Quiz-3 Detailed Solutions

1. Array → data structure that stores similar type of elements

```
int arr[];
```
↳ Stores only integer elements.

2. Array is used to store collection of data.

```
int arr[] = {1,2,3,4};
```
Collection of data.

3. An array is used to store data at contiguous memory locations.

```
int arr[] = {1,2,3};
```

Address → 100 104 108

4 bytes → integer size

100 + 1X4 = 104
104 + 1X4 = 108

4. Lowest address of array is 0. Indexing in array always start from 0.

5. `int arr [0];` → Does not store any data as size = 0

`int arr [1] = {1};` → Stores one integer

array size > 0 , we can conclude

6.

`int arr [5] = {1, 2, 3, 4, 5, 6};` Not possible

`int arr [5] = {1, 2, 3};` Possible

`int arr [5] = {1, 2, 3, 4, 5};` Possible

7. `int array [10];`

↳ name of array

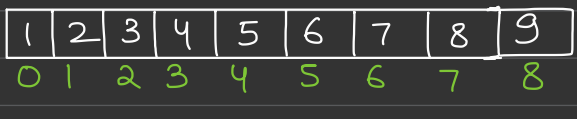
↳ size of array

↓ data type

8. Last index = size − 1 } formulae

= 9 − 1

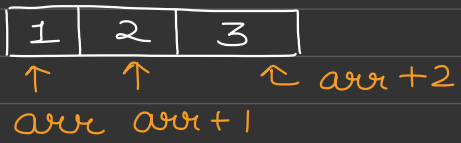
= 8



9. int arr [ ],

arr → Base address of array

↳ This will be the 1st element of the array.



10. result += .... ≠ result = result + __

array 1 → { 1200, 200, 2300, 1230, 1543 }

temp = 0

result = 0 + 1200 = 1200

temp = 1

result = 1200 + 200 = 1400

temp = 2

result = 1400 + 2300 = 3700

temp = 3

result = 3700 + 1230 = 4930

temp = 4

result = 4930 + 1543 = 6473

array2 → {12, 14, 16, 18, 20}

temp = 0

result = 6473 + 12 = 6485

temp = 1

result = 6485 + 14 = 6499

temp = 2

result = 6499 + 16 = 6515

temp = 3

result = 6515 + 18 = 6533 Ans

11.

Space complexity = O(1) as only variables are created which takes constant space

```
for (i=0 ; i < N ; i++) {
        --
}
for (i=0 ; i < M ; i++) {
        --
}
```

O(N)

O(M)

Time complexity = O(N+M) as the loops are not nested.

12. As the loops are nested, Time complexity will be multiplied

In worst case, outer for loop will run for N times.
In worst case, inner for loop will run for N times

Hence $O(N*N) \rightarrow O(N^2)$

13. As the loops are nested, Time complexity will be multiplied

Outer for loop will run for $\frac{n}{2}$ in worst case.

$$O\left(\frac{n}{2}\right) = O(n) \text{ only.}$$

Wherever we see $i = \frac{i}{2}$ or $i = i * 2$

In worst case the loop runs for $\log_2 n$ times.

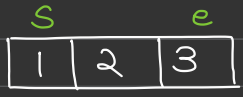
Hence $TC = O(\log_2 n)$

$TC = O(n \times \log_2 n)$

14. There is only single loop & we see $i = \frac{i}{2}$ also

Hence $TC = O(\log n)$

15. No this is not always true. This is the upper bound. Let's take an example of linear & binary search.



Linear search for key = 1 will be done in 1 comparasion

Whereas binary search won't do in 1 comparasion. Hence in this case linear search which is $O(n)$ is better than binary search $O(\log n)$.