Solo Coding Round Assignment - Banking

- Code can be written in any Object oriented or functional language.
- Code should be idiomatic for the language chosen.
- Code should follow SOLID principles.
- Solutions with Unit tests will be given more priority. (Add if possible)
- Use Git (or other version control tools) to make frequent commits. (Add if possible).
- Please follow the directory structure and file name conventions of the language you use.
- Follow good coding practices and principles (These carry weightage in the review)
- Remember we are not only looking at completing the solution, but also the approach one takes at solving it.

Interview Format

- After receiving the problem statement, you are expected to read it thoroughly.
- The interviewer will help out with any doubts related to the problem statement.
- **Please note, the submitted solution must pass the sample test cases to have a chance of qualifying for the next round.**

Problem Statement

- Create a mini banking system which handles deposits and withdrawals made by a user.
- The bank accepts 2 currency exponents - Dollar and Cents (**Dollar:Cents::1:100**)
- Write a program where you can
    - Deposit money into the user account in the bank
        - Money can be deposited in a combination of Dollars and Cents.
    - Withdraw money from the user account in the bank
        - Money can be withdrawn in combination of Dollars and Cents.
    - Check the balance.
        - The balance can do go into negative values as well
- Money input and output should be in the format "***x***D ***y***C" where ***x*** are dollars and ***y*** are cents
    - If x is zero it can be skipped, the value can just be "***y***C".
        - Same rule applies for ***y***.
    - ***x*** and ***y*** can be negative numbers as well, basic rules of mathematics will apply while crediting and debiting values.
        - 1D - (-1D) = 1D + 1D = 2D
        - 1D + (-1D) = 1D - 1D = 0D

Important Points

- Currently the bank only operates for a single user, i.e, you.

- Maximum amount of money that can be stored is 10^9.

Sample Flow

```
$ Select an option:
   1. Credit
   2. Debit
   3. Check Balance
   4. Exit
=> 1
$ Enter Amount:
=> 5D 60C
$ Successful!
$ Select an option:
   1. Credit
   2. Debit
   3. Check Balance
   4. Exit
=> 2
$ Enter Amount
=> 20C
$ Done
$ What would you like to do:
   1. Credit
   2. Debit
   3. Check Balance
   4. exit
=> 3
$ Current Balance is 5D 40C
$ What would you like to do:
   1. Credit
   2. Debit
   3. Check Balance
   4. Exit
=> 4
$ Thank you!
```

Note : The input and output can be published to command line interface or a file.