EECS 10: Assignment 5

October 21, 2005

Due Monday 10/31/2005 12:00pm

1 Shopping cart menu [20 points]

Write a program to display the menu of a simplified electronic shopping cart where users can do several operations such as “add an item”, “remove an item”, “add multiple items “, “remove multiple items”, and “exit”. The menu should display the current number of items in the cart, and the total amount of items in the cart after each operation. The menu should be repeated until the user chooses option “5”.
In this problem, we will assume that the user always inputs correct values (positive floating-point number for the item price and a positive integer for the number of items). We will also assume that the user does not take any items out that she/he has not put in first.

The menu should display as follows:

```
****************************************************
*  Welcome to EECS10 shopping cart menu
*  1) Add an item
*  2) Remove an item
*  3) Add multiple items
*  4) Remove multiple items
*  5) Exit
* Please enter your choice:
****************************************************
```

If the user chooses option 1, the program will ask the user to input the unit price of the item and then display the new “in cart” information:

```
Please input the unit price: $ 3.25
You have 1 item(s) in the shopping cart. Total amount is $ 3.25
```

If the user chooses option 2, the program will ask the user to input the unit price of the item and then display:

```
Please input the unit price of the item you want to remove: $ 3.25
You have 0 item(s) in the shopping cart. Total amount is $ 0.00
```

If the user chooses option 3, the progra m will ask the user to input the unit price and number of items as follows:

```
Please input the unit price: $ 5.00
Please input number of items you want to buy: 10
You have 10 item(s) in the shopping cart. Total amount is $ 50.00
```

If the user chooses option 4, the program will ask the user to input the unit price and number of items to remove:

```
Please input the unit price: $ 5.00
Please input number of items you want to remove: 4
You have 6 item(s) in the shopping cart. Total amount is $ 30.00
```

If the user chooses option 5, the program will display the end result and then end.

```
You have 6 item(s) in the shopping cart. Total amount is $ 30.00
Thanks for using EECS10 shopping cart. Bye-bye!
```

What to turn in:

Implement the program in C and record the results in a **`cart.script`** file for the following scenario:

The test case includes several steps. You must follow these steps once you run your program!

1. Add one item of $1.99
2. Add one item of $7.98
3. Add 10 items of $0.79 each
4. Remove one item of $1.99
5. Add 2 items of $37.50 each
6. Remove 4 items of $0.79 each
7. Exit

Submit the files **`cart.c`**, **`cart.script`**, and **`cart.txt`**.

2 Square root approximation [20 points + 5 points (extra credit)]

Write a program to calculate the square root of any positive floating-point value.
At the beginning, the program should ask the user to input a positive number **`N`** in the range between 0 and 10000.

```
Please input a positive number (1 to 10000):
```

We will use a binary search approximation technique for this assignment. In particular, the program will always keep a range of a left bound **`L`** and a right bound **`R`**, where the actual square root **`S`** lies somewhere between **`L`** and **`R`**:

```
          L = S = R
```

The binary approximation then works as follows. First, we compute a value **`M`** that lies in the middle between the left bound **`L`** and the right bound **`R`**: **`M`** = **`L`**+(**`R`**-**`L`**)/2. Then, if **`M*M`** is less than **`N`**, the square root obviously lies somewhere in the right half of the current range (i.e. within **`M`** to **`R`**), otherwise in the left half of the current range (i.e. within **`L`** to **`M`**). The program then can use the proper half of the range as the new range and repeat the whole process. With each iteration, the search range is effectively reduced to half its previous size. Because of this, this technique is called binary search.
To start the search, we will use the range from **`0`** to **`N`** (which is guaranteed to contain the square root of **`N`**). We will stop the iteration, once we have reached a range that is smaller than **`0.000001`** so that we reach a precision of 6 digits after the decimal point for our approximation.

The pseudo-code of the algorithm can be written as follows:

```
Start with a range of 0 to N
As long as the range is not accurate enough, repeat the following steps:
    Compute the middle of the range
    Compare the square of the middle value with N
    If the middle value is less than the square root
        Use middle-to-right as the new range
    Otherwise
        Use left-to-middle as the new range
Output the middle of the latest range as result
```

For example, to compute the square root of 10, the program will start with 5, which is in the middle between 0 and 10. Since 5*5=25 is larger than 10, the program will try the middle number 2.5 of left bound (0 to 5). Thus, the program compares 2.5*2.5 with 10. Because the result 6.25 is smaller than 10, it will pick 3.75 (the middle number of 2.5 and 5) as the next guess. By picking the middle number every time and comparing its square with the original number, the program gets closer to the actual square root.

To demonstrate the approximation procedure, your program should print the approximated square root in each iteration, as follows:

```
Please input a positive number (1 to 10000): 278
Iteration 1: square root of 278 is approximately 139.000000
Iteration 2, square root of 278 is approximately 69.500000
Iteration 3, square root of 278 is approximately 34.750000
...
Iteration n, square root of 278 is approximately 16.673332
```

For 5 extra credits:

Improve your program so that it can calculate the **n**-th root of any value. The value **n** should be a positive integer input by the user.

For example:

```
Please input a positive number (1 to 10000): 278
Please input the value of integer n (n>0): 5
Iteration 1, the 5th root of 278 is: *.******
...
Iteration n, the 5th root of 278 is: *.******
```

What to turn in:

If you modify the program for the extra credit, please **ONLY** submit **`root-extra.c`**, **`root.script`**, and **`root.txt`**.
The script file should show the output of your program when the user inputs 278 and n=5.

If you only finish the basic program, please submit the files **`root.c`**, **`root.script`**, and **`root.txt`**.
The script file should show the output of your program when the user inputs 278.

3 Submission

Submission for these files will be similar to last week's assignment. The only difference is that you need to create a directory called **`hw5/`**. Put all the files listed above in that directory and run the **`/ecelib/bin/turnin`** command to submit your homework.