CMPG 121 Test 1_Instructions

Reminder: Add your name, surname, and student number as a comment line at the top of each of your programs. Use comments to name and explain each program.

Submit: Upload the compressed (**.zip**) project folder to eFundi.

Create a C++ project in Code::Blocks, and save it as:

Test1_student number *(include your student number)*

Problem

You are tasked with creating a basic banking system program. The program should allow customers to perform three types of transactions: deposit, withdrawal, and balance inquiry. The program should continue running until the user chooses to exit:

Instructions

1. Prompt the user to enter a numerical password.
   a. A password is considered valid if the sum of its digits is divisible by 7.
   b. This means that when summing the digits of the password, then dividing it by 7 should result in an integer (without a decimal part).
   c. Ex. 1: **123456789** -> **1+2+3+4+5+6+7+8+9** = **45**. **45** ÷ **7** = **6.42**… INVALID PASSWORD.
   d. Ex. 2: **43434343** -> **4+3+4+3+4+3+4+3** = **28**. **28** ÷ **7** = **4** VALID PASSWORD.
   e. If the password is successful, enter the banking system and display the menu items as outlined in point 2 .
   f. If the password is unsuccessful, display an appropriate error message and exit the system.
2. Implement a menu system with the following options:
   a. Deposit
   b. Withdrawal
   c. Balance Inquiry
   d. Exit
3. Use appropriate loops to ensure that the user can repeatedly perform transactions until they choose to exit.
4. The program should check for invalid inputs and handle them appropriately, asking the user to re-enter the input (see notes in the following section on program flow).
5. The program should keep track of the account balance and update it accordingly after each transaction.
6. Assume that the initial account balance is R1000.50

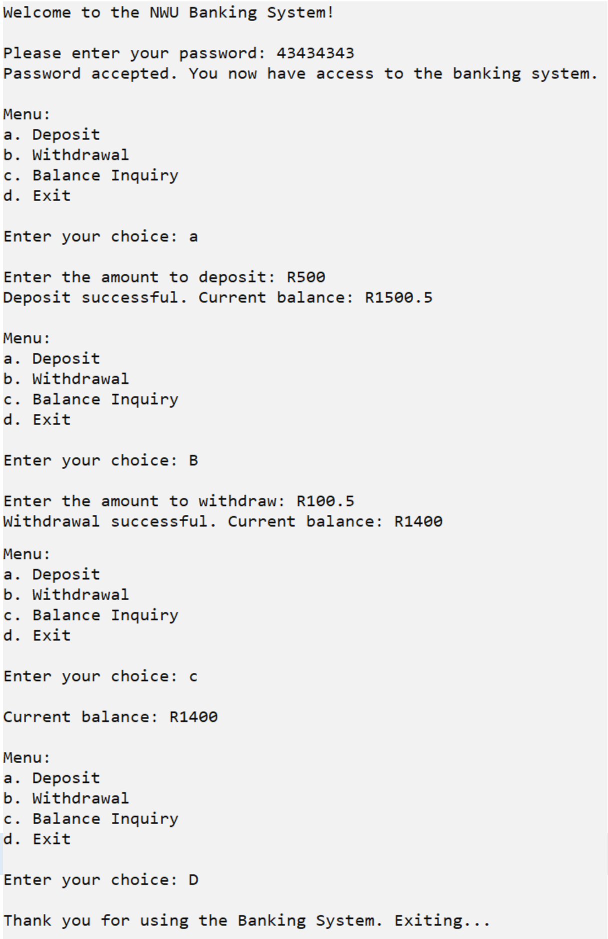
Hints – Program Flow

1. Prompt the user to enter a correct password. The program continues if the password is valid.
2. The program starts with an initial balance of R1000.50.
3. It displays a menu with options for deposit, withdrawal, balance inquiry, and exit.
4. The program uses a do-while loop to keep displaying the menu and processing transactions until the user chooses to exit.
5. The program uses a switch-case statements to handle the different options on the menu.
6. If the user's choice is option 1 "Deposit" they are prompted to enter the deposit amount, and the account balance is updated accordingly. No validation required.
7. If the user's choice is option 2 "Withdrawal" , they are prompted to enter the withdrawal amount, and the account balance is updated if the withdrawal is valid. A withdraw amount is valid if it is less or equals to the balance. If the withdraw amount is invalid, display an appropriate message which includes the available amount and request the user is re-enter the withdraw amount until a valid amount is entered.
8. If the user's choice is option 3"Balance Inquiry," the current account balance is displayed.
9. If the user selects "Exit," the program displays a farewell message and the loop terminates, ending the program.

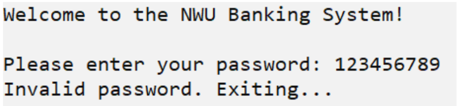
Note

- Remember to comment your code and follow good programming practices, such as descriptive variable names and input validation. **Include your Student number and Surname in the code as comments**
- Remember to format currency values to display two decimal values.
- Test your program thoroughly to ensure accuracy. Consider edge cases and invalid inputs to ensure your program handles all scenarios appropriately.

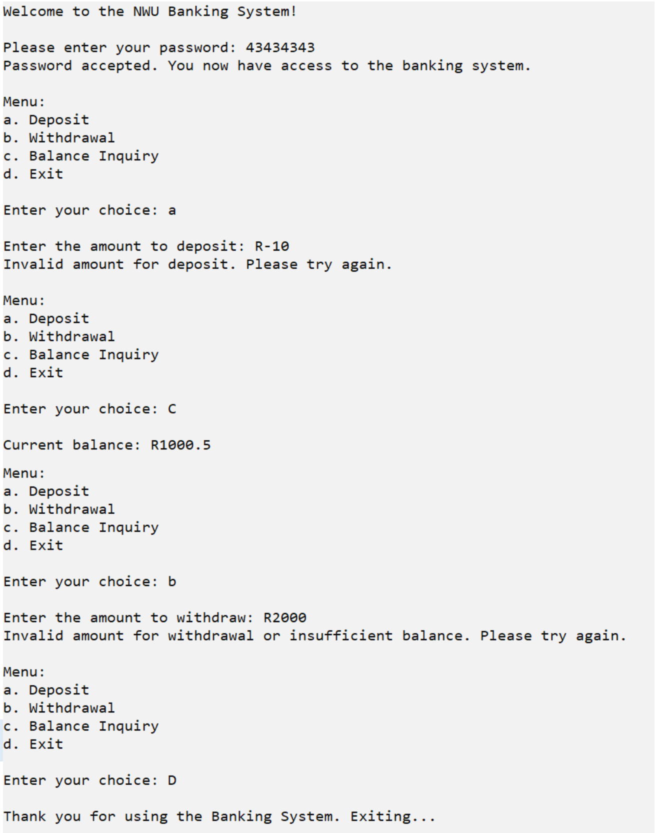
Example of input/output



```
Welcome to the NWU Banking System!

Please enter your password: 43434343
Password accepted. You now have access to the banking system.

Menu:
a. Deposit
b. Withdrawal
c. Balance Inquiry
d. Exit

Enter your choice: a

Enter the amount to deposit: R500
Deposit successful. Current balance: R1500.5

Menu:
a. Deposit
b. Withdrawal
c. Balance Inquiry
d. Exit

Enter your choice: B

Enter the amount to withdraw: R100.5
Withdrawal successful. Current balance: R1400

Menu:
a. Deposit
b. Withdrawal
c. Balance Inquiry
d. Exit

Enter your choice: c

Current balance: R1400

Menu:
a. Deposit
b. Withdrawal
c. Balance Inquiry
d. Exit

Enter your choice: D

Thank you for using the Banking System. Exiting...
```



```
Welcome to the NWU Banking System!

Please enter your password: 123456789
Invalid password. Exiting...
```



```
Welcome to the NWU Banking System!

Please enter your password: 43434343
Password accepted. You now have access to the banking system.

Menu:
a. Deposit
b. Withdrawal
c. Balance Inquiry
d. Exit

Enter your choice: a

Enter the amount to deposit: R-10
Invalid amount for deposit. Please try again.

Menu:
a. Deposit
b. Withdrawal
c. Balance Inquiry
d. Exit

Enter your choice: C

Current balance: R1000.5

Menu:
a. Deposit
b. Withdrawal
c. Balance Inquiry
d. Exit

Enter your choice: b

Enter the amount to withdraw: R2000
Invalid amount for withdrawal or insufficient balance. Please try again.

Menu:
a. Deposit
b. Withdrawal
c. Balance Inquiry
d. Exit

Enter your choice: D

Thank you for using the Banking System. Exiting...
```

CMPG 121 – Test 1		
Item	**Mark allocation**	
Password Validation (5 marks)		
Correctly calculating the sum of digits in the password.		/2
Appropriately validating the password based on the sum of digits (validating correctly and denying invalid passwords)		/3
Menu Implementation (3 marks)		
The menu is displayed correctly with options for deposit, withdrawal, balance inquiry, and exit.		/2
The user is prompted to enter their choice and the program handles invalid inputs gracefully		/1
Deposit Functionality (6 marks)		
The deposit option allows the user to enter a valid positive amount.		/2
The account balance is updated correctly after a successful deposit.		/2
Proper messages are displayed for successful and invalid deposits.		/2
Withdrawal Functionality (6 marks)		
The withdrawal option allows the user to enter a valid positive amount.		/2
The account balance is updated correctly after a successful withdrawal.		/2
Proper messages are displayed for successful and invalid withdrawals (insufficient balance or invalid amount).		/2
Balance Inquiry Functionality (2 marks)		
The balance inquiry option displays the current account balance correctly.		/2
Exit Option (2 marks)		
The program exits when the user chooses the exit option, with an appropriate message		/2
Looping and Control Structures (9 marks)		
The program uses appropriate do-while, while, and for loops.		/3
If statements are used effectively to handle different scenarios and user inputs.		/3
Switch-cases statements are used effectively to handle different scenarios and user inputs.		/3
Code Structure and Readability (2 marks)		
The code is well-structured with appropriate indentation and comments. Variable names and function names are meaningful and follow a consistent naming convention.		/2
File name according to the specification (penalty 2), correct file extension (project zipped) (penalty – not marked), only the latest versions submitted (penalty 1 per additional program submission)		
TOTAL	**0**	**35**