ET-580 – Operator Overloading - Practice

1. Implement the following class:

   a. A class named *Money*

   b. Integer data members: *dollars cents*

   c. Implement these constructors using constructor delegation:
      1. Default constructor
      2. Constructor with a single parameter, *dollars*
      3. Constructor with two parameters, *dollars* and *cents*
         If cents is greater than 100, convert into dollars and cents
      4. Implement constructor delegation for all constructors
      5. Assume dollars and cent input values are always positive.

   d. Accessors and Mutators
      If mutator cents is greater than 100, convert into dollars and cents
      Assume dollars and cent input values are always positive.

   e. A *print* function to display money in proper format (*$dollars.cents*)
      Take into account the following:
      $5.05 (dollar is 5, cents is 5)
      $0.35 (dollar is 0, cents is 35)

   f. Set appropriate functions to be const member functions

   g. Implement a driver program to test all class functions

<u>Example Output</u>

```
m1: $5.00
m2: $2.30
```

2. Implement the following:

   a. Overload the *unary* - operator to negate a Money object
      This should negate dollars and negate cents

   b. Overload the subscript [] operator so that index 0 returns
      dollars and index 1 returns cents

   c. Update the *print* function to print negated Money objects
      Take into account the following:
      $5.05 (dollar is 5, cents is 5)
      $0.35 (dollar is 0, cents is 35)
      -$5.05 (dollar is -5, cents is -5)
      -$0.35 (dollar is 0, cents is -35)

   e. Implement a driver program to test both operators

<u>Example Output</u>

```
$3.50
$-3.50
3
50
```

3. Implement the following:

   a. Overload the == operator. Make sure it supports auto type conversion

   b. Overload the + operator. Make sure it supports auto type conversion
      Assume this will only be used with positive Money objects.

   c. Implement a driver program to test both operators with and without
      automatic type conversion and operator chaining

<u>Example Output</u>

```
m1: $3.50
m2: $2.60
m1 == m2? 0
m1 + m2: $6.10
10 + m1 + m2: $16.10
```

4. Implement the following:

   a. Overload the insertion operator << to output Money objects

   b. Overload the extraction operator >> to input Money objects
      If cents input is greater than 100, convert into dollars and cents

   c. Implement a driver program to test both operators with chaining

<u>Example Output</u>

```
m1: $3.50
Enter dollars: 10
Enter cents: 120
m1: $3.50 m2: $11.20
```