Attending Workshops

A student signed up for \( n \) workshops and wants to attend the maximum number of workshops where no two workshops overlap. You must do the following:

Implement 2 structures:

1. **struct Workshop** having the following members:
   - The workshop's start time.
   - The workshop's duration.
   - The workshop's end time.

2. **struct Available_Workshops** having the following members:
   - An integer, \( n \) (the number of workshops the student signed up for).
   - An array of type **Workshop** array having size \( n \).

Implement 2 functions:

1. **Available_Workshops* initialize (int start_time[], int duration[], int n)**
   Creates an **Available_Workshops** object and initializes its elements using the elements in the **start_time[]** and **duration[]** parameters (both are of size \( n \)). Here, **start_time[i]** and **duration[i]** are the respective start time and duration for the \( i^{th} \) workshop. This function must return a pointer to an **Available_Workshops** object.

2. **int CalculateMaxWorkshops(Available_Workshops* ptr)**
   Returns the maximum number of workshops the student can attend—without overlap. The next workshop cannot be attended until the previous workshop ends.

**Note:** An array of unkown size \((n)\) should be declared as follows:

```c
DataType* arrayName = new DataType[n];
```

**Input Format**

Input from stdin is handled by the locked code in the editor; you simply need to write your functions to meet the specifications of the problem statement above.

**Constraints**

- \( 1 \leq N \leq 10^5 \)
- \( 0 \leq start\_time_i \leq 10^3 \)
- \( 0 \leq duration_i \leq 10^3 \)

**Output Format**

Output to stdout is handled for you.

Your **initialize** function must return a pointer to an **Available_Workshops** object.
Your **CalculateMaxWorkshops** function must return maximum number of non-overlapping workshops the student can attend.

**Sample Input**
Sample Output

`CalculateMaxWorkshops` should return 4.

Explanation

The first line denotes $n$, the number of workshops.
The next line contains $n$ space-separated integers where the $i^{th}$ integer is the $i^{th}$ workshop's start time.
The next line contains $n$ space-separated integers where the $i^{th}$ integer is the $i^{th}$ workshop's duration.

The student can attend the workshops 0, 1, 3, and 5 without overlap, so `CalculateMaxWorkshops` returns 4 to `main` (which then prints 4 to stdout).