Project 1

Recursion and Array-Based Sequences

Objective

This project is meant to help you develop skills at working with lists, strings, and recursion. You'll also learn to accept command-line arguments, read a text file, and record your program's execution time. In this project you will be provided with a project description, and list of program requirements, a sample input file, and some starting code (p1.py).

Project Description

Your program will read a text file composed of several lines, where each line is a sequence of digits, and rearrange each sequence so that all the even digits appear before all the odd digits.

Command Line Arguments

You must take the name of one text file as a command line argument. The following code checks that one argument was passed via the command line. It has been provided to you in the start file:

```
import sys

if len(sys.argv) != 2:
    raise ValueError('Please provide one file name.')

inputFileName = sys.argv[1]

print("\nThe file that will be used for input is " + inputFileName)
```

Read from a File

The input file will have many lines. Each line will be stored as a string in your program. Each string will be placed in a list, so that each item in the list will correspond to a line in the file.

1. Create a list (array-based sequence) of strings.
2. Read the text file.
3. Save each line as a string and add it to the list, in the order in which they are read from the file.

To open the file for reading, you must use the command:

```
f = open(inputFileName,"r")
```

Be careful, if you program can't find the file that it is looking for it will crash.

At this point you can read the lines from the file using:

```
myList = f.readlines()
```

This will read the whole file and store each line as an item in the list of strings *myList*.

Don't forget to close the file after you finish reading it:

```
f.close()
```

Process the List

In the main body of your program, loop through the list of strings and, for each item in the list, call the *arrange()* function passing that item as a parameter. The function will return the modified string, which you will print. You do not need to update the list of strings.

The arrange() Function

Write a **recursive** function *arrange()* that takes a string as a parameter. This string will be a sequence of digits only (but there may be a newline character (\n) at the end of the string). The function must rearrange the digits so that all the even digits appear before all the odd digits, and return the modified string. The *arrange()* function may take other parameters besides the string. No loops are allowed inside this function – it must use recursion.

Statistics

You need to record the total running time of your program. This time should include the time to read the text file, process each string, and print them. It is suggested that you start a timer as one of the first things you do in the program and stop it immediately before printing out the statistics. The following is a little sample code that shows how you can time events.

```
import time

start = time.time()
# ...
end = time.time()
print(end - start)
```

Sample Run

Here is a sample run of the program. A few notes:

1. I have the text file (digits.txt) and my python solution (p1.py) in the same folder (C:\Code). You may place them in your preferred folder.
2. Using the command prompt on Windows (terminal on MacOS), I run the program by typing the program name followed by the input text file. You may use an IDE instead if you prefer, such as VS Code, the Python's IDLE, or PyCharm.
3. When the program runs, it reads the input file and creates a list of strings.
4. Then loops through the list to modify and print each string.
    a. The order of the digits does not need to match the sample output – just make sure the even digits appear before the odd digits.
5. At the end, it prints the running time.
6. There is no user interaction.

```
C:\Code>python p1.py digits.txt

The file that will be used for input is: digits.txt

************************
*** Modified Strings ***
************************
1
2
2468462
08797531
24680753
2468097531
99975317531
280642435959573
420862860286153917
8420026875737791577
24242424243131313131
68866886687579757975

********************
*** Running Time ***
********************
Total Time of Program: 0.03655900 milliseconds
```

Handing in Your Project

Be sure to comment your code well so that it can be easily browsed to see what is going on. Code that is excessively difficult to read may be penalized.

Turn in a single file called "p1.py" to the assignment on Canvas.

Grading

The project will be graded out of 100 points and be graded on the following criteria:

- Code Organization
- Correct Output
- Performance (running time)

We will grade your code by running it against a series of test files and checking the output of the program against a known, correct implementation. More in depth tests may be run than the initial one provided. This means you should write some of your own test files to check for any errors.

We will also examine parts of your code to make sure that you are following the instructions listed in the requirements.