Max Number of K-Sum Pairs

arr = [4 , 7 , 1 , 3 , 6 , 8 , 5]

K = 10

pair1 → (4, 6)
pair2 → (7, 3)
} ans = 2

K = 10

num2 = K - num1

hashmap

element vs index

4 → 0	6 → 4
7 → 1	8 → 5
1 → 2	5 → 6
3 → 3	

num1 = 4 , num2 = 6
num1 = 7 , num2 = 3
num1 = 1 , num2 = 9
num1 = 3 , num2 = 7
num1 = 6 , num2 = 4
num1 = 8 , num2 = 2
num1 = 5 , num2 = 5

count = ~~0~~ ~~1~~ 2

code

```java
public static void main(String[] args) {
    Scanner scn = new Scanner(System.in);
    int n = scn.nextInt();
    int[] arr = new int[n];
    for (int i = 0; i < n; i++) {
        arr[i] = scn.nextInt();
    }
    int k = scn.nextInt();
    System.out.println(maxNumofK(arr, n, k));
}
public static int maxNumofK(int[] arr, int n, int k) {
    HashMap<Integer, Integer> map = new HashMap<>();
    for (int i = 0; i < n; i++) {
        map.put( arr[i], i );
    }

    int count = 0;
    for (int i = 0; i < n; i++) {
        int num1 = arr[i];
        int num2 = k - num1;
        if ( map.containsKey(num2) ) {
            if ( i != map.get(num2) ) {
                count++;
                map.remove(num1);
                map.remove(num2);
            }
        }
    }
    return count;
}
```

N

N

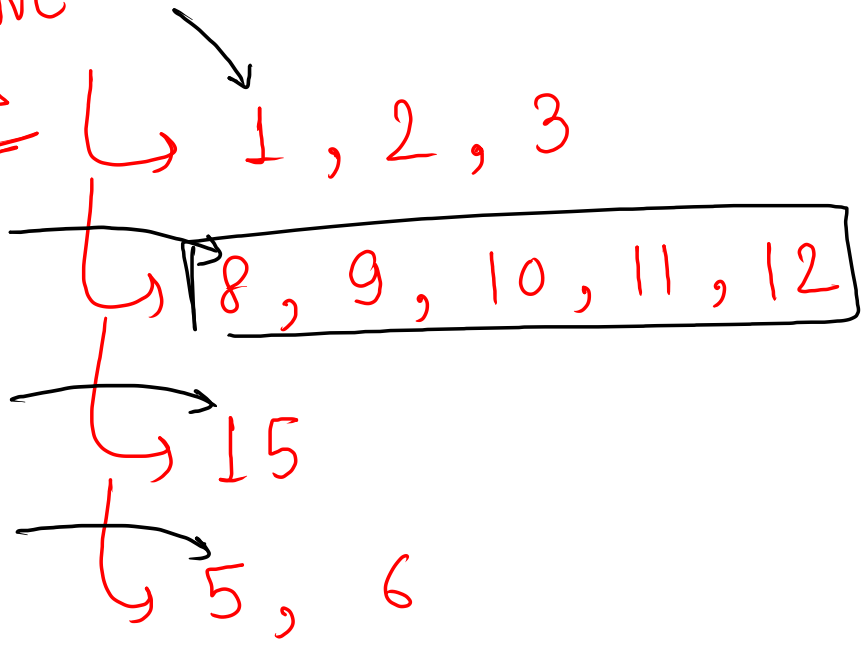
T.C = O(N)

S.C = O(N)

Longest Consecutive Sequence 2

arr = [10 , 5 , 9 , 1 , 11 , 8 , 6 , 15 , 3 , 12 , 2]

all consicutive sequences



ans = 5

Dry Run

arr = [10 , 5 , 9 , 1 , 11 , 8 , 6 , 15 , 3 , 12 , 2]

assumption assume all no. are starting of a consecutive seq.

step1 map

Integer vs Boolean

10 ✓	8 ✓
5 ✓	6 ✓
9 ✓	15 ✓
1 ✓	3 ✓
11 ✓	12 ✓
	2 ✓

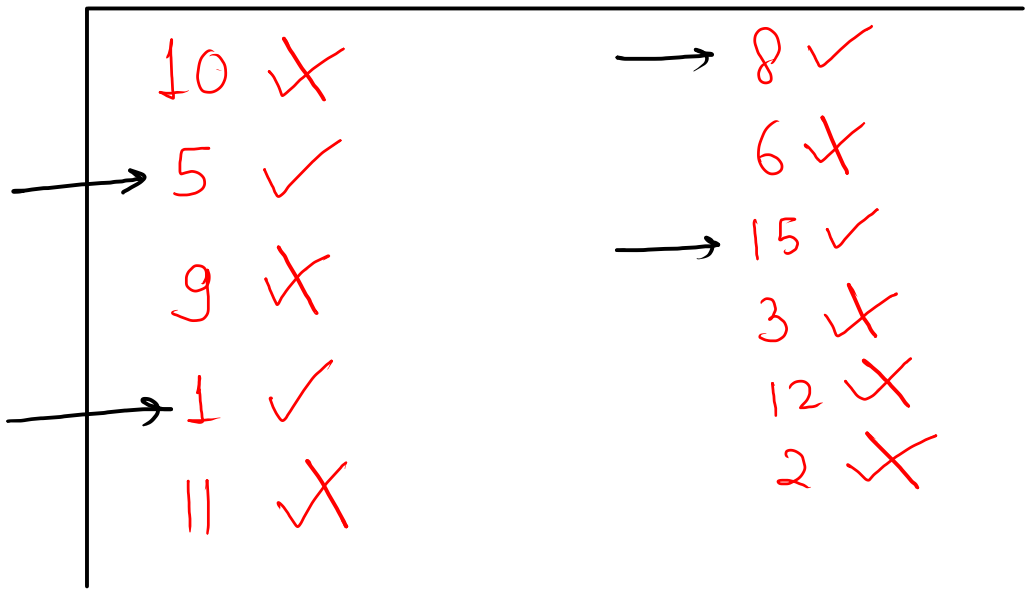
arr = [10 , 5 , 9 , 1 , 11 , 8 , 6 , 15 , 3 , 12 , 2]

make all those elements false, which are not the starting of a seq.

step 2

Integer vs Boolean

map



arr = [10 , 5 , 9 , 1 , 11 , 8 , 6 , 15 , 3 , 12 , 2]

start from true elements and count the length of each sequence

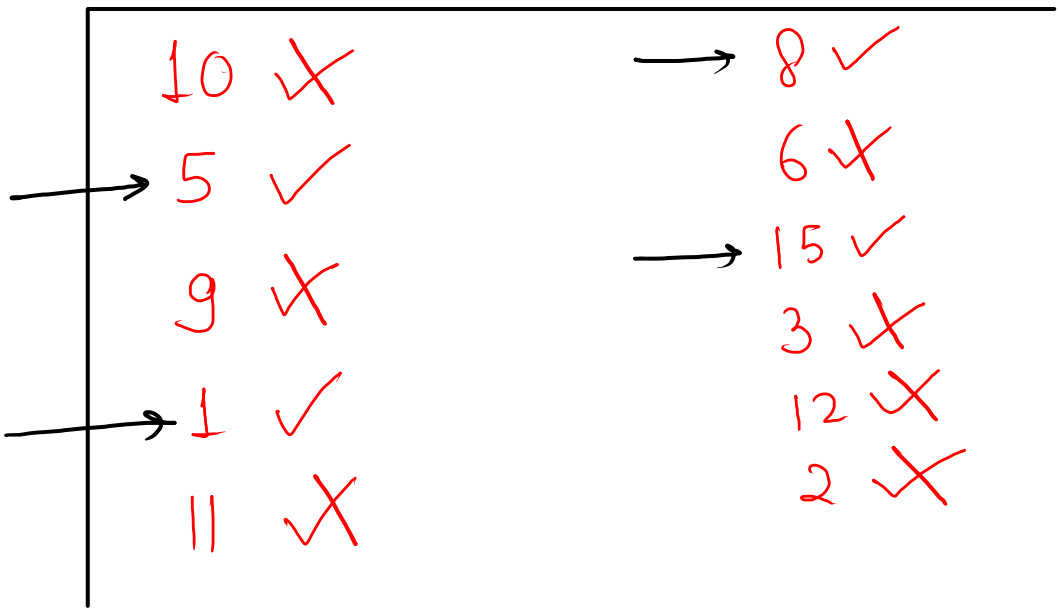
len = 1

ans = ~~0~~ ~~2~~ ~~3~~ 5

Integer vs Boolean

step 3

map



code

```java
public static int longestConsecutive(int[] arr, int n) {
    HashMap<Integer, Boolean> map = new HashMap<>();
    for (int val : arr) {
        map.put( val, true );
    }

    for (int val : arr) {
        if ( map.containsKey(val - 1) == true ) {
            map.put( val, false );
        }
    }

    int maxLen = 0;
    for (int val : arr) {
        if ( map.get(val) == true ) {
            int len = 1;
            int start = val;
            while ( map.containsKey( start + len ) == true ) {
                len++;
            }

            if ( len > maxLen ) {
                maxLen = len;
            }
        }
    }
    return maxLen;
}
```

Check if array pair are divisible by K (even length)

0	1	2	3	4	5	6	7	8	9
77	22	56	11	45	34	78	29	23	55

arr = [77, 22, 56, 11, 45, 34, 78, 29, 23, 55]

K = 10

size = 10

no. of pairs = 5

pair1 = 77 + 23
pair2 = 22 + 28
pair3 = 56 + 34
pair4 = 11 + 29
pair5 = 45 + 55

} make all pairs divisible by K

Note:- $(num1 + num2) \% K == 0$

$K = 10$

$(10n + 7)$

$77 \% 10 = 7$

$(10m + 3)$

$23 \% 10 = 3$

$num1 = 10n + 7$, $num2 = 10m + 3$

$$num1 + num2 = 10n + 7 + 10m + 3$$

$$= 10n + 10m + 10$$

$$= 10(n + m + 1)$$

22 (2)

28 (8)

56 (6)

34 (4)

11 (1)

29 (9)

45 (5)

55 (5)

$k = 10$

num1 = 77 , num2 = 23

rem1 = 7 rem2 = 3

if (x) then $(k - x)$

rem1 + rem2 = k

(x)

if 10 20

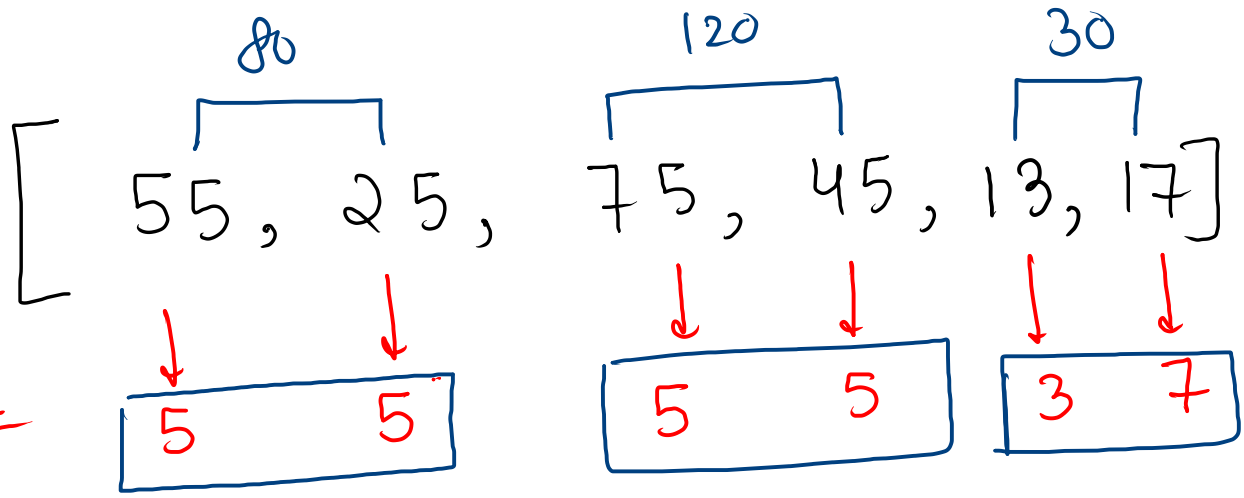
k = 10

rem = 0 rem = 0

[10, 20, 40, 70, 60, 23]

Note:- no. with rem = 0, should be even times

$K = 10$



means:- when rem is $\frac{K}{2}$, then they should also be appearing even no. of time

Imp :- (If any of the cond^n is wrong then return false)

1) rem1 + rem2 should be divisible by K

2) if rem = 0, then they should be appearing even time

3) if rem = $\frac{K}{2}$, then they should be appearing even time

arr = [77, 22, 56, 11, 45, 34, 78, 29, 23, 55]

0 1 2 3 4 5 6 7 8 9

map

remainder vs freq

7	1
2	1
6	1
1	1
5	2
4	1
8	1
9	1
3	1

rem = 7

rem = 3

1

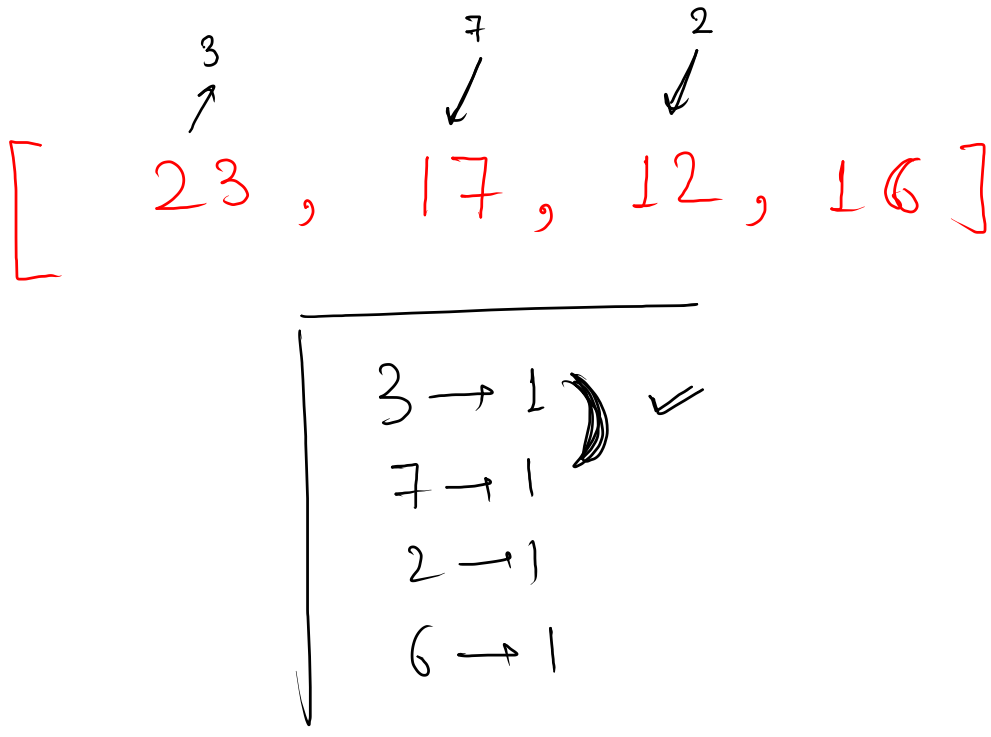
rem1 = x

rem2 = k - x

both should appear same no. of times

Error

$k = 10$



Code

```java
public static boolean pairDivisiblebyK(int[] arr, int n, int k) {
    HashMap<Integer, Integer> map = new HashMap<>();
    for (int i = 0; i < n; i++) {
        int rem = arr[i] % k;
        if ( map.containsKey(rem) == false ) {
            map.put( rem, 1 );
        } else {
            int freq = map.get(rem);
            map.put( rem, freq + 1 );
        }
    }

    for (int i = 0; i < n; i++) {
        int rem = arr[i] % k;
        if ( rem == 0 ) {
            int freq = map.get(rem);
            if ( freq % 2 == 1 ) { // odd
                return false;
            }
        } else if ( 2 * rem == k ) {
            int freq = map.get(rem);
            if ( freq % 2 == 1 ) { // odd
                return false;
            }
        } else {
            int freq1 = map.get(rem);
            int freq2 = 0;
            if ( map.containsKey(k - rem) == true ) {
                freq2 = map.get( k - rem );
            }
            if ( freq1 != freq2 ) {
                return false;
            }
        }
    }
    return true;
}
```

N

N

T.C = O(N)

S.C = O(N)

ex. to
dry run

arr = [9, 56, 25, 52, 72, 44, 80, 36, 96, 71, 55]

k = 8