Python Programming Assignment # 4

Instructions

For each of the following problems, create an error free efficient Python program including a main function calling other functions. Each program should be submitted in a separate Python file respectively that follows a particular naming convention. (E.g. The Python program for Question 1 should be in .py file with name Assign4Answer1.py. The Python program for question 2 should be in .py file with name Assign4Answer2.py. The program should execute properly in PyCharm)

Please, make sure that you have PythonLabs properly installed and it is properly configured in PyCharm. Submit your assignment by Monday 12 November 2018 EoD.

Problems

Problem 1: List of Gray codes of a specified length (10 points)

Write a function that returns a list of Gray codes of a specified length. A Gray code is a special type of binary code organized so that any two successive bit patterns differ by only one bit. For example, a 2-bit Gray code is ['00', '01', '11', '10']. Notice how in the transition from the second code to the third code only the first bit in the code changes. Here is a recursive algorithm for generating an n-bit Gray code:

• If n = 1 return the list with two strings ['0','1']

• Otherwise let a be the list of Gray codes with n - 1 bits; the output should be a list of codes with a 0 in front of every code in a followed by codes that have a 1 in front of every code in a but in reverse order.

Here is an example that shows what your function should produce for n = 3:

>>> graycode(3)

['000', '001', '011', '010', '110', '111', '101', '100']

Notice how the first four codes have 0's in front of the four 2-bit codes and the last four codes have a 1 in front of the 2-bit codes in the opposite order. Some hints:

• The + operator can be used to concatenate two strings and to append two lists

• A built-in function named reversed will invert the order of items in a list.

Problem 2: Merge sort deck of playing cards (10 points)

An ordered deck of cards is as follows:

orderedDeck = ["2C", "3C", "4C", "5C", "6C", "7C", "8C", "9C", "10C", "JC", "QC", "KC", "AC", "2D", "3D", "4D", "5D", "6D", "7D", "8D", "9D", "10D", "JD", "QD", "KD", "AD", "2H", "3H", "4H", "5H", "6H", "7H", "8H", "9H", "10H", "JH", "QH", "KH", "AH", "2S", "3S", "4S", "5S", "6S", "7S", "8S", "9S", "10S", "JS", "QS", "KS", "AS"]

A deck of cards could shuffled randomly using function random.shuffle. Modify msort2.py program given in Lecture6-Examples such that given a randomly shuffled deck of cards, it sorts the deck of cards as per orderedDeck above.

Tip: Suitably modify the comparison criteria in merge function to determine the order of a particular card.

Problem 3: Sorting a stack (10 points)

Given a stack, sort it using **recursion**. Use of any loop constructs like while, for.. etc is not allowed. You have to create and only use the following functions on Stack S:

is_empty(S) : Tests whether stack is empty or not.
push(S, element) : Adds new element to the top of the stack.
pop(S) : Removes top element from the stack. (this function is different than Python list.pop function).
top(S) : Returns value of the top element. Note that this function does not remove element from the stack.

Example of input and output stack:

Input: -3 <--- Top 14 18 -5 30	Output: 30 <--- Top 18 14 -3 -5

The idea of the solution is to hold all values in Function Call Stack until the stack becomes empty. When the stack becomes empty, insert all held items one by one in sorted order. Here sorted order is important.

We can use below algorithms (this is not Python code) to solve the problem:

Algorithm to sort stack elements:	**Algorithm is to insert element is sorted order:**

sortStack(stack S)
  if stack is not empty:
  temp = pop(S);
  sortStack(S);
  sortedInsert(S, temp);
	sortedInsert(Stack S, element)
	  if stack is empty OR element > top element
	  push(S, element)
	  else
	  temp = pop(S)
	  sortedInsert(S, element)
	  push(S, temp)

- Create a random list of 20 numbers where element at index 0 is top of the stack.
- Create a Python implementation of stack functions described above and implement sortStack and sortedInsert algorithms.
- Sort created list using sortStack function.