**Classwork 5**

**Classwork\_recursion**

1)

def factorial(n):  
 if n == 1 or n == 0:  
 return 1  
 else:  
 return n \* factorial(n-1)  
n = int(input())  
print(factorial(n))

2)

def fibonacci(n):  
 if n == 0:  
 return 0  
 elif n == 1:  
 return 1  
 else:  
 return fibonacci(n-2) + fibonacci(n-1)  
n = int(input())  
print(fibonacci(n))

3)

def digits\_sum(n):  
 if n < 10:  
 return n  
 else:  
 return n % 10 + digits\_sum(n//10)  
n = int(input())  
print(digits\_sum(n))

4)

def power(base, exponent):  
 if exponent == 0:  
 return 1  
 else:  
 return base \* power(base, exponent-1)  
base = int(input("Enter a base: "))  
exponent = int(input("Enter a exponent: "))  
print(power(base, exponent))

5)

def reversed\_str(string):  
 if len(string) == 1:  
 return string  
 else:  
 return string[-1] + reversed\_str(string[:-1])  
string = input("Enter a string: ")  
print(reversed\_str(string))

6)

def binary\_search(arr, target, left, right):  
 if left > right:  
 return -1  
 mid = (left+right) // 2  
 if arr[mid] == target:  
 return mid  
 elif arr[mid] < target:  
 return binary\_search(arr, target, mid, right)  
 else:  
 return binary\_search(arr, target, left, mid)  
arr = [1,2,3,4,5,6,7,8]  
target = int(input())  
print(binary\_search(arr, target, 0, len(arr)-1))

7)

def gcd(x, y):  
 if y == 0:  
 return x  
 else:  
 return gcd(y, x % y)  
x = int(input())  
y = int(input())  
print(gcd(x, y))

8)

def hanoi(n, first, second, third):  
 if n > 0:  
 hanoi(n - 1, first, third, second)  
 print(f"Move disk {n} from {first} to {second}")  
 hanoi(n - 1, third, second, first)  
  
# Example usage:  
n = 3  
hanoi(n, 'A', 'C', 'B')

9)

def palindrome(x):  
 if len(x) <= 1:  
 return "Yes"  
 if x[0] == x[-1]:  
 return palindrome(x[1:-1])  
 else:  
 return "No"  
x = input()  
print(palindrome(x))

10)

def flatten(nested):  
 flattened = []  
 for i in nested:  
 if isinstance(i, list):  
 flattened.extend(flatten(i))  
 else:  
 flattened.append(i)  
 return flattened  
nested = [1, [2, [3, 4]], 5, [6, [7, 8, [9, 10]]]]  
print(flatten(nested))

11)

num1 = float(input())  
def subsets(orig\_set):  
 if not orig\_set:  
 return [orig\_set]  
 first = orig\_set.pop()  
 rest = subsets(orig\_set)  
 return rest + [subset.union({first}) for subset in rest]  
orig\_set = {1,2,3}  
print(subsets(orig\_set))