CSCI 2170 OLA 3

Program One

Write a program named *change.cpp*. It takes one change value (between 1 and 99) and prints out what coins can be used to make that change. Use coin denominations of 25 cents (quarters), 10 cents (dimes), and 1 cent (pennies). Do not use nickels and half-dollar coins.

Your program should implement the following function (among others):

```
void ComputeCoin(int cointValue, int& number, int& amountLeft);
// Precondition:    0 < coinValue < 100; 0 <= amountLeft < 100.
// Postcondition:   number has been set equal to the maximum number
//                  of coins of denomination coinValue cents that can be obtained
//                  from amountLeft cents. amountLeft has been decreased by the
//                   value of the coins, that is, decreased by (number * coinValue).
```

For example, if the value of the variable amountLeft is 86, after the following call:

```
ComputeCoins(25,number,amountLeft);
```

the value of number will be 3 and the value of amountLeft will be 11, because if you take 3 quarters from 86 cents, that leaves 11 cents.

After a second call:

```
ComputeCoins(10, number, amountLeft);
```

the value of number will be 1 and the value of amountLeft will be 1, because if you take 1 dime from 11 cents, that leaves 1 cent.

Therefore, for this example, the coins can be used to make change for 86 is 3 quarters and 1 dime and 1 cent.

Your main function should consist of a while loop that reads the change values from a data file one value at a time. For each value read, it computes and displays what coins can be used to make that change.

Copy the data file coins.dat into your project directory.

Here is an example data file:

```
86
30
56
69
```

For this data file, the output from the program should be:

```
86 cents can be changed using:
3 quarter(s)
```

```
1 dime(s)
1 cent(s)

30 cents can be changed using:
1 quarter(s)
0 dime(s)
5 cent(s)

56 cents can be changed using:
2 quarter(s)
0 dime(s)
6 cent(s)

69 cents can be changed using:
2 quarter(s)
1 dime(s)
9 cent(s)
```

Program Two

Write a program named ***isbn.cpp*** to check for ISBN number. The International Standard Book Number (ISBN) is a unique, numerical commercial book identifier. The ISBN is 13 digits long if assigned after January 1, 2007. The last digit of the thirteen-digit ISBN is called *check digit*, which is used to verify if an ISBN is a valid ISBN. The following describes the way to calculate the check digit.

The calculation of an ISBN-13 check digit begins with the first 12 digits of the 13-digit ISBN (thus excluding the check digit itself):

- Each digit, from left to right, is alternately multiplied by 1 or 3,
- Those products are summed and calculated modulo 10 to give a value ranging from 0 to 9.
- The value is subtracted from 10, that leaves a result from 1 to 10.
- A zero (0) replaces a ten (10), so, in all cases, a single check digit results.

For example, the ISBN-13 check digit of 978-0-306-40615-? is calculated as follows:

```
s = 9 * 1 + 7 * 3 + 8 * 1 + 0 * 3 + 3 * 1 + 0 * 3 + 6 * 1 + 4 * 3 + 0 * 1 + 6 * 3 + 1 * 1 + 5 * 3
  =  9   + 21  +  8  +  0   +  3  +  0  +  6  + 12 +  0   + 18 +   1   + 15
  = 93
93 mod 10 = 3
10 - 3  = 7
```

Thus, the check digit is 7, and the complete sequence is ISBN 978-0-306-40615-7.
Therefore, given a thirteen-digit ISBN, we can calculate the check digit using the above approach and compare it with the last digit in the given ISBN to decide if it is valid or not.

Your program reads ISBN numbers from the data file (***isbn.dat***) and checks if they are valid ISBN numbers. Assume that there is no space, dash or other symbols within an ISBN number. You are required to declare and define the following two value returning functions:

- The CheckDigit function takes a C++ string containing ISBN as the parameter, and returns the theoretic check digit from the first 12 digits of the ISBN.
  The calculation of an ISBN-13 check digit begins with the first
  12 digits of the 13-digit ISBN (thus excluding the check digit itself).
  - Each digit, from left to right, is alternately multiplied by 1 or 3,
  - Those products are summed and calculated modulo 10 to give a value ranging from 0 to 9.
  - The value is subtracted from 10, that leaves a result from 1 to 10.
  - A zero (0) replaces a ten (10), so, in all cases, a single check digit results.

  This function returns (explicitly) the theoretic check digit.
- The IsValidISBN function takes a C++ string containing ISBN as the parameter, and returns a boolean value to indicate if the ISBN is valid or not. It calls the CheckDigit function to compute the check digit of the ISBN number.

Here is the skeleton program you may copy/paste to get started.

```
#include <iostream>
#include <string>
#include <fstream>
#include <cassert>
using namespace std;

// Function Prototypes
//Provide function prototypes for CheckDigit and IsValidISBN


int main( )
{
    string      isbn;       //ISBN number to be processed
    bool        isValid;     //indicates if the isbn is valid
    ifstream     myIn;

    myIn.open("../isbn.dat");
    assert(myIn);
    while ( myIn >> isbn)   //read a ISBN number from data file
    {
        // Add statement that makes a call to the function IsValidISBN to find out the value of
variable isValid.



        if ( isValid )
            cout << isbn << " is a valid ISBN number" << endl;
        else
            cout << isbn << " is not a valid ISBN number" << endl;
    }

    myIn.close();
    return 0;
}
```

```
// Define the IsValidISBN function here. The IsValidISBN function will call function
"CheckDigit" to compute the check digit


// Define CheckDigit function here.
```

Here is an example output of the program:

```
9780262026499 is a valid ISBN number

9780321480798 is a valid ISBN number

9780596514552 is not a valid ISBN number

9780596514556 is a valid ISBN number

9780596529260 is a valid ISBN number

9781596510510 is not a valid ISBN number
```

What to submit?

Submit the source programs: **change.cpp** and **isbn.cpp.**