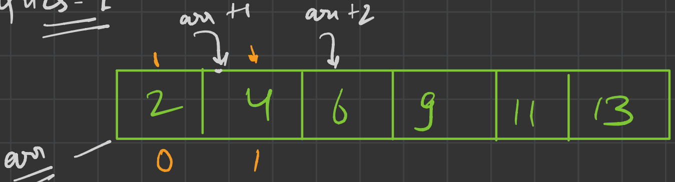
Recursion (Day-3)

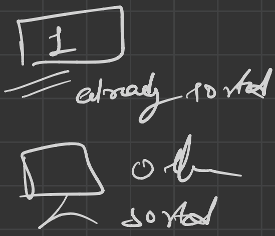
→ Ques-1

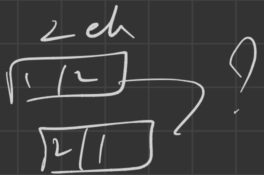


function → isSorted

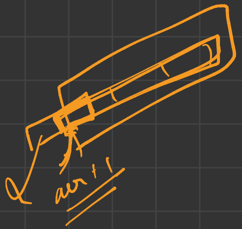
Loops

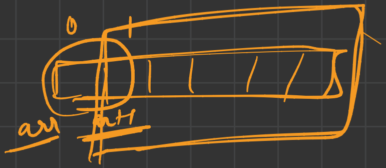
Recursion





```
bool isSorted (int arr[], int size)
{

    // base case

    if (size==0 || size==1)
        return true;

    if (arr[0] > arr[1])   //sorted nhi h
        return false

    else
        bool ans = isSorted (arr+1 , size-1)
        return ans;
```





H|W →

i/p →

3	2	5	1	6

o/p → Sum = $\boxed{17}$

→ Loop ✓

→ Recursion

③ Linear Search

i/p → array -

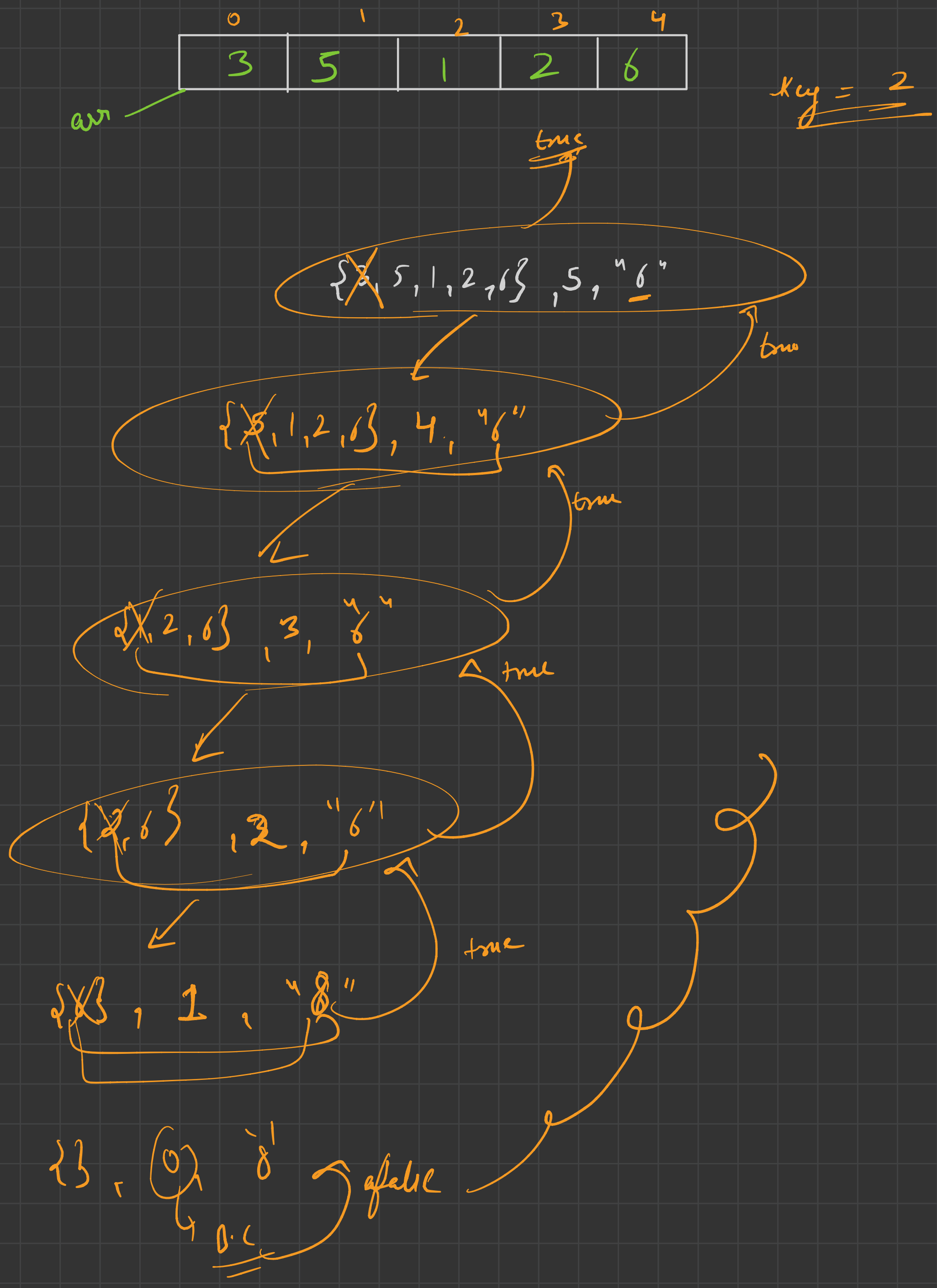
1	2	3	4	5

key/element = $\boxed{6}$

o/p → Found / Not found

→ Loop ✓ ✓

→ Recursion ✓



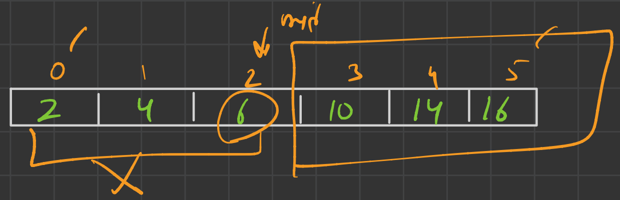
0 1 2 3 4
3 5 1 2 6
arr
key = 2
true
{3, 5, 1, 2, 6}, 5, "6"
true
{5, 1, 2, 6}, 4, "6"
true
{1, 2, 6}, 3, "6"
true
{2, 6}, 2, "6"
true
{6}, 1, "8"
{}, 0, "8"
B.C
false

Binary Search

→ impl: ↳ while (s<=e)

```
{
    if (        )
        ret   mid
    if (         )
        s = mid +1
    els
        e = mid - 1
    mid = (      )/2
}
```

Recursion → B.S implement

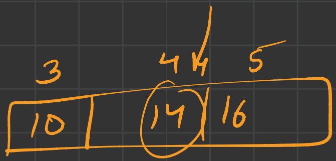


Algo:

mid = $\frac{0+5}{2}$ = ②

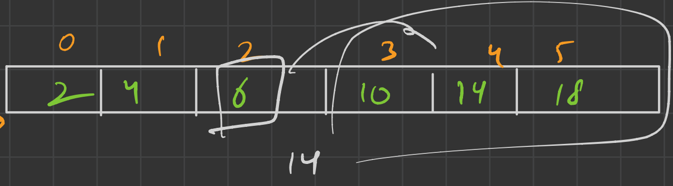
14 = key

14 > 6



mid = $\frac{3+5}{2}$ = ④

14 = = 14 ↳ found



arr →

Key = 18

mid = $\frac{0+5}{2}$ = 2

$\boxed{arr[mid] = 4}$

if (arr[mid] < key)
↳ search in right half

else
↳ search karo left half

search in right half → s = mid + 1

search in left half → e = mid - 1

binarySearch(arr, s, e, K)

→ binarySearch(arr, mid + 1, e, K)

→ binarySearch(arr, s, mid - 1, K)

H/w

B&S Question (12, Lec — 15)

↳ solve again using recursion