

20) Write C programs for solving recurrence relations using the Master Theorem, Substitution Method, and Iteration Method will demonstrate how to calculate the time complexity of an example recurrence relation using the specified technique.

CODE:

```
def master_theorem(a, b, k):
    if a > b**k:
        return "O(n^log_b(a))"
    elif a == b**k:
        return "O(n^k * log(n))"
    else:
        return "O(n^k)"

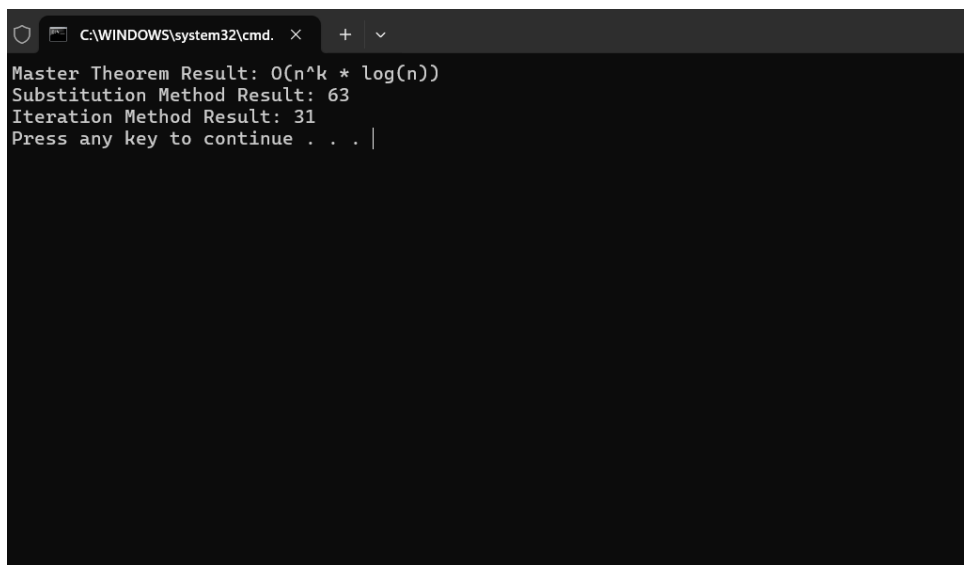
def substitution_method(t,n):
    if n == 0:
        return 1
    else:
        return 2 * substitution_method(t,n-1) + 1

def iteration_method(t,n):
    result = 0
    for i in range(n):
        result += 2**i
    return result

a = 2
b = 2
k = 1
t=2
n=5
master_theorem_result = master_theorem(a, b, k)
substitution_method_result = substitution_method(t,n)
iteration_method_result = iteration_method(t,n)

print("Master Theorem Result:", master_theorem_result)
print("Substitution Method Result:", substitution_method_result)
print("Iteration Method Result:", iteration_method_result)
```

OUTPUT:

A screenshot of a Windows command prompt window. The title bar shows the path 'C:\WINDOWS\system32\cmd.' and standard window controls. The command prompt displays the output of the Python program: 'Master Theorem Result: O(n^k \* log(n))', 'Substitution Method Result: 63', and 'Iteration Method Result: 31'. Below these results, it shows 'Press any key to continue . . . |' with a cursor. The background of the command prompt is black, and the text is white.

TIME COMPLEXITY :  $O(n \log n) + O(2n) + O(n)$