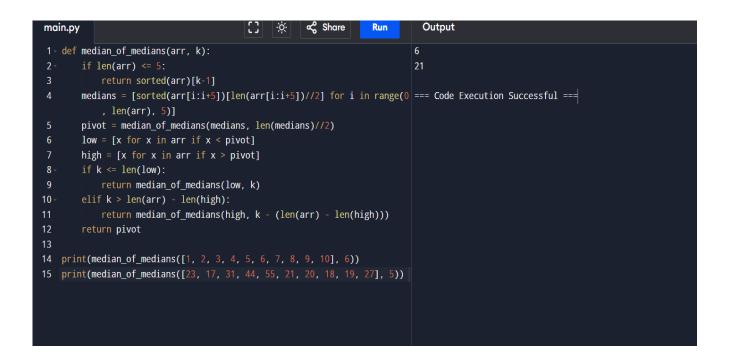
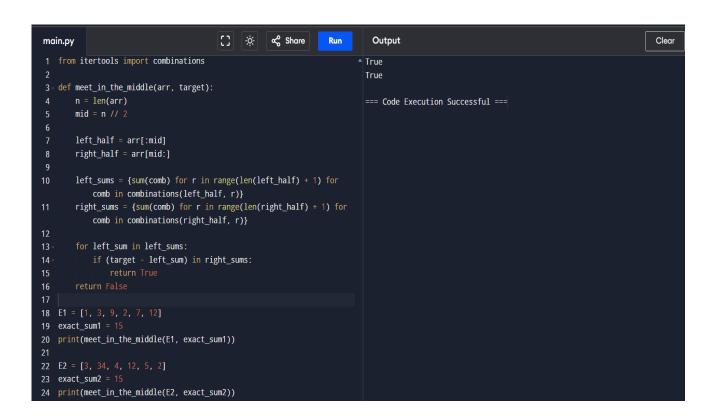
```
[] ×
                                                              ∝ Share
                                                                             Run
                                                                                         Output
                                                                                                                                                                     Clear
main.py
 1 def median_of_medians(arr, k):
              return sorted(arr)[k-1]
         medians = [sorted(arr[i:i+5])[len(arr[i:i+5])//2] for i in range(0 === Code Execution Successful ===
             , len(arr), 5)]
         pivot = median_of_medians(medians, len(medians) // 2)
         low = [x for x in arr if x < pivot]</pre>
         high = [x for x in arr if x > pivot]
         k_index = len(low)
         if k_index == k - 1:
             return pivot
         elif k_index > k - 1:
14
             return median_of_medians(low, k)
16
             return median_of_medians(high, k - k_index - 1)
19 print(median_of_medians([12, 3, 5, 7, 19], 2))
20 print(median_of_medians([12, 3, 5, 7, 4, 19, 26], 3))
21 print(median_of_medians([1, 2, 3, 4, 5, 6, 7, 8, 9, 10], 6))
```



```
Output
                                                    ∝ Share
                                                                Run
                                                                                                                                            Clear
main.py
                                        [] 🔅
1 from itertools import chain, combinations
                                                                        Closest subset for set1: (34, 7)
                                                                         Closest subset for set2: (0, 10)
3 def subsets(arr):
       return chain(*map(lambda x: combinations(arr, x), range(0, len
                                                                         === Code Execution Successful ===
           (arr)+1)))
6 def closest_sum(arr, target):
       n = len(arr)
       half = n // 2
       left_half = list(subsets(arr[:half]))
       right_half = list(subsets(arr[half:]))
       left_half_sums = [sum(subset) for subset in left_half]
       right_half_sums = [sum(subset) for subset in right_half]
       closest = float('inf')
       closest_subset = None
       for left_sum in left_half_sums:
           for right_sum in right_half_sums:
               current_sum = left_sum + right_sum
               if abs(target - current_sum) < abs(target - closest):</pre>
                  closest = current_sum
                   closest_subset = (left_sum, right_sum)
       return closest_subset
```



```
[] 🔅
                                                          ≪ Share
                                                                                    Output
main.py
                                                                         Run
                                                                                 [[34. 22.]
 1 import numpy as np
                                                                                    [38. 34.]]
 3 def strassen(A, B):
                                                                                   === Code Execution Successful ===
        if len(A) == 1:
        mid = len(A) // 2
        A11 = A[:mid, :mid]
        A12 = A[:mid, mid:]
        A21 = A[mid:, :mid]
        A22 = A[mid:, mid:]
        B11 = B[:mid, :mid]
        B12 = B[:mid, mid:]
        B21 = B[mid:, :mid]
        B22 = B[mid:, mid:]
        M1 = strassen(A11 + A22, B11 + B22)
        M2 = strassen(A21 + A22, B11)
        M3 = strassen(A11, B12 - B22)
M4 = strassen(A22, B21 - B11)
        M5 = strassen(A11 + A12, B22)
        M6 = strassen(A21 - A11, B11 + B12)
M7 = strassen(A12 - A22, B21 + B22)
26
        C11 = M1 + M4 - M5 + M7
```

