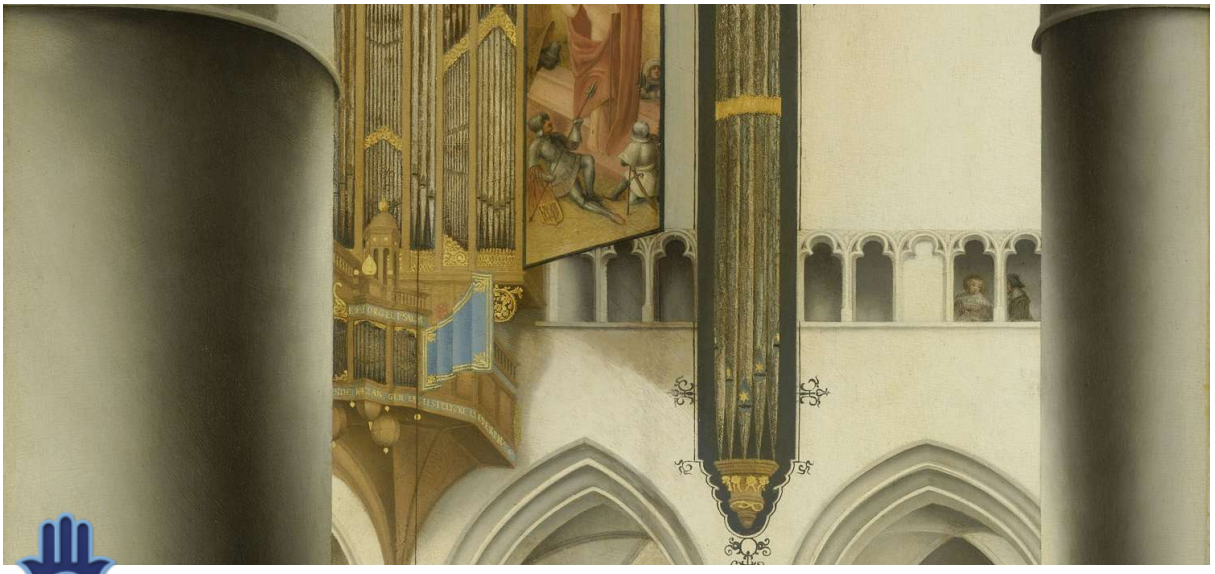

# Day 13:  Recursive Function

## Task 1:

### Factorial of a Number using Recursion

Define a function `factorial(n)` that returns the factorial of a number using recursion.

Example:

```
factorial(5)  # Output: 120
```

## Task 2:

### Fibonacci Sequence using Recursion

Define a function `fibonacci(n)` that returns the nth number in the Fibonacci sequence using recursion.

Example:

```
fibonacci(6)  # Output: 8
```

## Task 3:

### Check if a String is a Palindrome

Write a recursive function `is_palindrome(s)` that checks if a string is a palindrome.

Example:

```
is_palindrome("madam")  # Output: True
is_palindrome("hello")  # Output: False
```

---

## Task 4:

### Print Elements from a List Using Recursion

Define a function `print_elements(lst)` that prints each element of a list recursively.

Example:

```
print_elements([1, 2, 3, 4])
# Output:
# 1
# 2
# 3
# 4
```

---

## Task 5:

### Sum of First N Natural Numbers

Define a function `sum_of_natural_numbers(n)` that returns the sum of the first n natural numbers using recursion.

Example:

```
sum_of_natural_numbers(5)  # Output: 15
```

---

## Task 6:

### Reverse a String Using Recursion

Write a recursive function `reverse_string(s)` that takes a string and returns its reverse.

Example:

```
reverse_string("hello")  # Output: "olleh"
```

---

## Task 7:

### Count the Number of Digits in a Number

Define a recursive function `count_digits(n)` that counts the number of digits in a number.

Example:

```
count_digits(12345)  # Output: 5
```

---

## Task 8:

### Sum of Digits of a Number

Define a recursive function `sum_of_digits(n)` that returns the sum of the digits of a number.

Example:

```
sum_of_digits(123)  # Output: 6
```

---

## Task 9:

### Find the Maximum Element in a List Using Recursion

Write a recursive function `find_max(lst)` that finds the maximum element in a list.

Example:

```
find_max([3, 1, 4, 1, 5, 9])  # Output: 9
```

---

## Task 10:

### Exponentiation Using Recursion

Write a recursive function `power(base, exp)` that calculates the result of raising `base` to the power of `exp`.

Example:

```
power(2, 3)  # Output: 8
```

---