In this exam, given an array of positive numbers, you are asked to find a the maximum sum of a subsequence of the array with the constraint that any two numbers in the subsequence should have at least an index difference of 3 in the array (e.g. in a={'0','1','2','3','4'}, index difference of '4' and '1' is 3). To illustrate, when arr = {50, 30, 100, 10, 80, 100} is given, your functions should return 200 (sum of 100 and 100) or when arr = {8, 9, 15} is given, they should return 15.
You will implement three different functions for three different solutions of that problem:

- Direct recursive implementation in ***recursive_sln()***
- Recursion with memoization in ***memoization_sln()***
- Dynamic programming in ***dp_sln()***

All three functions are expected to **return** the answer to the given problem which is **the maximum sum value** (such that index difference between elements is at least 3).
Return **only** the max sum value and nothing more.

The number of recursive calls that your recursive function makes should be counted. That number should be counted and stored using the ***int &number_of_calls*** variable***,*** which is the last parameter at the definition of the *recursive_sln()*. Basically, the value of that variable should be incremented by one at each execution of the *recursive_sln()* function. In order to accomplish that, the increment operation may be done at the first line of the function implementation, as already done in the function template given to you. So, **do not change the first line of the *recursive_sln()* function and do not manipulate the *number_of_calls* variable at anywhere else**. Do **not return** that variable. Since it is passed by reference, its final value will be available for testing/grading without returning it.

For memoization and dynamic programming, you should use ***int*& mem*** variable (i.e. array), which is the last parameter at definitions of those functions, as **the array of memoized values**. For both *memoization_sln()* and *dp_sln()* functions, final values in the *mem* variable will be considered for grading. While testing and grading, the *mem* array will be initialized to all -1's. So, while implementing your functions, **you can assume that *mem* is an array of -1's. Do no return that variable/array.**

The ***int*& arr*** variable is the parameter which passes the input array to your functions. **Do not modify that array!**

At *recursive_sln()* and *memoization_sln()*, ***int i*** is intended to represent and pass indices of arr. While testing and grading, it will be initialized to **sizeof(arr)-1** (i.e. the last index of the array) . At *dp_sln()*, instead of such a variable, directly the **size of the arr** is given via ***int size*** parameter.

Implement the functions in most efficient way.