CSE/ISE 337 Assignment 1 – Python Programming (Spring 2019)

Due Date: Friday, February 22, 2019, at 11:55 PM

Instructions:

(a) When writing programs, you **must** use the techniques that are described in the lectures. You may **not** use methods, modules, packages that were not covered in this course.
(b) Start working on this assignment right away; you will find it very difficult to finish if you wait until the last day.
(c) Please use Python 3.
(d) Your code **must** run on a Computer Science department computer (either allv Linux machines or Windows 10 lab machines). Running only on your own computer will cost you up to 15 points.

1. (12 pts) This question concerns recursive functions.
(a) Consider the recursive function given during lecture to determine whether a string is a palindrome:

```
def pal_test(string):
    if len(string) <= 1:
        return True
    elif string[0] != string[-1]:
        return False
    else:
        return pal_test(string[1:-1])
```

Write a non-recursive version of this function. Then write a small Python program to demonstrate the correctness of your implementation.

(b) Consider the recursive function given during the lecture to compute factorial:

```
def factorial(n):
    if n > 1:
        return n * factorial(n-1)
    elif n == 1:
        return 1
    elif n == 0:
        return 1
```

First, write a distinct recursive version of this function which takes two arguments. The first argument, *n*, is the number for which we are computing the factorial function. The second argument, *ax*, is an *accumulator*. Use the accumulator, *ax*, to pass the current, running total for the factorial function to the next recursive call. Explain how the use of the accumulator changes how the recursive algorithm works.

Second, write a non-recursive version of this function.

Then write a small Python program to demonstrate the correctness of your implementations. Test your new implementations on large values (values large enough to cause recursion depth errors for the version of the factorial function we looked at in class) and see what happens. Try to explain the results.

2. (10 pts) Write a Python program to decode the following ciphertext (which is some encrypted English text) into plaintext (which is the original English text that we can understand):

```
"T qcura wthl rrscmc q yyew qsxgthhh drkjh F vtoyat bw qgvlhqnr pmdjrb,
rhz ymxd B fhqxftpt R qcura poqsiw xmx scmzvhdt uvut df vdsjhwfuhlcds
hvtzmdbp godulyt dn smx scmnbzx cefjbifnji. Gh mtp qrud vmbi lvlh 35000
khuxv sr vs smhbs jmxd, vqnm lxnfi knt rsiv fsxgthlhf, thz usrbnnyv, M
rdn's pjnw, i xfnyqb ktcth 35000 khuxv sf dmvpyi, wihluxp, fboko, tmxkc,
ezc bsymw qcnrysct. Jy e txqdwqbiw oefifhf eyovbhd, S kie e fjhulvyb
vrufrxh." --Wmxgthebtf
```

The following encryption scheme is used:

- The *n*th plaintext alphabetic letter, for all $n > 1$, is encrypted to the letter whose ordinal value is the ordinal value of the *n*th letter plus the ordinal value of the *n*-1th alphabetic letter in the message modulo 26.
- The first plaintext alphabetic letter of the message is encrypted to the letter whose position in the alphabet is the sum modulo 26 of the ordinal value of the first letter and the ordinal value of a randomly chosen alphabetic letter.
- For example, suppose the message is the string “I am not here right now”.
    - o The second plaintext alphabetic letter is “a”. It will be encrypted as the alphabetic letter whose position in the alphabet is the sum modulo 26 of the ordinal value of “a” [97] and the ordinal value of the preceding plaintext alphabetic letter, “I”, [73], which is the fourteenth letter of the alphabet “o” [(((97 + 73) % 26)) = 14] (if ‘a’ is the 0th letter of the alphabet).
    - o The first plaintext alphabetic letter, “I”, is encrypted as the alphabetic letter whose position in the alphabet is the sum modulo 26 of the ordinal value of “I” [73] and the ordinal value of a randomly chosen alphabetic letter. Suppose, you randomly generate “g” [103]. Then, “I” will be encrypted as the twentieth letter of the alphabet, “u” [(((73 + 103) % 26)) = 20].
- The randomly generated alphabetic letter must be appended to the ciphertext string. It will be needed for decryption.
- Capitalization is preserved by encryption and decryption. Any letter capitalized in the plaintext should be capitalized in the ciphertext, and vice versa.
- All non-alphabetic characters are unchanged.

You may hard code the ciphertext in your program. Output should be printed to stdout. Hints: You should first derive the decryption scheme that corresponds to the encryption scheme. In your program, consider using the `isalpha()`, `isupper()`, `islower()`, `chr()`, and `ord()` functions. Note that the double quotes are a part of the ciphertext string; think about how to enclose the string when you copy it into your program.

3. (8 pts) This is a question about lists

```
Column 1:              Column 2:                  Column 3:
>>> lst = [1, 2, 3]    >>> lst = [1, 2, 3]        >>> lst = [1, 2, 3]
>>> lst * 3            >>> arr = [ lst ] * 3      >>> arr = [ lst[:] ]* 3
>>> [ lst ] * 3        >>> lst[1] = 7             >>> arr[1][1] = 7
                       >>> arr # what do u see?   >>> arr # what do u see?
```

(a) Explain the difference between the second and third expressions in Column 1.

(b) On the second line of Column 2, we assign the result of the third expression of column 1 to the variable “arr”. If we then make a change to the content of “lst”, as in line 3 of Column 2, what happens to the value stored in “arr”? Explain the behavior. Explain what happens if we replace line 3 of Column 2 with “arr[1][1] = 7”?

(c) What if you tried to avoid the problem above using the slice copy, as in Column 3 above? What happens if you change an element in the array? Explain the result.

(d) Change line 2 in Column 3 so that “arr[0][1]” and “arr[2][1]” are not changed after line 3 of Column 3, “arr[1][1] = 7”

4. (8 pts) Given the following python program, determine the next few lines to be executed after line #7 if *n* and *r* are of the following values as the result of execution. Please label your answers clearly.

(a) if n = 0 and r = 0.
(b) if n = 1 and r = 2.
(c) if n = 2 and r = 0.
(d) if n = 3 and r = 3.

```
1   #!/usr/local/bin/python3
2   from random import randrange
3
4   for n in range(4):
5       print(n)
6       r = randrange(0,4)
7       print(r)
8       if n == r: continue
9       if n > r: break
10      print("x")
11  else:
12      print("Wow, you are lucky\n")
13
14  if n < 2:
15      print("Better luck next time\n")
```

5. (10 pts) This is a question about exceptions. Complete parts (a) and (b) for each of the following scenarios:
   i. Division by zero
   ii. Using an improper key with a dictionary
   iii. Indexing a list with an illegal value

(a) Find out what exception is produced by each of the following scenarios. Show how you find them out.

(b) Afterwards, write a small program for each scenario to illustrate how to catch the corresponding exception using a try statement and then continue with execution after the exception was raised.

6. (10 pts) The Linux shell command ls -lR > lsoutput.txt prints recursively all content in the filesystem tree starting and below the current directory (using a long listing format) to the standard output, which gets redirected to a file called lsoutput.txt that is in the current directory. Write a python program that processes an input file that contains some ls -R output and counts the number of entries in each depth of the filesystem tree, starting with the current directory. We assume that the current directory is at depth #1. As an example,
- if the current directory has four files and two subdirectories Dir1 and Dir2. Then at depth #1, there are six entries (four files plus two subdirectories).
- Assume there are five files in Dir1 with no subdirectories; and there are three files and one subdirectory (called A1) in Dir2. Then at depth #2, there are total nine entries (five from Dir1 plus four from Dir2).

- Further assume there are only two files in directory A1 and nothing else. Then at depth #3, there are two entries.
- Your program should then print {1: 6, 2: 9, 3: 2} to stdout correspondingly
- Your program should first ask the program user to enter the name of the input file.

7. (10pts) Functional programming.
(a) Write a Python function that takes two parameters as input: a list of integers and the even() function:

```
def even(x):
    return x % 2 == 0
```

Using list comprehension, the function returns a list that contains the positions of the integers in the input list for which even() returns true. E.g., if the input list is a=[2,3,4,5,6,8], and your function is called listIndices(), then calling listIndices(a, even) would return [0,2,4,5]. Write a small program to demonstrate the correctness of your implementation.

(b) Write a Python function that takes two parameters as input: a list of integers and the even() function defined above. The function returns true if even() returns true for all integers in the input list. Otherwise, the function returns false. E.g., if the input list is [2,4,6,8], the output would be true; if the input list is [3,4,6,8], the output would be false. Your function must use at least one of the three functions: map(), filter(), and reduce(). Write a small program to demonstrate the correctness of your implementation.

(c) What does the following Python expression do? Using annotations, demonstrate your complete understanding of the expression; explain what each part of the expression does.

```
[i for i in range(2,n) if len([x for x in range(2,i) if i%x==0]) == 0]
```

Submission Format (2 pts for the right submission format):
Submit a "a1.zip" file that includes the following:
-- A README.txt or README.pdf file. It should contain your name, ID, and CS machine used (Linux allv machines or Windows 10 lab machines). It should also contain the answers to all questions that require a written response. Please label each of your answers clearly with the question number and part for which they are a response.
-- All programs that you write. Each program should be in a separate .py file. Please name each program file using its question number and part. For example, the program for your non-recursive palindrom function should be written in a file named "1a.py".

Total Points: 70

Submission Instructions:
Assignments will be turned in through Blackboard. Make sure that your submission is correctly formatted before you turn it in, and that submission occurs before the deadline.

You can only submit twice. Only click "Submit" when you are sure you have followed all the submission requirements.

Late submissions will not be accepted. The due date is **11:55 PM, Friday, February 22.**