**Optimal Strategy for A Game**

**Medium**Accuracy: 52.29% Submissions: 17349 Points: 4

You are given an array **A of size N**. The array contains integers and is of **even length**. The elements of the array represent N **coin**of **values V1, V2, ....Vn**. You play against an opponent in an **alternating**way.

In each **turn**, a player selects either the **first or last coin** from the **row**, removes it from the row permanently, and **receives the value** of the coin.

You need to determine the **maximum possible amount of money**you can win if you **go first**.  
**Note:** Both the players are playing optimally.

**Example 1:**

**Input:**

N = 4

A[] = {5,3,7,10}

**Output:** 15

**Explanation:** The user collects maximum

value as 15(10 + 5)

**Example 2:**

**Input:**

N = 4

A[] = {8,15,3,7}

**Output:** 22

**Explanation:** The user collects maximum

value as 22(7 + 15)

**Your Task:**  
Complete the function **maximumAmount()** which takes an array arr[] (represent values of N coins) and N as number of coins as a parameter and returns the **maximum possible amount of money**you can win if you **go first**.

**Expected Time Complexity** : O(N\*N)  
**Expected Auxiliary Space**: O(N\*N)

**Constraints:**  
2 <= N <= 103  
1 <= Ai <= 106

class Solution{

    public:

    int dp[1001][1001];

    long long maximumScore(int i, int j, int arr[]) {

        if (j==i) return arr[i];

        else if (j-i==1) return max(arr[i], arr[j]);

        if (dp[i][j]!=-1) return dp[i][j];

        return dp[i][j]=max(arr[i]+min(maximumScore(i+2, j, arr), maximumScore(i+1, j-1, arr)),

        arr[j]+min(maximumScore(i, j-2, arr), maximumScore(i+1, j-1, arr)));

    }

    long long maximumAmount(int arr[], int n){

        // Your code here

        memset(dp, -1, sizeof(dp));

        return maximumScore(0, n-1, arr);

    }

};