Assignment #1

Topic(s): I/O & Functions in C
Date assigned: Wednesday, August 29, 2012
Date due: Wednesday, September 5, 2012
Points: 15

You have now had a full year of C++ and we are now going to move to the world of Data Structures in C. For this first assignment, I would like to introduce you to the differences of input and output I/O in C and C++ while reinforcing the use of functions when writing code.

You are to rewrite the sample solution below using functions.

Sample Problem:

Write a complete C (fully documented-in other words fully commented) program that allows the user the ability to input from the keyboard the price of an item to be purchased and the amount received in payment (both amounts are in cents). Compute the change in dollars, half-dollars, quarters, dimes, nickels, and pennies using the highest amounts of each starting with the dollars. As output, you are to print the number of dollars, half-dollars, quarters, dimes, nickels, and pennies given back as change.

Sample Solution:

```
/*************************************************************************
 File name:  01ChangeMaker.c
 Author:     Doug Ryan
 Date:       9/1/10
 Edited:     Shereen Khoja
 Edited Date:8/22/12
 Class:      CS300
 Assignment: Sample Problem
 Purpose:    This program shows a C program that calculates change from
             an item price and amount received. The change is in the form
             of dollars, half-dollars, quarters, dimes, nickels, and pennies.
*************************************************************************/

#include <stdio.h>
#include <stdlib.h>

const int DOLLARS = 100;
const int HALF_DOLLARS = 50;
const int QUARTERS = 25;
const int DIMES = 10;
const int NICKELS = 5;
const int PENNIES = 1;

int main(void)
{
  int purchasePrice;
  int amountReceived;
  int change;
  printf("*************************************************");
  do
  {
    printf("\nEnter the price of the item (in cents): ");
    scanf("%d", &purchasePrice);
  } while (purchasePrice <= 0);
```

```c
do
  {
    printf("\nEnter the amount received (in cents): ");
    scanf("%d", &amountReceived);
  } while (amountReceived <= 0);

  change = amountReceived - purchasePrice;


  printf("\n*************************************************");
  printf("\nThe item price is:\n");
  printf("\nDollars = %d", purchasePrice / DOLLARS);
  purchasePrice %= DOLLARS;
  printf("\nHalf-dollars = %d", purchasePrice / HALF_DOLLARS);
  purchasePrice %= HALF_DOLLARS;
  printf("\nQuarters = %d", purchasePrice / QUARTERS);
  purchasePrice %= QUARTERS;
  printf("\nDimes = %d", purchasePrice / DIMES);
  purchasePrice %= DIMES;
  printf("\nNickels = %d", purchasePrice / NICKELS);
  purchasePrice %= NICKELS;
  printf("\nPennies = %d", purchasePrice);

  printf("\n*************************************************");
  printf("\nYour payment is:\n");
  printf("\nDollars = %d", amountReceived / DOLLARS);
  amountReceived %= DOLLARS;
  printf("\nHalf-dollars = %d", amountReceived / HALF_DOLLARS);
  amountReceived %= HALF_DOLLARS;
  printf("\nQuarters = %d", amountReceived / QUARTERS);
  amountReceived %= QUARTERS;
  printf("\nDimes = %d", amountReceived / DIMES);
  amountReceived %= DIMES;
  printf("\nNickels = %d", amountReceived / NICKELS);
  amountReceived %= NICKELS;
  printf("\nPennies = %d", amountReceived);

  printf("\n*************************************************");
  if (change < 0)
  {
    printf("\nYou owe more money");
  }
  else
  {
    printf("\nYour change is:\n");
    printf("\nDollars = %d", change / DOLLARS);
    change %= DOLLARS;
    printf("\nHalf-dollars = %d", change / HALF_DOLLARS);
    change %= HALF_DOLLARS;
    printf("\nQuarters = %d", change / QUARTERS);
    change %= QUARTERS;
    printf("\nDimes = %d", change / DIMES);
    change %= DIMES;
    printf("\nNickels = %d", change / NICKELS);
    change %= NICKELS;
    printf("\nPennies = %d", change);
  }

  printf("\n*************************************************");
  printf("%c",'\n');

  return EXIT_SUCCESS;
}
```

Your Problem:

The above solution was written by someone not comfortable with functions. You are to modify the above solution so that the solution is correctly written using functions. Place the function prototypes above **`int main (void)`** and place the function definitions below the main function. You must determine what functions to write. Think about reusability and also look for patterns of repetition in the code.

Notes:

1. Your output is to look and work EXACTLY the same as the sample solution above.
2. We will use the coding guidelines coding.C.v6.pdf found on the CS300 home page.
3. To get started, make a directory named **cs300_1_PUNetID** and place your **01ChangeMaker.c** file in this directory. These names are required.
4. Your code is to be written in C using Geany and tested on Zeus. Programs written in other environments will not be graded. We will be using a submit script for submitting programs. I will talk about using the submit script in class and we will submit a sample program in class.
5. Function documentation can be found in the coding standards document. Make sure you follow the documentation exactly.
6. You must provide a hard copy (color, double-sided, stapled) as well as an electronic copy by 9:15am September 5, 2012.
7. Keep a record of the time you spent on this assignment. Place a comment in your code indicating how long you worked on this assignment. For example: **`//hours worked = 5.5 hours`**.

For testing on Zeus:

How to compile your project from the command line:

```
shereen@zeus:~> gcc -Wall -g -o 01ChangeMaker 01ChangeMaker.c
```

How to run your project from the command line:

```
shereen@zeus:~> ./01ChangeMaker
```

How to submit your project

To produce a compressed tar file:

```
shereen@ralph:~> tar czf cs300_1_PUNetID.tar.gz cs300_1_PUNetID
```

To copy your compressed tar file to zeus:

```
shereen@ralph:~> scp cs300_1_PUNetID.tar.gz  shereen@zeus:
```

To extract your compressed tar file on zeus for testing:

```
shereen@zeus:~> tar xzf cs300_1_PUNetID.tar.gz
shereen@zeus:~> cd cs300_1_PUNetID
shereen@zeus:~> now run gcc and then test your code
```

To use the submit script the command is:

```
submit classname filename
```

EXAMPLE

```
shereen@zeus:~> submit cs300f12 cs300_1_PUNetID.tar.gz
```

which will submit the zipped up tar file `cs300_1_PUNetID.tar.gz`

Once you have successfully submitted a file you will get a receipt which is a file that will end in **`.receipt`**. You can make sure your file was submitted correctly by typing the command:

```
checkReceipt submittedfile classname receiptfile
```

EXAMPLE

```
shereen@zeus:~> checkReceipt cs300_1_PUNetID.tar.gz cs300f12

     cs300_1_PUNetID.tar.gz.cs300f12.receipt
```

will produce the result:

```
HASH>>>kı`Uyh
HASH>>>kı`Uyh
SUCCESS! Your receipt is valid
```

DO NOT MOVE OR MODIFY THE SUBMITTED FILE **OR** THE RECEIPT FILE OR THE CHECK RECEIPT COMMAND WILL NOT BE SUCCESSFUL.