Project 1
CS 1344: Programming in C
Due October 5 at 11:59 pm

Vending machine problem

Project 1 must be entirely your own work. Assistance should only be sought or accepted from the course instructor during his office hours or during his specifically announced lab hours in McB 116/118. This assignment will give you the opportunity to learn 1) File input and output 2) Use of nested loops and 3) Use of two way (if) and multi way (switch) selectors.

Problem Statement:

You are to write a program simulating the coin counting and change giving part of a vending machine. When a customer buys an item from the vending machine, your program should calculate the denominations of coins that are to be given back to the customer as change. After all the transactions are over, you have to find the total collection, total balance given and the net collection for that day. In addition, you need to give a report showing how much of each denomination of coins should be removed or added to maintain the beginning balance and count of coins.

Input:

The first line of the input file will be the number of coins of denominations starting from one dollar, quarter, dime and nickel. From the second line onwards there will be an arbitrary number of transactions showing the price of the product purchased and the denominations of coins inserted. **Your program must read inputs from a file named "Project1.in".**

Assumptions:

1) The vending machine will use only coins of following denominations.
   One Dollar, Quarters, Dimes and Nickels.
2) The customer will make a selection for the item, and insert the coins only after selection. This way, once the machine counts coins for amount exceeding the price of the item, the selected item will be vended.
3) All items cost less than one dollar and all prices are multiples of five.
4) Because of these assumptions, the balance to be given will be always less than a dollar.
5) At no point will the machine run out of coins to give change to the customer.
6) To save on the number of coins to be given as change, you will start from the highest denomination (quarters) to the lowest denomination (nickels) while computing the denominations to be given as change.

Sample input data:

0 39 55 40 *(Beginning balance of coins indicating the numbers of one Dollar, Quarters, Dimes and Nickels in that order)*
20 10 5 100 *(The first number represents the price of the item, and the rest of the values indicate the values of coins inserted)*
65 10 5 25 10 5 25
85 25 10 5 100
50 10 5 5 5 100
75 25 10 5 25 10
45 10 5 100
90 100
60 100
75 25 10 5 25 10

In this sample, the fist line means that there are 0 dollars, 39 quarters, 55 dimes, and 40 nickels adding up to 17 dollars and 25 cents at the beginning of the day.

The second line represents the first of the arbitrary number of transactions. The first number (20) is the price of the item and remaining numbers represent the denominations of coins that are inserted. Our customer may insert a one-dollar coin at the end when he/she is exhausted of all other coins.

Output:

Your program should create and send output to a file named "Project1.out". Your output should have the exact form illustrated below. Be very careful to provide for blank lines exactly where they are shown, and follow spelling and capitalization exactly. You can change the name of the programmer to your name. Do not worry about the exact number of hyphens on the divider lines that are continuous across a majority of the width of the page. Failure to follow the remainder of the form exactly will result in the automated grader deducting points. Here is an example of output for the previously mentioned input data.

```
VENDING MACHINE PROBLEM
Name of programmer: Jayan K. Nair
-------------------------------------------------------
Total amount at the beginning of the day in Dollars:
17.25

Cust.No  Cost    Paid  Balance Quarters   Dimes Nickels
-------------------------------------------------------
   1      20     115      95        3       2       0
   2      65      80      15        0       1       1
   3      85     140      55        2       0       1
   4      50     125      75        3       0       0
   5      75      75       0        0       0       0
   6      45     115      70        2       2       0
   7      90     100      10        0       1       0
   8      60     100      40        1       1       1
   9      75      75       0        0       0       0
-------------------------------------------------------
             Report on Net collection in Dollars
-------------------------------------------------------
Total received      Total Change Given  Net Collection
      9.25               3.60               5.65
-------------------------------------------------------
To maintain the original coin count, add(+) or remove(-)
(-) 6 Dollar(s)
(+) 4 Quarter(s)
(-) 3 Dime(s)
(-) 7 Nickel(s)
```

Documentation: Your program should include all the documentation called for in the "Elements of Programming Style," including a header for the main program and readable in-line documentation that describes the purpose of each logical segment of code.

Submission:
You should submit your source code (the *.cpp file) electronically to the automated grader. Do not submit the input or the output file. Before you submit your program, make sure your program runs perfectly and provides right output with the sample test data. Be aware that your program will be tested against a randomly generated input data. You will be allowed a maximum of four submits. Your submissions are archived by the grader and will be hand graded for the entire class for conformance to departmental documentation and coding style standards. This project should be submitted as Project Number 1.

Distribution of Points:

Design outline	: 10%	(Due on September 15 in class)
Correct output	: 65%	
Documentation	: 25%	
Total	:100%	

Bonus points and late penalty:

Bonus points	: 10% (2% per day up to a maximum of 10 points)
Late penalty	: 20% per each day late

Hints:

1) This project is non-trivial, especially if you do not have enough programming experience. So start early.
2) Develop a clear idea (design) about how to solve the problem before you start coding. If you are not sure about it, ask the instructor.
3) Solve your program incrementally. Do not attempt to code all of it at once and then do the testing. Instead, divide the problem into logical sub units and code and test the sub units incrementally. Do not attempt to proceed unless the code already written is working properly.
4) One possible way of dividing the problem will be:
    a) Code and Test the file input and output and proceed only if the I/O is working properly.
    b) Now write the loops. Proceed only when the loops are working properly.
    c) Then you can write the code to calculate the total of the day's transactions.
    d) Once all these are working properly, do the code for the final denominations of coins to be removed or added.
5) Since looping and selections are required for this project, to get a head start you can read ahead on your notes or work through the sample programs posted on the web.