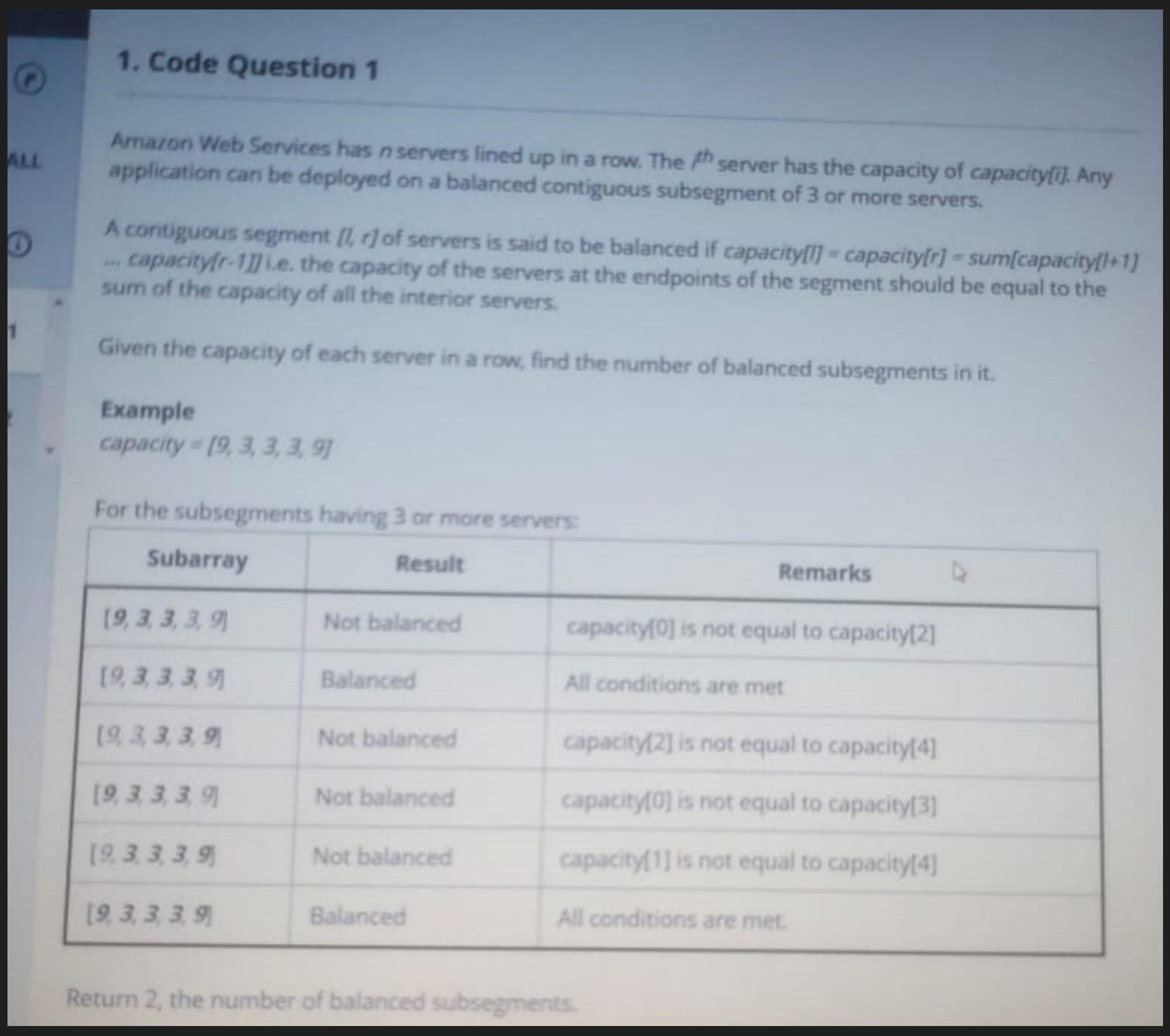
Amazon Dublin Internship Questions.



1. Code Question 1

Amazon Web Services has *n* servers lined up in a row. The *i*th server has the capacity of *capacity[i]*. Any application can be deployed on a balanced contiguous subsegment of 3 or more servers.

A contiguous segment *[l, r]* of servers is said to be balanced if *capacity[l] = capacity[r] = sum[capacity[l+1] ... capacity[r-1]]* i.e. the capacity of the servers at the endpoints of the segment should be equal to the sum of the capacity of all the interior servers.

Given the capacity of each server in a row, find the number of balanced subsegments in it.

Example

capacity = [9, 3, 3, 3, 9]

For the subsegments having 3 or more servers:

Subarray	Result	Remarks
[9, 3, 3, 3, 9]	Not balanced	capacity[0] is not equal to capacity[2]
[9, 3, 3, 3, 9]	Balanced	All conditions are met
[9, 3, 3, 3, 9]	Not balanced	capacity[2] is not equal to capacity[4]
[9, 3, 3, 3, 9]	Not balanced	capacity[0] is not equal to capacity[3]
[9, 3, 3, 3, 9]	Not balanced	capacity[1] is not equal to capacity[4]
[9, 3, 3, 3, 9]	Balanced	All conditions are met.

Return 2, the number of balanced subsegments.

Pseudo code :-

```
Solve(N, capacities) {
    vector<int> prefixSum(N,0);
    prefixSum[0] = capacities[0];

    for (i ← 1 to N) {
            prefixSum[i] = prefixSum[i-1]
                                  + capacities[i];
    }

    unordered_map<int, int>   lookup.

    for (i ← 0 to N) {
        if ( lookup.find(capacities[i])
                  == Lookup.end()) {
    →         Lookup[capacities[i]= i;
              continue;
        }

        int interion_capacities = prefixSum(i-1)
                                  - prefixSum[capaities[i]];

        if (interion_Capacities == capacities[i]) {
                Counter++;
    →           Lookup[capacities[i]] = i;
                contime;
        }

        if (interioR _ Capaities) {
    →           Lookup(capacities[i]] = i;
        }
```

i
↓

0	1	2	3	4	
9	3	3	3	9	← Server capacities
9	12	15	18	27	← prefix Sum

lookup map :

9	→	0
3	→	3

prefixSum[i-1] − prefixSum[j]

18 − 9 = 9

9 == 9

Count → +2

1. Create a prefixSum array.
2. Initialize a map and a counter variable
   ↳ 0 (initial val).
3. Traverse the servers' capacities array.

* 4. Check if the current capacity is present in the map.

```
↳ if present:
      → ser_cap[i] == prefixSum[i-1]
                        − prefixSum[map[server[i]]]
              ↳ counter += 1
      → prefixSum[i-1] − prefixSum[map[server[i]]] == 0
              ↳ map[server[i]] = i
↳ else
      ↳ map[server[i]] = i.
```

5. Return Counter.