write multiple functions to accomplish these tasks.

Function 1: initializeWallet() - 3 pts
Write a function named initializeWallet() that doesn't take any arguments and initializes a wallet with a balance of 0. It returns a dictionary representing the wallet, which should include a balance and an empty transaction history list.

Function 2: displayBalance(wallet) - 3 pts
Write a function named displayBalance() that takes a wallet dictionary as its argument and prints the current balance. The function should return None.

Function 3: addFunds(wallet, amount) - 3 pts
Write a function named addFunds() that takes a wallet dictionary and an integer amount as its arguments. The function should add the amount to the wallet's balance and record the transaction in the transaction history. If the amount is zero or negative the function should return immediately without doing any transaction and prints “Amount can’t be negative” onto the screen. The function should return None.

Function 4: makePayment(wallet, amount) - 4 pts
Write a function named makePayment() that takes a wallet dictionary and an integer amount as its arguments. Ensure there are sufficient funds in the wallet. If so, deduct the amount from the balance and record the transaction. If not, print a warning. The function should return None.

Function 5: transactionHistory(wallet) - 4 pts
Write a function named transactionHistory() that takes a wallet dictionary as its argument. The function should display a list of all transactions made, both credits (additions) and debits (payments).

Function 6: useWallet() - 8 pts
Write a function named useWallet() that ties everything together. The following steps should be followed

Steps:

1. **Setup:** Welcome the user and initialize a new digital wallet using initializeWallet().
2. **Menu Loop:** Create a menu where the user can choose to:
    a. View balance
    b. Add funds
    c. Make a payment
    d. View transaction history
    e. Exit
3. **User Input:** Depending on the user's choice:
    a. Display the balance using displayBalance().
    b. Add funds using addFunds().
    c. Make a payment using makePayment().
    d. Display transaction history using transactionHistory().
4. Continue looping until the user chooses to exit.

Example Session:

1. The user starts the interface and is welcomed with a menu.
2. The user chooses to view the balance. The balance is displayed as $0.
3. The user decides to add $50 to the wallet.
4. After adding, the user views the balance again, confirming it's now $50.
5. The user makes a payment of $20.
6. The user then checks the transaction history, seeing a credit of $50 and a debit of $20.
7. The user exits the interface.