# Maximum Sum Combinations

**Problem Description**

Given two equally sized 1-D arrays **A, B** containing **N** integers each.

A **sum combination** is made by adding one element from array **A** and another element of array **B**.

Return the **maximum C valid sum combinations** from all the possible sum combinations.

**Input Format**

First argument is an one-dimensional integer array **A** of size **N**.

Second argument is an one-dimensional integer array **B** of size **N**.

Third argument is an integer **C**.

**Output Format**

Return a one-dimensional integer array of size **C** denoting the top C maximum sum combinations.

**NOTE:**

The returned array must be sorted in non-increasing order.

**Example Input**

Input 1:

```
A = [3, 2]
B = [1, 4]
C = 2
```

Input 2:

```
A = [1, 4, 2, 3]
B = [2, 5, 1, 6]
C = 4
```

**Example Output**

Output 1:

```
[7, 6]
```

Output 1:

```
[10, 9, 9, 8]
```

### Example Explanation

Explanation 1:

```
7      (A : 3) + (B : 4)
6      (A : 2) + (B : 4)
```

Explanation 2:

```
10    (A : 4) + (B : 6)
9    (A : 4) + (B : 5)
9    (A : 3) + (B : 6)
8    (A : 3) + (B : 5)
```

```
vector<int> Solution::solve(vector<int> &A, vector<int> &B, int C)
{
   sort(A.begin(), A.end(), greater<int>());
   sort(B.begin(), B.end(), greater<int>());
   priority_queue<int, vector<int>, greater<int>> pq;
   int N = A.size();
   for(int i=0; i<C; i++)
   {
      pq.push(A[i] + B[i]);
   }
   vector<int> ans(C);
   for(int i=0; i<N; i++)
   {
      for(int j=0; j<N; j++)
      {
         if(i==j)
         continue;
         if(A[i] + B[j] > pq.top())
         {
            pq.pop();
            pq.push(A[i] + B[j]);
         }
         else
         break;
      }
   }
```

```
    for(int i=C-1; i>=0; i--)
    {
        ans[i] = pq.top();
        pq.pop();
    }
    return ans;
}
```