CS 1301 – Spring 2010

Homework 6 – List & String Manipulation and File I/O

Due: Friday, March 5th, before 6PM
(10% late penalty if turned in before Monday, March 8th, before 6PM)

Scored out of 100 points
File to submit: hw6.py

This is a PAIR PROGRAMMING Assignment: Work with your partner!
For pair programming assignments, you and your partner should turn in identical assignments. Your submission must not be substantially similar to another pairs' submission. Collaboration at a reasonable level will not result in substantially similar code. Students may only collaborate with fellow students currently taking CS 1301, the TA's and the lecturer. Collaboration means talking through problems, assisting with debugging, explaining a concept, etc. You should not exchange code with or write code for other pairs.

For Help:

– TA Help Desk – Schedule posted on class website.
– Email the TAs

Notes:

- **Do not forget to include the required comments and collaboration statement as outlined on the course syllabus.**
- **You must name the file you create and the functions you write exactly as described in this assignment. Failure to do so will result in a major penalty.**
- Do not wait until the last minute to do this assignment in case you run into problems.
- If you find a significant error in the homework assignment, please let a TA know immediately.

Part I – List Manipulation

For this part of the assignment, your group will be writing four functions regarding list manipulation.

➔ *(5pts)* Define a function called *onlyFourths* that takes in a single parameter: a list of integers. The function should return a new list containing all of the numbers from the original list that are evenly divisible by four.

Examples:
onlyFourths([3,6,9,10]) should return [ ]
onlyFourths([3,4,5,6,7,8,9]) should return [4,8]

➔ *(5pts)* Define a function called *member* that takes in two parameters: a list and an item. It should return the boolean True if the item is in the list and False if the item is not in the list. It ***may not*** us the Python built "in" test. Instead it must compare the item to each element in the list.

Examples:
member([1,"16",True,6], "16") should return True
member([1,"16",True,6], 16) should return False

- ➔ *(10pts)* Define a function called *union* that takes in two parameters—both lists—and returns the union of the two lists. The function should not change any of the inputted lists. The elements in the returned list can be in any order, but there should be no repeats in the final list. Remember that the union of two lists is a list of all elements that appear in either list, without duplicates!
(Hint: The member function may help.)
Examples:
union([1,2,3,4,5,6], [4,5,6,7,8,9]) should return [1, 2, 3, 4, 5, 6, 7, 8, 9]
union([ ],[ ]) should return [ ]

- ➔ *(15pts)* Define a function called *flip* that takes in a single parameter—a list. The function should return a new list with the first and second elements switched, the third and fourth elements switched, etc. If there is an odd number of elements the last element should be left alone. The original list should not be changed.
Examples:
flip([1,2,3,4,5,6,7]) should return [2, 1, 4, 3, 6, 5, 7]

<u>Part II – String Manipulation</u>

For this part of the assignment, your group will be writing three functions regarding string manipulation.

- ➔ *(5pts)* Write a function called *stringMultiplication* that takes in a single parameter that is a string. The string will be constructed purely of integers (ex: “536345”). Your function should return the integer value representing the product of each (single digit) number in the string.
Examples:
stringMultiplication("54321") should return 120
stringMultiplication("22222”) should return 32

- ➔ *(15pts)* Write a function called *capitalize* that takes in no parameters. The function should prompt (ask) the user for a string and return a new string with the first letter of every word capitalized.
(Hint: it may be useful to use the *upper* function.)

- ➔ *(15pts)* Write a function called *parse* that takes in two parameters: a string and a delimiter. Your function should return a list of separated strings depending on the delimiter. You may assume the delimiter will be a single character. Each string in the list should not contain the delimiter itself. You **may not** use any built-in Python functions.
Examples:
parse("This is a Sentence", " ") should return ['This', 'is', 'a', 'Sentence']
parse("h-i-m-o-m", "-") should return ['h', 'i', 'm', 'o', 'm']
parse("f,.5,.5",",") should return ['f', '.5', '.5']

Part III – File I/O

For this part of the assignment, your group will be developing a function that can read a set of commands from a properly formatted text file and that can also make the robot execute each command properly. A sample text file (sample.txt) has been provided for your testing purposes. Please note that we will use our own different .txt file when grading, so make sure your code works very well for any combination of commands.

➔ *(30pts)* Define a function named *roboFile* that takes in one parameter: a string that represents a file name. Your function must open the file and read each line. Using the *parse* function that you wrote earlier, you must make the robot do some action according to each line in the text file.

  You may assume all lines in the text file are formatted as follows (notice that there are no spaces in the command):

  ```
  command,[ number1,number2 ]
  ```

  The "command" will be a single letter/word. The (optional) numbers will be either floating point or integers. You may parse all numbers as floating point, and then convert them to integers when needed.

  The following are the commands that we expect you to handle:

  - ✔ if the command is “f” then you want your robot to move forward for number1 seconds at number2 speed.
  - ✔ if the command is “b” then you want your robot to move backwards for number1 seconds at number2 speed.
  - ✔ if the command is “l” (the lowercase letter “L”) then you want your robot to turn left for number1 seconds at number2 speed.
  - ✔ if the command is “r” then you want your robot to turn right for number1 seconds at number2 speed.
  - ✔ if the command is “beep” then you want your robot to beep for number1 seconds at frequency number2.
  - ✔ if the command is “ir” then you want your robot to print the current value of the *getIR* function (the ir command is NOT followed by numbers!)
  - ✔ if the command is unrecognizable, then you want to print a message to the python shell that says the command is unrecognizable and continue to the next line.