C Programming lab
Batch1 Midterm Exam

Prepared by: M. Rao

Revision Date: September 29, 2014

Preliminaries

This exam has a total of 20 points. All tasks are all or nothing. There is no partial credit given for individual tasks. The duration of this exam is two hours. You must put all your programs in the correct directories and submit from the specified directory or you will receive zero credit.

Your are allowed to write the code in unindented form. Also no need of comments in your files. However make sure that your output should be in the format specified. For example, spaces, newlines and decimals should be followed as specified in the test cases. Your code will be checked for only output. Given a test case, your code should work as expected otherwise you will lose points for that task.

Create a directory named *batch1* that hangs off your `/home/IMT2013XXX` directory using 'mkdir' linux command. Create four directories inside the *batch1* directory: *root*, *delay*, *vegetables*, and *donation* directories. You should see the following output:

```
4   ./root
4   ./delay
4   ./vegetables
4   ./donation
4   .
20  .
```

The names must be as shown, but the order and the numbers do not matter.

Task 1: Separate square root and cube root integer numbers from a list of numbers(5 points)

Develop a program in *root.c* in *root* directory which will take command line arguments consisting of all positive integer numbers greater than ZERO. The data parsed from the command line arguments are separated by a space. We will pass only positive integer numbers. Your program should identify whether the number has a integer (no real value) square root or integer cube root number and put these numbers into *sqroot.dat*, *croot.dat* and *others.dat* files. Note that there may be some number which has both integer square root and cube root, in that case the number will be inserted in *sqroot.dat* file.

If your program is tested as follows

```
./root 1 2 100 81 27 8 300
```

then your program should have the following data in sqroot.dat file

```
1 100 81
```

and following data in croot.dat file

```
 27 8
```

and following data in others.dat file

```
2 300
```

Remember to put a space between the numbers in output files as well. Note that your program will be tested with different numbers and different count of numbers.

[HINT: If you need to write to a file, you can use fprintf statement with a 'w' mode]

Task 2: Delay (5 points)

Write a single function *compute* in *util.c* file in *delay* directory to evaluate parisitic delay, and effort delay with respect to number of inputs (n). The parisitic delay is given by the equation $\frac{(n^2+n+2)}{3}$ and effort delay is given by the equation $\frac{(2n-1)}{5}$. The total delay is given by parisitic delay and effort delay. However it is represented as p$< parisitic - delay >$ + e$< effort - delay >$.

The *compute* function will take atmost two arguments as formal parameters. You need to call compute function from other file *delay.c*, where you will write *main* function. The number of inputs n is supplied as command line argument. Link and compile both files to produce a *delay* executable. Your program should demonstrate the following output.

```
./delay 3
p4.67 + e1.00
```

Another example:

```
./delay 0.5
p0.91
```

Note that in the previous example, effort-delay comes out to be zero and hence only parasitic delay is mentioned.

Note that you have to print the values only in main function in *delay.c* file. No printf's/fprintf's should be placed in *util.c* file. *delay.c* file will only call the function and display the result. No computation will is expected in *delay.c* file. Computation to find parasitic and effort delay needs to be done in *util.c* file in the *compute* function. Again note that you have to write a single function in *util.c* file. If more than one function is written, your solution will not be considered for grading. Also if more than 2 formal parameters are used in function definition, then your solution will not be considered correct.

Task 3: Software for vegetables shopping (5 points)

Develop a program in *veg.c* file in *vegetables* directory. Your program should take two input files (*cost.dat* and *wish.dat*) from the command line argument and should calculate the total amount to be carried before going to the vegetables shopping. Your program will be executed in the given below format.

```
./veg cost.dat wish.dat
```

The cost of each vegetable per kg is mentioned in the *cost.dat* file. The vegetables wish list is mentioned in *wish.dat* file. The *cost.dat* file has data in the following format as shown below:

```
<vegetable-name> <cost per KG>
```

One such *cost.dat* file is shown below:

```
brinjal 35.00
beans   50.00
```

The *wish.dat* file has data in the following format as shown below:

```
<vegetable-name> <Quantity in KG>
```

One such *purchase.dat* file is shown below:

```
brinjal 2.00
```

Your program is supposed to calculate the total cost of vegetables listed in *purchase.dat* file. The total cost of your shopped vegetables in the above test case is

```
70.00
```

Another example with different list of *vegetables.dat* and *purchase.dat*. One such *cost.dat* file is shown below:

```
beans   50.00
carrot  12.50
```

One such *purchase.dat* file is shown below:

```
brinjal 2.00
```

The total cost of your shopped vegetables in the above test case is

```
0.00
```

In the above example, the cost comes out to be ZERO, since *brinjal* vegetable was not found in the *cost.dat* file. Note that your final answer should have two decimal values. Also your program will be tested with different test files which will have different list but the format will remain the same. [HINT: You may wish to use strcmp to compare two strings from different files. Do man strcmp to know more about it.]

Task 4: mDonation (5 points)

IMTech2013 batch students are developing a mobile donation embedded systems in IIITB. This will help users select the donation in terms of different denomination such as Rs. 10, Rs. 5, Rs. 2 and Rs. 1. Write a program considering your code will be ported to their sending embedded controller. Embedded system development represents using minimum number of variables in your code and thus minimizing the memory required to run program. You will have to choose the variable according to the size required. If the number of variables is more than the minimum required, your solution will not be evaluated.

Write a program in a file *donation.c* in *donation* directory. Your program should take four inputs of donation from the user in the form of total amount in denomination (Rs. 10, Rs. 5, Rs. 2, and Rs. 1).

The items format in mDonation device is shown below:

```
|Ten(5-bit)|Five(3-bit)|Two(4-bits)|One(5-bits)|
```

The four inputs provided by the user should be accomodated in this format and displayed to the output console. The four inputs are provided in the following format:

```
./donation <Rs.Ten-value> <Rs.Five-value> <Rs.Two-value> <Rs.One-value>
```

For example if your program is executed with the following example:

```
./donation 10 5 2 1
```

then your program should print the following in integer format:

```
4641
```

Note that Rs. 10 is passed in the command line argument, which represents 1 in the five-bit space of Rs. 10.
Rs. 5 is passed as command line arguments, which represents 1 in the 3-bit space of Rs. 5.
Rs. 2 is passed as command line arguments, which represents 1 in the 4-bit space of Rs. 2.
Rs. 1 is passed as command line arguments, which represents 1 in the 5-bit space of Rs. 1.

Another example is shown below:

```
./donation 10 0 0 0
```

then your program should print the following in integer format:

```
4096
```

Submitting your midterm exam

First, move into your `~/batch1` directory. Then run this system command:

```
    du -a
```

and check whether all your files are located in appropriate directory.

Once you are done and ready to submit, stand up and wait for your instructor to come to your place before submission. You need to submit your midterm exam only infront of the instructor. In case if you submit without instructors presence, your submission will be unaccepted. Also sign the attendance sheet with the time of submission before leaving the room. Submit your solutions while in the `~/batch1/` directory with the command:

```
    submit clab mr batch1 <your-iiitb.org-email-address>
```

Write your name, email-id and roll numbers in the question paper as well. Give your exam question paper back to your instructor. Have a nice break!!