Tutorial 06
Recursion

1. We can define the sum of the numbers from 1 to x (i.e. $1 + 2 + \ldots + x$) recursively as follows (for integer $x \geq 1$):

   - 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
    COMPLETE THE CODES HERE

main()
```

2. 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`.

3. 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()
```

   a. What is the output of the program?

   b. Draw the recursive call tree for the program.

4. 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”.

   a. 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?]

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

5. The exponential function x^n can be expressed as x multiplied by itself n times. For example, 2^8 would be computed as $2 * 2 * 2 * 2 * 2 * 2 * 2 * 2$.

   a. Write a non-recursive Python function – `exp(x,n)`, that takes two non-negative integers `x` and `n`, and returns the value x^n. For example, `exp(2,8)` will return the number `256`.

   b. 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 x^n. For example, `exp_recursive(2,8)` will return the number `256`.

-- End of Tutorial --