SubArrays

* for printing all the subarrays

```
for(int sp=0; sp< N ; sp++)
{
   for(int ep=sp; ep< N ; ep++)
   {
      for(int i=sp ; i<=ep ; i++)
      {
         print (arr[i]);
      }
   }
}
```

Starting point

ending point

Subarray with Starting point and ending point

* if you want to print sum of each subarrays

① first create a prefixSum array → psum[ ]

```
for( int sp =0 ; sp< n ; sp++)
{
   for(int ep =sp; ep<n; ep++)
   {
      if (sp ==0)  print (psum[ep])
      else  print (psum[ep] - psum[sp-1])
   }
}
```

Array

6	9	3	2	1
0	1	2	3	4

if sp = 1
ep = 4

6	9	3	2	1
psum[sp-1]	0	1	2	3	4

This is nothing but psum[ep] - psum[sp-1]

* Run a loop of sp → (0 to n)

ep → (sp to n)

// [sp, ep] → Our valid subarray range

To get sum of all subarrays

Contribution Technique

arr[] : (5, 3, -1, 8)

5			
5	3		
5	3	-1	
5	3	-1	8
	3		
	3	-1	
	3	-1	8
		-1	
		-1	8
			8

Subarrays starting with (5) (3) (-1) (8)

Value — Occurance

arr[0] = 5 — 4 times ⇒ 5 * 4 = 20

arr[1] = 3 — 6 times ⇒ 3 * 6 = 18

arr[2] = -1 — 6 times ⇒ -1 * 6 = -6

arr[3] = 8 — 4 times ⇒ 8 * 4 = 32

Ans = 20 + 18 + (-6) + 32 = 64

Generalising the approach of value and their occurences

arr[4] : (5, 3, -1, 8)
↑ ↑ ↑ ↑
x_1 x_2 x_3 x_4 (no. of times appeared)

∴ ans = (arr[0] * x_1) + (arr[1] * x_2) + (arr[2] * x_3) ... ...

6 9 3 Then how to find $x_0, x_1, \ldots$ (no. of occurances of each element)

for example

* A subarray which includes (-1) is having :

arr[] : (3, -2, 4, -1, 2, 6)

sp ep

(sp : starting point
ep : ending point)

Similarly ← endpoint

0	3
1	4
2	5
3	

sp=0 → (3, -2, 4, -1, 2, 6)

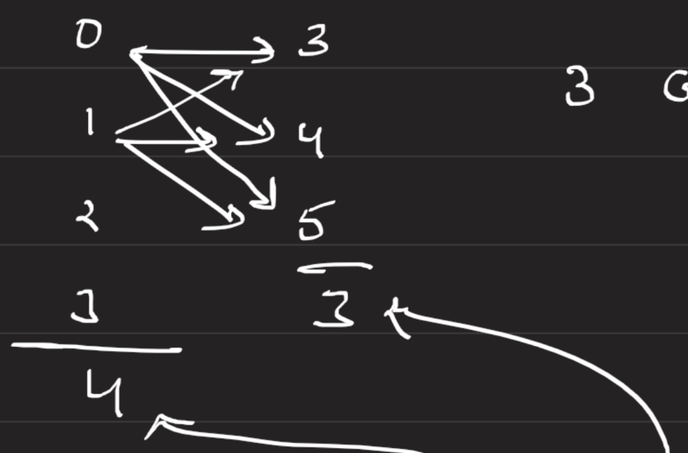
sp=1 → (3, -2, 4, -1, 2, 6)

sp=2 → (3, -2, 4, -1, 2, 6)

sp=3 → (3, -2, 4, -1, 2, 6)

sp ep

(sp : starting point
ep : ending point)



3 combinations with 1

3 combinations with '0'

Likewise, we total have (4) * (3) combinations (occurences)

∴ if i is '3' → Valid starting points = (i+1) = 4 (i : index)

Valid ending points = (n-i)

∴ ans = (i+1) * (n-i) → if you want x_1 for (n=4) = (1+1) * (4-1) = 2*3 = 6

// Pseudocode

```
int ans = 0;
for (int i=0 ; i<n; i++)
{ int occ = (i+1)*(n-i);
  ans += (occ * arr[i]) ;
}
```

x_0
x_1
x_2
x_3