ASSIGNMENT 9

1.Given an integer `n`, return *`true` if it is a power of two. Otherwise, return `false`.*

An integer `n` is a power of two, if there exists an integer `x` such that `n == 2x`.

```
def is_power_of_two(n):
    if n == 1:
        return True
    elif n < 1 or n % 2 != 0:
        return False
    else:
        return is_power_of_two(n // 2)
```

2. Given a number n, find the sum of the first natural numbers.

```
def sum_of_natural_numbers(n):
    if n == 1:
        return 1
    else:
        return n + sum_of_natural_numbers(n - 1)
```

3. Given a positive integer, N. Find the factorial of N.

```
def factorial(n):
    if n == 0 or n == 1:
        return 1
    else:
        return n * factorial(n - 1)
```

4. Given a number N and a power P, the task is to find the exponent of this number raised to the given power, i.e. N^P.

```
def power(N, P):
    if P == 0:
        return 1
    else:
        return N * power(N, P - 1)
```

5. Given an array of integers **arr**, the task is to find maximum element of that array using recursion.

```
def find_max(arr):
    if len(arr) == 1:
        return arr[0]
    else:
        return max(arr[0], find_max(arr[1:]))
```

6. Given first term (a), common difference (d) and a integer N of the Arithmetic Progression series, the task is to find Nth term of the series.

```
def nth_term_of_ap(a, d, N):
    if N == 1:
        return a
    else:
        return nth_term_of_ap(a, d, N - 1) + d
```

7. Given a string S, the task is to write a program to print all permutations of a given string.

```
def generate_permutations(S, l, r):
    if l == r:
        print(''.join(S))
    else:
        for i in range(l, r + 1):
            S[l], S[i] = S[i], S[l]  # Swap characters
            generate_permutations(S, l + 1, r)  # Recursive call
            S[l], S[i] = S[i], S[l]  # Backtrack (undo the swap)

def permutations(S):
    n = len(S)
    generate_permutations(list(S), 0, n - 1)
```

8. Given an array, find a product of all array elements.

```
def product_of_array(arr):
    if len(arr) == 1:
        return arr[0]
    else:
        return arr[0] * product_of_array(arr[1:])
```