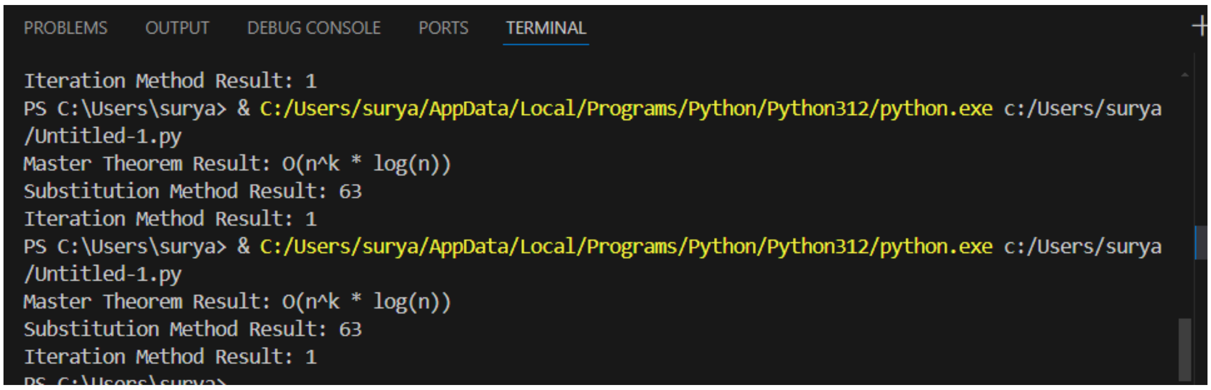
20.Write C programs for solving recurrence relations using the Master Theorem, Substitution Method, and Iteration Method will demonstrate how to calculate the time complexity of an example recurrence relation using the specified technique.

PROGRAM:

```
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:



```
Iteration Method Result: 1
PS C:\Users\surya> & C:/Users/surya/AppData/Local/Programs/Python/Python312/python.exe c:/Users/surya
/Untitled-1.py
Master Theorem Result: O(n^k * log(n))
Substitution Method Result: 63
Iteration Method Result: 1
PS C:\Users\surya> & C:/Users/surya/AppData/Local/Programs/Python/Python312/python.exe c:/Users/surya
/Untitled-1.py
Master Theorem Result: O(n^k * log(n))
Substitution Method Result: 63
Iteration Method Result: 1
PS C:\Users\surya>
```

TIME COMPLEXITY:

Time complexity for the above code is

F(n)=O(nlogn)+O(2n)+O(n)