

Q1 Reverse the array.

i/p  $\rightarrow \{1, 2, 3\}$

o/p  $\rightarrow \{3, 2, 1\}$

The approach of the above question can be of 2 pointer approach. Take  $i=0$  &  $j=\text{size}-1$ , Swap them & increment the value of  $i$  and decrement the value of  $j$ . This will go on until  $i \leq j$ .

Dry run on array of characters

1)  $s = \{ 'h', 'e', 'l', 'l', 'o' \}$

$i=0, j=4$

in this case as  $s.\text{size}() - 1 = 4$  here

$i \leq j \rightarrow \text{True}$

Swap( $s[i], s[j]$ )

$i++$

$j--$

2)  $s = \{ 'o', 'e', 'l', 'l', 'h' \}$

$i=1, j=3$

$i \leq j \rightarrow \text{True}$

Swap( $s[i], s[j]$ )

$i++$

$j--$

3)  $s = \{ 'o', 'l', 'l', 'e', 'h' \}$

$i=2, j=2$

$i \leq j \rightarrow \text{True}$

Swap( $s[i], s[j]$ )



$i++$  $j--$ 

4) { 'o', 'l', 'l', 'e', 'h' }

Here  $i > j$  & hence the condition is false.

Hence here we have got the reversed array.

Code

```
void reverseString (vector <char> & s) {
```

```
    int i = 0; // First index
```

```
    int j = s.size() - 1; // Last index
```

```
    // Running loop
```

```
    while (i <= j) {
```

```
        swap (s[i], s[j]); // Swap
```

```
        i++; // Increment the value of i
```

```
        j--; // Decrement the value of j
```

```
    }
```

```
}
```

The above code can be run for integer arrays also. Just use vector of integer.

\* Time Complexity =  $O(n)$

\* Space Complexity =  $O(1)$  → no extra space is used.