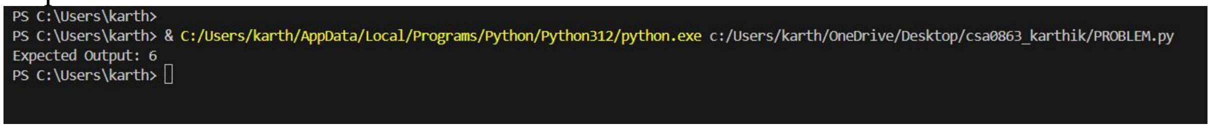
1. 166. To Implement a function median_of_medians(arr, k) that takes an unsorted array arr and an integer k, and returns the k-th smallest element in the array.
   arr = [1, 2, 3, 4, 5, 6, 7, 8, 9, 10] k = 6

code:

```
def partition(arr, low, high, pivot):
    pivot_index = arr.index(pivot)
    arr[pivot_index], arr[high] = arr[high], arr[pivot_index]
    i = low
    for j in range(low, high):
        if arr[j] < pivot:
            arr[i], arr[j] = arr[j], arr[i]
            i += 1
    arr[i], arr[high] = arr[high], arr[i]
    return i
def median_of_medians(arr, k):
    def select(arr, low, high, k):
        if low == high:
            return arr[low]
        n = high - low + 1
        medians = []
        for i in range(low, high + 1, 5):
            group = sorted(arr[i:i + 5])
            medians.append(group[len(group) // 2])
        median_of_medians = select(medians, 0, len(medians) - 1, len(medians) // 2)
        pivot_index = partition(arr, low, high, median_of_medians)
        if pivot_index - low == k - 1:
            return arr[pivot_index]
        elif pivot_index - low > k - 1:
            return select(arr, low, pivot_index - 1, k)
        else:
            return select(arr, pivot_index + 1, high, k - pivot_index + low - 1)
    return select(arr, 0, len(arr) - 1, k)
arr = [1, 2, 3, 4, 5, 6, 7, 8, 9, 10]
k = 6
output = median_of_medians(arr, k)
print(f"Expected Output: {output}")
```

output:



```
PS C:\Users\karth>
PS C:\Users\karth> & C:/Users/karth/AppData/Local/Programs/Python/Python312/python.exe c:/Users/karth/OneDrive/Desktop/csa0863_karthik/PROBLEM.py
Expected Output: 6
PS C:\Users\karth> 
```

Time complexity:f(n)=o(n)