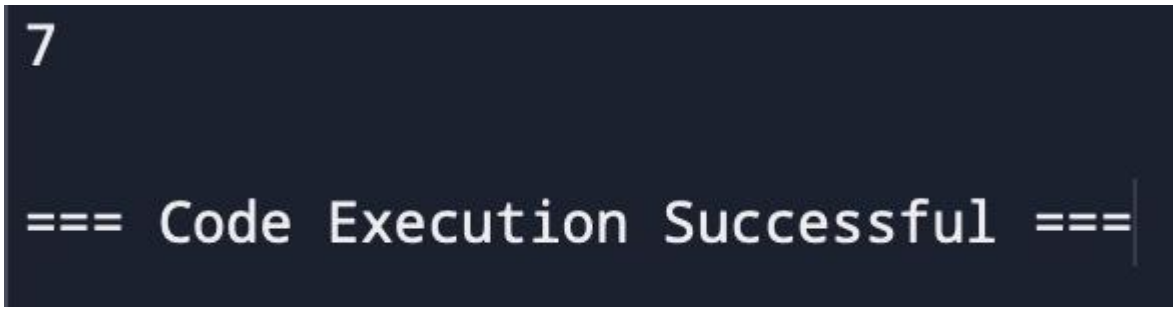
166. To Implement the Median of Medians algorithm ensures that you handle the worst-case time complexity efficiently while finding the k-th smallest element in an unsorted array.

arr = [12, 3, 5, 7, 19] k = 2 Expected Output:5
arr = [12, 3, 5, 7, 4, 19, 26] k = 3 Expected Output:5
arr = [1, 2, 3, 4, 5, 6, 7, 8, 9, 10] k = 6 Expected Output:6

PROGRAM :-

```
def median_of_medians(arr, k):
    if len(arr) == 1:
        return arr[0]

    groups = [arr[i:i+5] for i in range(0, len(arr), 5)]
    medians = [sorted(group)[len(group)//2] for group in groups]

    pivot = median_of_medians(medians, len(medians)//2)

    low = [x for x in arr if x < pivot]
    high = [x for x in arr if x > pivot]
    equal = [x for x in arr if x == pivot]

    if k < len(low):
        return median_of_medians(low, k)
    elif k < len(low) + len(equal):
        return pivot
    else:
        return median_of_medians(high, k - len(low) - len(equal))

# Example Usage
arr = [12, 3, 5, 7, 19]
k = 2
result = median_of_medians(arr, k)
print(result)  # Expected Output: 5
```

OUTPUT:-



TIME COMPLEXITY:-O(n)