```
from collections import defaultdict
def fourSumCount(A, B, C, D):
    AB_sum = defaultdict(int)
    count = 0
    for a in A:
        for b in B:
            AB_sum[a + b] += 1
    for c in C:
        for d in D:
            count += AB_sum[-c - d]
    return count
A1 = [1, 2]
B1 = [-2, -1]
C1 = [-1, 2]
D1 = [0, 2]
print(f"Output for Test Case 1: {fourSumCount(A1, B1, C1, D1)}")

arr = [1, 2, 3, 4, 5, 6, 7, 8, 9, 10]
k = 6
print(sorted(arr)[k-1])




def strassen_matrix_multiply(A, B):
    if len(A) == 2:
        a, b, c, d = A[0][0], A[0][1], A[1][0], A[1][1]
        e, f, g, h = B[0][0], B[0][1], B[1][0], B[1][1]

        p1 = a * (f - h)
        p2 = (a + b) * h
        p3 = (c + d) * e
        p4 = d * (g - e)
        p5 = (a + d) * (e + h)
        p6 = (b - d) * (g + h)
        p7 = (a - c) * (e + f)

        C = [[p5 + p4 - p2 + p6, p1 + p2], [p3 + p4, p1 + p5 - p3 - p7]]
        return C
A = [[1, 7], [3, 5]]
B = [[6, 8], [4, 2]]

C = strassen_matrix_multiply(A, B)
print(C)



x=1234
y=5678
z=x*y
print(z)



def quick_sort(arr):
    if len(arr) <= 1:
        return arr
    pivot = arr[len(arr) // 2]
    left = [x for x in arr if x < pivot]
    middle = [x for x in arr if x == pivot]
    right = [x for x in arr if x > pivot]
    return quick_sort(left) + middle + quick_sort(right)
arr = [3, 6, 8, 10, 1, 2, 1]
sorted_arr = quick_sort(arr)
print(sorted_arr)
```

```
import heapq
def prim(n, edges):
    adj_list = [[] for _ in range(n)]
    for u, v, weight in edges:
        adj_list[u].append((weight, v))
        adj_list[v].append((weight, u))
    visited = [False] * n
    min_heap = [(0, 0)]
    mst = []
    total_weight = 0
    while min_heap:
        weight, u = heapq.heappop(min_heap)
        if visited[u]:
            continue
        visited[u] = True
        total_weight += weight
        if weight != 0:
            mst.append((prev, u, weight))
        for next_weight, v in adj_list[u]:
            if not visited[v]:
                heapq.heappush(min_heap, (next_weight, v))
                prev = u
    return mst, total_weight
n1 = 4
edges1 = [(0, 1, 10), (0, 2, 6), (0, 3, 5), (1, 3, 15), (2, 3, 4)]
mst1, total_weight1 = prim(n1, edges1)
print("Test Case 1:")
print("Edges in MST:", mst1)
print("Total weight of MST:", total_weight1)
```

```
import heapq

def kclosest(points,k):

    max_heap=[]

    for x,y in points:

        dist=-(x*x+y*y)

        if len(max_heap)<k:

            heapq.heappush(max_heap,(dist,x,y))

        else:

            heapq.heappushpop(max_heap,(dist,x,y))

    return [(x,y)for _,x,y in max_heap]

points=[[1,3],[-2,2],[5,8],[0,1]]

k=2

print(kclosest(points,k))
```