Tutorial 06 Recursion

1. We can define the sum of the numbers from 1 to 𝑥 (i.e. 1 + 2 + . . . + 𝑥) recursively as follows (for integer 𝑥 ≥ 1):
   * 1 if 𝑥 = 1
   * 𝑥 + sum of the numbers from 1 to 𝑥 − 1 if 𝑥 > 1

Based on the above definition, complete the following Python program to compute the sum 1 + 2 + 3 + 4 + 5 + 6 + 7 + 8 + 9 + 10 recursively.

**def** main():

*# Compute and print the sum of (1 + 2 + ... + 10)*

print( sum(10) )

**def** sum(x):

*# Assuming x >= 1*

*# Complete this function recursively*

main()

COMPLETE THE CODES HERE

Graphical user interface, text, application

Description automatically generated

1. Write a recursive Python function – sumDigits(n), that takes a positive integer n and returns the sum of all its digits. For example, sumDigits(368) will return the number

3 + 6 + 8 = 17.

Graphical user interface, text, application, email

Description automatically generated

1. Consider the following Python program:

**def** main():

y = foo( 4 ) bar( 2 )

**def** foo( x ):

**if** x % 2 != 0:

**return** 0

**else**:

**return** x + foo( x-1 )

**def** bar( n ):

**if** n > 0:

bar( n-1 ) print( n )

main()

1. What is the output of the program?

***1***

***2***

1. Draw the recursive call tree for the program.

return 4 + 0

main()

return 0

foo(4)

bar(2)

print(2)

bar(1)

foo(3)

print(1)

bar(0)

\*just in case if the formatting change, this pic is for backup

Graphical user interface, diagram

Description automatically generated

1. A palindrome is a word, phrase, or sequence that reads the same backwards as forwards, e.g. level, madam, noon, “don’t nod”, “top spot”.
2. Design and implement a recursive Python function – isPalindrome(aStr), for determining whether a string of characters - aStr, is a palindrome.

[**HINTS**: Note that a string with one or fewer characters is a palindrome. Possible base case? What about the recursive case? And how to ensure the recursive function makes progress towards the base case?]

Graphical user interface, text, application

Description automatically generated

1. Draw the recursive call tree for the isPalindrome(aStr) function when called with a string – madam.

return True

isPalindrome(‘madam’)

return True

isPalindrome(‘ada’)

return True

isPalindrome(‘d’)

\*\*just in case if the formatting changes the structure, the below is the picture of my diagram

Diagram

Description automatically generated

1. The exponential function 𝑥𝑛 can be expressed as 𝑥 multiplied by itself 𝑛 times. For example, 28 would be computed as 2 ∗ 2 ∗ 2 ∗ 2 ∗ 2 ∗ 2 ∗ 2 ∗ 2.
2. Write a non-recursive Python function – exp(x,n), that takes two non-negative integers x and n, and returns the value 𝑥𝑛. For example, exp(2,8) will return the number 256.

Graphical user interface, text, application

Description automatically generated

1. Now using recursion, write a recursive Python function – exp\_recursive(x,n), that takes two non-negative integers x and n, and returns the value 𝑥𝑛. For example, exp\_recursive(2,8) will return the number 256.

Graphical user interface, text, application

Description automatically generated

***-- End of Tutorial --***