Task-1

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

Task-2

```
def fibonacci(n):
    if n <= 1:
        return n
    else:
        return fibonacci(n-1) + fibonacci(n-2)
```

Task-3

```
def print_array_recursive(arr, index=0):
    if index < len(arr):
        print(arr[index])
        print_array_recursive(arr, index + 1)

# Example usage:
my_array = [1, 2, 3, 4, 5]
print_array_recursive(my_array)
```

Task-5

```
def power_recursive(m, n):
    if n == 0:
        return 1
    else:
        return m * power_recursive(m, n - 1)

# Example usage:
m = 5
n = 4
result = power_recursive(m, n)
print(result)
```

Searching and Sorting

1)

```
def selection_sort(arr):
    n = len(arr)
    for i in range(n-1):
        # Find the index of the minimum element in the unsorted part of the array
        min_index = i
        for j in range(i+1, n):
            if arr[j] < arr[min_index]:
                min_index = j

        # Swap the found minimum element with the first element
        arr[i], arr[min_index] = arr[min_index], arr[i]
```

```
# Example usage:
my_array = [64, 25, 12, 22, 11]
selection_sort(my_array)
print("Sorted array:", my_array)
```

2)

```
def recursive_insertion_sort(arr, n=None):
    if n is None:
        n = len(arr)

    # Base case: If the array has only one or zero elements, it is already sorted.
    if n <= 1:
        return

    # Sort the first n-1 elements recursively
    recursive_insertion_sort(arr, n-1)

    # Insert the last element at its correct position in the sorted array
    key = arr[n-1]
    j = n-2
    while j >= 0 and arr[j] > key:
        arr[j+1] = arr[j]
        j -= 1
    arr[j+1] = key

# Example usage:
my_array = [64, 25, 12, 22, 11]
recursive_insertion_sort(my_array)
print("Sorted array:", my_array)
```

3)

```
def binary_search_recursive(arr, low, high, target):
    if low <= high:
        mid = (low + high) // 2

        # Check if the target is present at the middle
        if arr[mid] == target:
            return mid

        # If the target is smaller, search in the left half
        elif arr[mid] > target:
            return binary_search_recursive(arr, low, mid - 1, target)

        # If the target is larger, search in the right half
        else:
            return binary_search_recursive(arr, mid + 1, high, target)

    # If the target is not present in the array
    return -1

# Example usage:
my_array = [2, 3, 4, 10, 40]
target_element = 10
result = binary_search_recursive(my_array, 0, len(my_array) - 1, target_element)

if result != -1:
    print(f"Element {target_element} is present at index {result}.")
else:
    print(f"Element {target_element} is not present in the array.")
```