COSC 1336 Lab: 8

Relevant reading: Sections 7.1-7.3

Due: Nov. 6, 2:30 pm

(Late date: Nov. 13, 2:30 pm)

50 Points

Problem 1. [**15 points**] Create a file called `read_average_numbers.py`. In it, write a program that reads in a filename from the user, and displays the average of the numbers in the file. The program should assume that the file specified by the user contains only numbers, one number per line. To test your program, create a text file using Notepad and save it to the same folder where you put `read_average_numbers.py`. Put some numbers in your text file and verify that your program works correctly.

Problem 2. [**15 points**] Download the file `jokes.txt`. In it, you will find two types of (very cheesy) jokes: two-line jokes (e.g. “Why did the chicken cross the road?” jokes) and knock-knock jokes. Each joke is preceded by a number on a line by itself indicating how many lines are in the joke.

In a file called `joke_teller.py`, write a program that reads in the jokes and tells them to the user. The program should pause and wait for the user to press the enter key whenever a human joke teller would wait for the listener to respond. That is, if the listener would be expected to say “I don’t know; why?” or “Who’s there?” the program should wait for the user to press enter by using the `input` function. If you’d like, you can actually look at the user’s input and try to make the program respond wittily, but you don’t have to look at the input at all.

Before writing the `main` program, write a function called `tell_joke` which receives an already opened input file stream, and reads in and displays the joke. The function should return a Boolean indicating whether it successfully told a joke. That is, it should return `False` if the file did not contain any more jokes.

When each joke is over, the program should ask the user whether they want to hear another joke and behave accordingly. If there are no more jokes, the program should display an appropriate message and quit (it should **not** crash or hang).

The file doesn’t have many jokes, so feel free to add some (but remember to put a line indicating the number of lines in the joke before each one). Either way, make sure you submit both your program and the jokes file when you’re done.

Problem 3. [**20 points**] Download the file `cash_register.py`. In it you will find a cash register program that has a somewhat bizarre user interface, as it prints out the customer’s receipt information on the screen as it goes, which is confusing to read. Modify the program so that it prints the customer’s receipt to a file rather than on the screen. When your program works correctly, you should have an interaction like the following on the screen:

```
Enter the item name (blank to quit): coke
Enter the price for coke: $1.25
Enter the item name (blank to quit): sandwich
Enter the price for sandwich: $3.95
Enter the item name (blank to quit):

Total due: $  5.63
Enter the total paid by customer: $10
Change due:  $ 4.37
```

Meanwhile, the program should produce a file called `receipt.txt` which has the following contents:

```
******************************
*         Welcome to         *
*        Cafe Python!        *
******************************
coke....................$ 1.25
sandwich................$ 3.95

Subtotal................$ 5.20
Tax.....................$ 0.43
------------------------------
Total...................$ 5.63

Amount paid.............$10.00
Change due..............$ 4.37
```

Note that all of the formatting for the receipt has already been done correctly; it's just printed on the screen rather than output to a file. Note further that that all the calls to the built-in `print` function in the `main` function are necessary for the on-screen user/computer interaction. So you need to modify the functions and function calls so that they write to a file rather than the screen. You will need to pass an output file stream from the main function to each of the functions that are called (and to the functions that are called within those functions). The changes you need to make here are pretty minimal; the trick is to read and understand the original program before you start to make changes. Feel free to talk to me about your plan of action before you start modifying.

For this lab, you should submit the following files:

- `read_average_numbers.py` and the file of numbers you used to test it
- `joker_teller.py` and `jokes.txt`
- `cash_register.py` and the `receipt.txt` file you created by running it