(8) Binary search in a Nearly sorted Array

0	1	2	3	4	5	6
10	3	40	20	50	80	70

→ Nearly sorted array.

0	1	2	3	4	5	6
3	10	20	40	50	70	80

→ sorted array.

An element which is present at ith index of sorted array will ~~or~~ be present at ith, i-1th or i+1th index of nearly sorted array. ex: (1) 50 → 4th index of sorted array

$$\begin{tikzcd} & 50 \arrow[dl] \arrow[d] \arrow[dr] & \\ 3 & 4 & 5 \end{tikzcd}$$ [nearly sorted]

50 also present at 4th index of n.s a.

(2) 70 → 5th index s.a

$$\begin{tikzcd} & 70 \arrow[dl] \arrow[d] \arrow[dr] & \\ 4 & 5 & 6 \end{tikzcd}$$ 7 → 6th index of n.s. a

(3) 20 → 2th index of s. a

$$\begin{tikzcd} & 20 \arrow[dl] \arrow[d] \arrow[dr] & \\ 1 & 2 & 3 \end{tikzcd}$$ → 20 → 3rd index of n.s.a.

```
int nearly_sorted_array( vector<int> arr, int target) {
    int s=0,  e= arr.size()-1;
    int mid = s+ (e-s)/2;
    while (s<=e) {
        if(arr[mid] == target) {
              return mid;
        }
        else if (arr[mid-1] == target) {
             return mid-1;
        }
        else if ( arr[mid+1] == target) {
             return mid+1;
        }
        else if (arr[mid]> target) {
             end= mid-1;
        }
        else if ( arr[mid] < target) {
             start= mid+1;
        }
        mid= s+ (e-s)/2;
    }
    return -1;
}
```

(9) Dividing 2 number using Binary search:

$\frac{\text{dividend}}{\text{divisor}}$ = quotient → quotient = dividend * divisor + remainder

want to ignore then

If we divide any dividend by divisor then the quotient will lie b/w 0 → dividend (search space) in sorted order

$\frac{\text{dividend}}{\text{divisor}}$ = quotient

dividend >= quotient * divisor

```
int divide2nos (int dividend, int divisor){
  int s=0, e=abs(dividend);
  int ans=-1.
  int mid= s~~tart~~+(e-s)/2;
  while (s<=e){
        if (abs(mid* divisor) ~~>~~=abs (dividend)){
          . ans=mid;
            break;
        }
        else if ~~{{~~(abs(mid* divisor) < abs (dividend)){
            e~~o~~ = mid-1;
        }
        else if {(abs(mid* divisor) > abs (dividend)) {
            ans = mid.
            s = mid+1;
        }
        m= s+(e-s)/2;
  }.
  if((dividend >0 && divisor>0) && (dividend <0 && divisor<0))
  {
   return ans;
  }
  else {
   return -ans;
  }
}
```

for decimal part same as sqrt of any no.
just to keep remainder of using abs func.

⑩ <u>find odd</u> <u>occuring element</u> in <u>an array</u>:

→ <u>Input</u>

0	1	2	3	4	5	6	7	8	9	10	11	12
1	1	2	2	3	3	4	4	3	600	600	4	4
even	odd	e	o	e	o	e	o		o	e	o	e

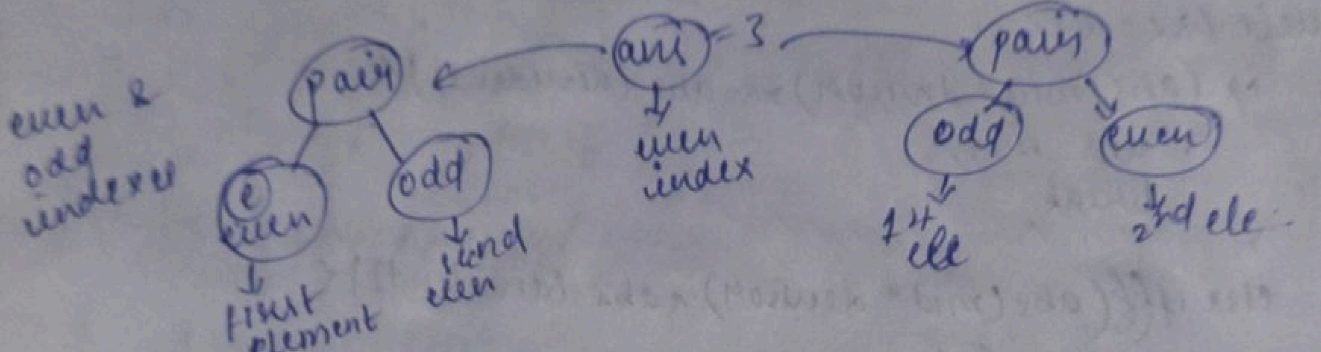
<u>Statement</u>: ① All elements occurs even no. of times.

$1 \to 2,\ 2 \to 2,\ 3 \to 3,\ 4 \to 4 \; 0,\ 600 \to 2$

↓ even no. of times

② The repeating elements must be in pair and no 2 pairs are adjacent to each other.

③ find the element that occur odd no. of times.



```
int findOddOccurEle (vector<int> arr){
  int s=0 ; e= arr.size() -1;
  int mid = s+(e-s)/2;
  int ans;
  while( s<= e) {
    if (s == e) {
      return s;
    if ( mid % 2 == 0) {
        if (arr[mid] == arr[mid+1]) {
            s = mid+2;
        }
        else{
          e = mid;
        }
    else{
      if ( arr[mid] == arr[mid-1]) {
          s = mid+1;
      }
      else { e = mid - 1;
      }  ——————→  m= s+ (e-s)/2; } return -1;
```