

Assignment 7

Due Tuesday, November 2nd, 5:45 pm

Reading

Read Chapter 10 in **Introduction to Computing using Python: An Application Development Focus** by Ljubomir Perković.

Logistics

In this class programming assignments may be completed in consultation with up to two other classmates. You must identify the classmates with whom you collaborate in a comment at the top of the assignment, and the number of collaborators on any assignment **may not exceed two other people**. You must also submit a comment in your submission for each assignment that describes in detail how each collaborator contributed to the assignment. If you did not collaborate with anyone on the assignment, you must include a comment that says that. You may not under any circumstances discuss the assignments with classmates other than your identified collaborators. Working so closely with anyone other than your identified collaborators, Mr. Zoko or the teaching assistant, so as to produce identical or near identical code is a violation of the Academic Integrity policy. This policy will be strictly enforced.

Please include the following with your assignment submission:

1. A comment at the top of your Python file identifying any classmates with whom you discussed or in any other way collaborated on the assignment. You may work (directly or indirectly) with **no more than two** other people.
2. Add a comment at the top of your Python file that describes for each person what they contributed to the assignment. This must be at least 2-3 sentences and be **very specific and detailed**.

A submission that does not include a list of collaborators and a comment indicating how you collaborated with classmates will earn a 0. If you worked alone, you must put a comment at the top of your file that indicates that or you will also receive a 0. There will be no exceptions to this rule.

Again, you are subject to all the rules specified in the Academic Integrity policy. Please read it carefully before beginning this assignment.

Assignment

Please remember that you are not allowed to consult online resources when completing homework assignments. If you have questions about this assignment, please contact me.

Implement the functions below in a file called **csc242hw7.py** a template for which has been provided on the D2L site.

The functions written for this assignment must be **recursive (Problems 1-5)** and **must not** use global variables. Do not modify the function names or parameters in the template file. The functions **may not use loops**. In some cases, certain built-in Python functions are disallowed. Please read the problem description carefully. Functions that are described as returning values should not do any printing. Solutions that do not follow these guidelines will not earn full credit, even if they produce the correct results in all cases.

Please note, local variables are absolutely fine in any of the functions.

1. Write a recursive function **recLessThan()** that takes a an integer and **prints** a “greater than symbol” pattern using the character ‘#’. The parameter represents the largest indent of the pattern. If n is 0 or negative the function doesn't print anything. The following shows several examples of patterns using different values of n.

```
>>> recLessThan (5)
```

```
    #
```

```
   #
```

```
  #
```

```
 #
```

```
#
```

```
#
```

```
 #
```

```
  #
```

```
   #
```

```
    #
```

```
>>>
```

```
>>> recLessThan (7)
```

```
    #
```

```
   #
```

```
  #
```

```
 #
```

```
#
```

```
#
```

```
#
```

```
#
```

```
 #
```

```
  #
```

```
   #
```

```
    #
```

```
   #
```

```
    #
```

```
.
```

2. Write a recursive function **recHourGlassShape()** that takes a character and two integers *n* and *indent* as parameters and **prints** an hour glass pattern using the character. The number of characters in the top line of the pattern and in the bottom line of the pattern (i.e. the biggest lines for each of the characters) is *n*. The *indent* parameter represents the indentation used in the first line of the pattern and in the last line of the pattern. The indentation increases in the top triangle (using the first character) and decreases in the bottom triangle (using the second character). If *n* is 0 or negative or one of the characters is the empty string, the function doesn't print anything. You should assume that the *indent* parameter will be non-negative (i.e. ≥ 0). The following shows several examples of patterns using different characters, values of *n*, and amount of indentation.

```
>>> recHourGlassShape ('#', 8, 8)
```

```
#####  
#####  
####  
##  
##  
####  
#####  
#####
```

```
>>> recHourGlassShape ('$ ', 4, 8)
```

```
$$$  
$  
$  
$$$
```

3. Write a recursive function **recAbbrev()** that takes a one-dimensional list of strings as a parameter and **returns** the abbreviation of those strings capitalized. You can assume the list of has strings. The only string functions you are allowed to use are `len()`, indexing (`lst[i]` for an integer `i`), or slicing (`lst[i:j]` for integers `i` and `j`). In particular, the function should not in any way alter the list passed as a parameter. The following shows several sample runs of the function. You should test with other inputs as well.:

```
////
>>> recAbbrev ( [])
''
>>> recAbbrev ( ['central', 'processing', 'unit'])
'CPU'
>>> recAbbrev ( ['ramdom', 'access', 'memory'])
'RAM'
```

4. Write a recursive function **recEvenNumbers** () that takes a list as a parameter and **returns** a count of how many even numbers were found in the list. Recall that you can determine whether an item is a number by checking `type(item) == int` or `float`. The only list functions you are allowed to use are `len()`, indexing (`lst[i]` for an integer `i`), or slicing (`lst[i:j]` for integers `i` and `j`). The following shows several sample runs of the function. You should test with other inputs as well.:

```
>>> recEvenNumbers(['c', 'a', 't', ' ', 0, 'h', 1, 'a', 2, 't', 100])
3
>>> recEvenNumbers(['c', 4, 8, 'a', 10, 1, 't', 20])
4
>>> recEvenNumbers([])
0
```

5. Write a recursive **recMerge** that takes two strings and **returns** a single string that is a combination of all the strings with the values ordered roughly the same way as in the original strings but in one string. The list must be traversed and combined recursively, or no points will be given. The following shows several sample runs of the function. You should test with other inputs as well.:

```
>>> recMerge('', '')
''
>>> recMerge('Atoy', 'nhn')
'Anthony'
>>> recMerge('ti a u', 'hswsfn')
'this was fun'
```

Submitting the assignment

You must submit the assignment using the assignment 7 dropbox on [the D2L site](#). Submit a Python file (csc242hw7.py) with your implementation in it and comments describing your collaboration status. Submissions after the deadline listed above will be automatically rejected by the system. See the syllabus for the grading policy.

Grading

The assignment is worth 100 points. Each recursion question is worth 20 points. Any student who does not submit comments in the Python file describing the contributions of each team member or indicating that he/she worked alone will earn a 0 on the assignment.