Common elements b/w 3 sorted Array

App-1 (With extra space)

arr1 = 1, 5, 10, 20, 40, 80

arr2 = 6, 7, 20, 80 100

arr3 = 3, 4, 15, 20 30, 70, 80, 120

step 1 → Make unordered map of Arr1

mp1

1 → 1
5 → 1
10 → 1
20 → 1
40 → 1
80 → 1

Make two map mp1, mp1 and 2 vector a, b

```
for (int i=0; i<n2; i++)
    { if (mp.find(B[i]) != mp.end())
        { a.push_back(A[i]
          mp.erase(B[i])
    }
}
```

a

20	80

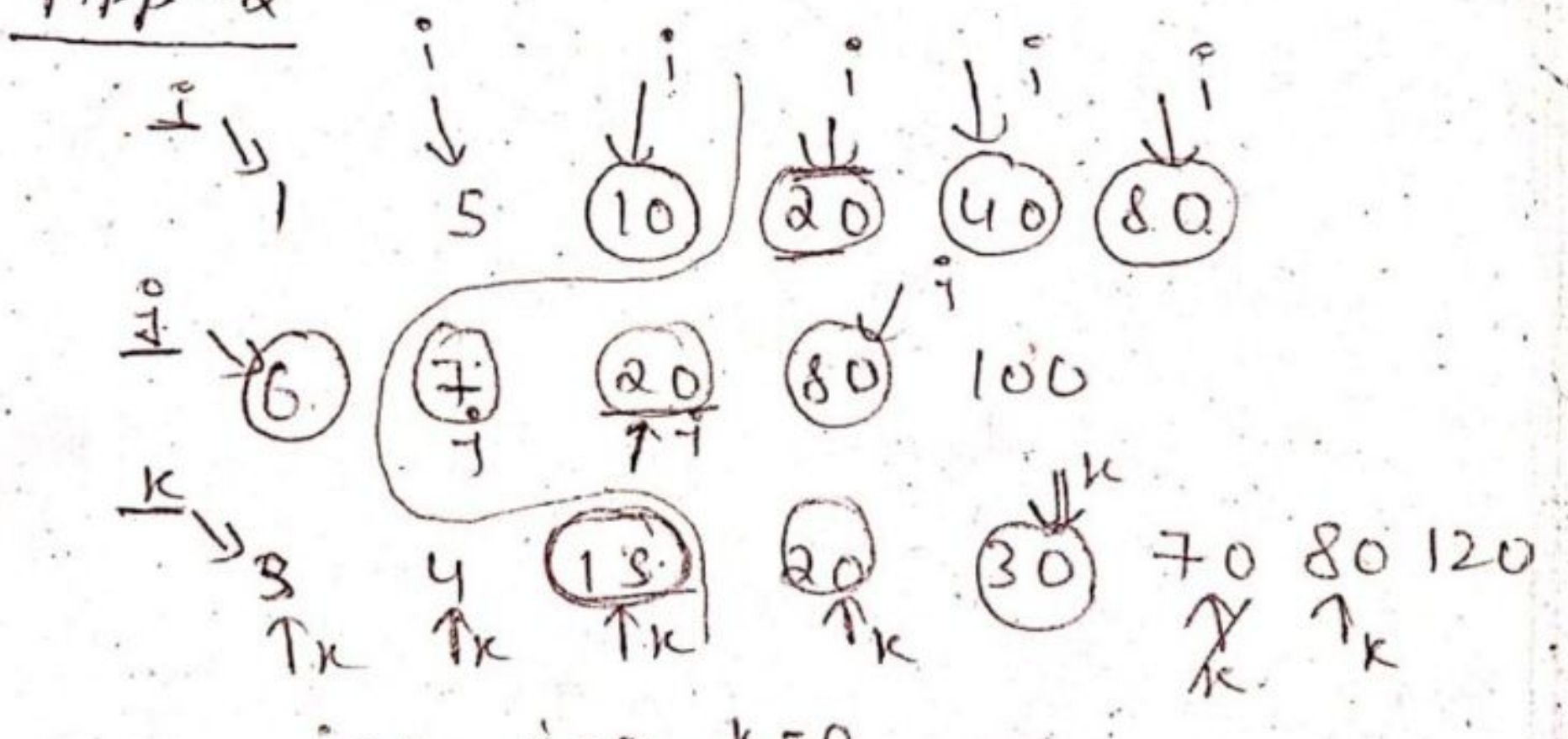
Scanned by CamScanner

Step2 - Map Map vector 9

mp1 20 → 1
80 → 1

```
for(int i=0; i<n3; i++)
  { if.(mp1.find() != mp.end())
      { b.push_back(c[i])
        mp1.erase(c[i]);
  } }
return b;
```

App-2



i=0, j=0, k=0

Maxi = max(a[i], max(b[j], c[k])

① max = 6 ans = 20, 80
② max = 15
③ max = 20
④ max = 80

Scanned by CamScanner

To protect from inserting duplicate use prev variable

when push value into vector
prev = that value

```
while (i<n1 && i<n2 && k<n3)
{ if (A[i]==B[j] && B[j]==C[k])
  { if (A[i]==prev[ous)
    { i++, j++, k++, continu
    else
      previous = A[i]
      result.push back(prev)
      i++ , j++, k++ ;
  }
else { int max = ————————
  while (arr[i]<max)
      i++
  while (arr[j]<max)
      j++
  while (arr[k]<max.)
      k++
  }
}
ritvrn result.
```

If Arr ki current value == maxi ke equal hoga to do nothing

Scanned by CamScanner

****

Q - Sort an array of 0's, 1's, 2's without extra space.

1st App - Sort the array in Ascending order.

Tc - O(nlogn)
Sc - O(1)

2nd App - Use Counting Sort (count variable)

| 0 | 1 | 1 | 0 | 1 | 2 | 1 | 2 | 0 | 0 | 0 | 1 |

Find count 0 = 5, count 1 = 5, count 2 = 2

| 0 | 0 | 0 | 0 | 0 | 1 | 1 | 1 | 1 | 1 | 2 | 2 |

- Which take two pases
1 → First for count occurance no
2 → put in new array one by one.

TC → O(N) + O(N)
SC → O(N)

3rd App - Use Dutch National Flag Algorithm

Scanned by CamScanner

6 27 28 13 27 28 29 30 31 Saturday 1 - -

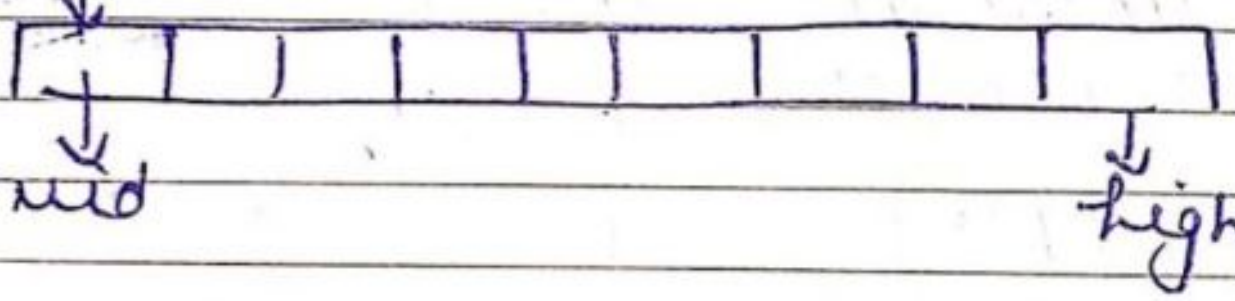
Note - Dutch National Flag → Also known as
0, 1, 2 sort

logic

if 0, swap a[low] & a[mid]
low++, mid++

if 1, mid++;

if 2, swap a[mid] & a[high]
high--



Scanned by CamScanner

factorial of a large no

5! → 5x4x3x2x1

or

A big no → 10! = 3628800

let's take ex

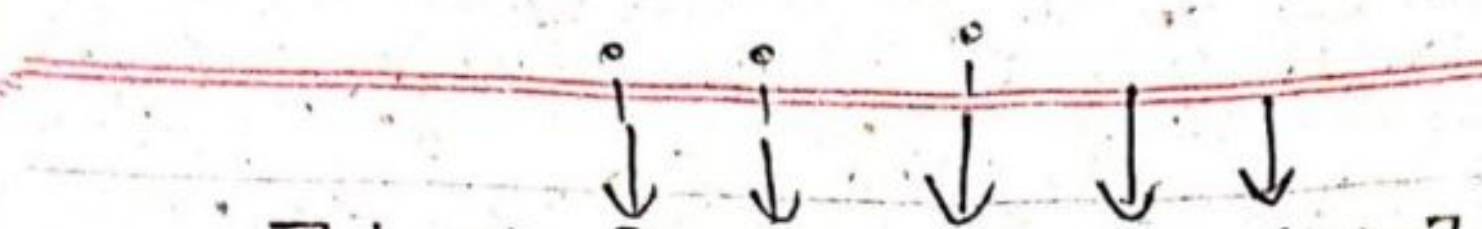
7! → 2, 3, 4, 5, 6, 7

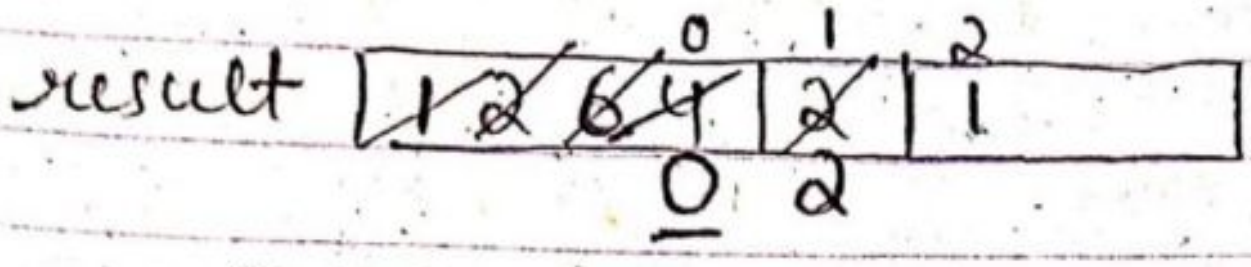
1

result | 1 |

```
for(2 — n)
{
    for(j=0; j<result.size(); j++)
    {
        mul = res[j] * i + carry;
        res[j] = mul % 10;
        carry = mul / 10;
    }
    while(carry > 0)
    {
        finalcarry = carry % 10
        result.push(finalcarry)
        carry /= 10;
    }
}
```


Scanned by CamScanner





① 2 x 1 = 2

② 3 x 2 = 6

③ 6 x 4 = 24

res[0] = mul%10

carry = 2

carry > 0 res.push(finalcarry)

④ j=0 5 x 4 = 20

res[0] = 20%10

carry = 2

i=1 5x2 + 2 = 12

res[1] = 2

carry = 1

⑤ i=0 6 x 0 = 0

mul = res[0] * i + carry

0 * 6 + 0

Scanned by CamScanner

$j=1$ $mul = 6 \times 2 +$ carry (carry = 0)

$mul = 12$

$res[1] = mul \% 10 = 2$

$carry = 1$

$j=2$ $mul = 6 \times 1 + 1$

$= 7$

$res[2] = mul \% 10$

$carry = 0$

0	2	7
0	1	2

(6) $j=0$ $7 \times 0 = 0$

$j=1$ $7 \times 2 = 14$

$res[1] = 4$

$carry = 1$

0	1	
0	4	0

$j=2$ $7 \times 7 + 1$

$49 + 1 = 50$

$res[2] = 0$

$carry = 7$

Scanned by CamScanner

Find Duplicate in an Array

There are multiple method to find duplicate in an array.

(i) Type 1 Question → Question Contains only 1 duplicate element in an array

(ii) Type 2 Question - Questions contains More than one duplicate element.

1 Method → Brute force
Here we use two for loop.

T(→ O(N²) S(→ O(1).

2) Method - Sort & traverse
For Type 1 → first sort the array, the linearly traverse
# condition
if (a[i] == a[i+1]
return a[i]

T(→ O(N log N) S(→ O(1)
/ \
linearly traverse For sorting (Merge sort)

Scanned by CamScanner

2017-02-11 | 042-323

11

06
2017
february
saturday

January 2017

Wk	M	T	W	T	F	S	S
1							1
2	2	3	4	5	6	7	8
3	9	10	11	12	13	14	15
4	16	17	18	19	20	21	22
5	23	24	25	26	27	28	29

February 2017

Wk	M	T	W	T	F	S	S
5			1	2	3	4	5
6	6	7	8	9	10	11	12
7	13	14	15	16	17	18	19
8	20	21	22	23	24	25	26
9	27	28					

counting Approach

3) Hash Tabe → It is best four both Type 1 our Type 2 Questions.

08.00
09.00 But, major problem is, it uses extra space

10.00
11.00
12.00

TC → O(N) SC → O(N)
four traversing. four creating Hashtable

Best four single duplicate

4) Tourtoise Method
(Fast & slow pointer) → This Algorithm work only on Type 1 Question
which array contain only 1 duplicate element.

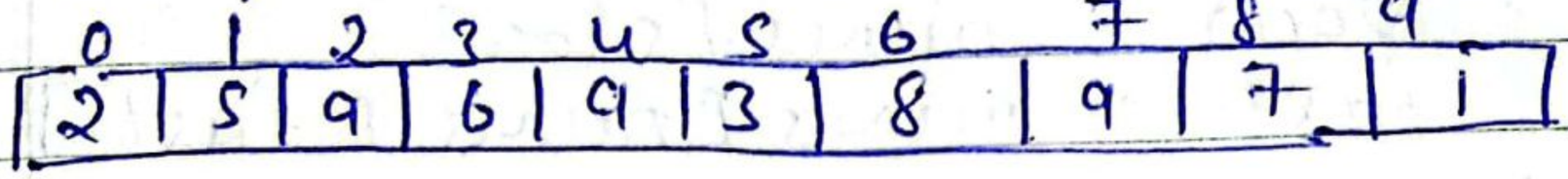
01.00
02.00
03.00
04.00 TC → O(N) SC → O(1)

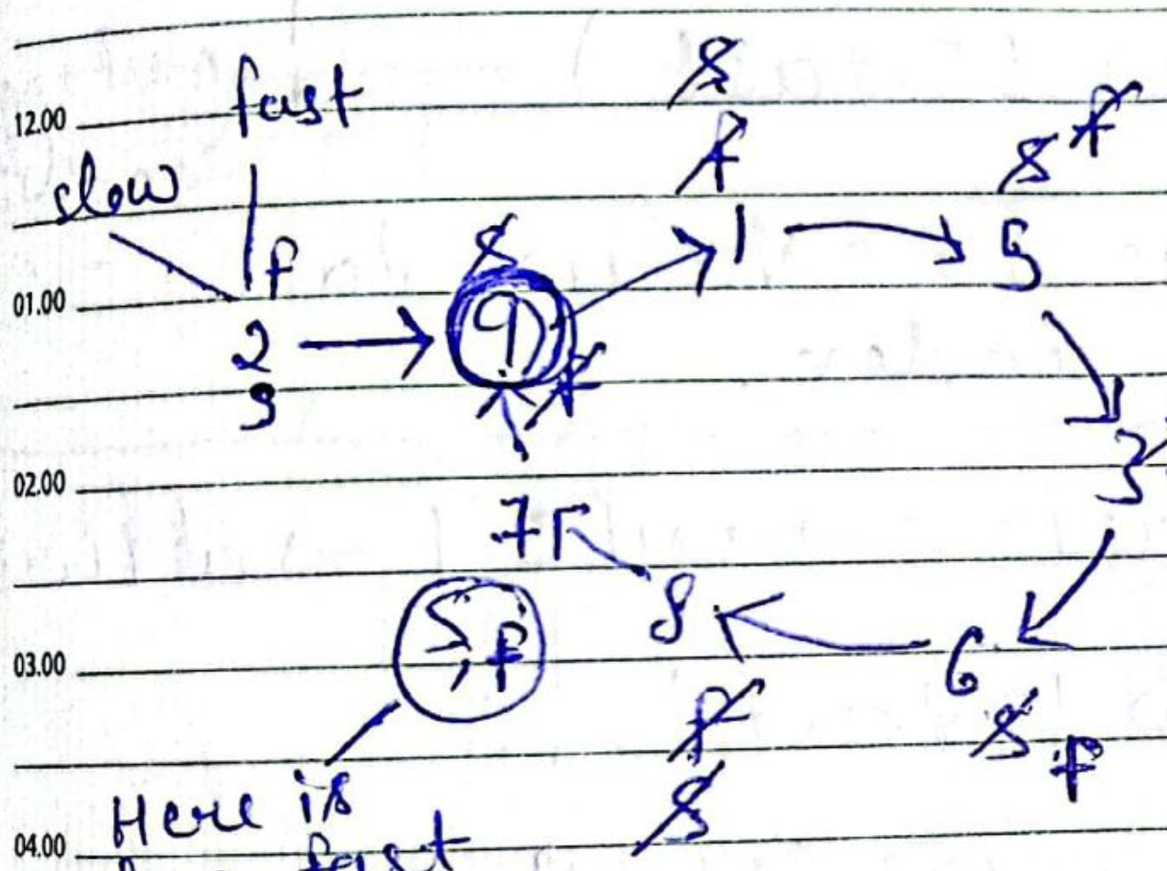
Scanned by CamScanner

3rd App Linked List Cycle method

↓

Here we use tourtois method in which we have two pointer slow & fast





slow move 1 step
fast move 2 step

if both collide, the take fast pointer on first @ element then fast, slow moved by one pointer.

TC → O(N)
SC → O(1)

2017-01-19 | 019-346 | 03

19

2017 january thursday

fast & slow Pointer.

December 2016

Wk	M	T	W	T	F	S	S
49				1	2	3	4
50	5	6	7	8	9	10	11
51	12	13	14	15	16	17	18
52	19	20	21	22	23	24	25
53	26	27	28	29	30	31	

January 2017

Wk	M	T	W	T	F	S	S
53	30	31					1
1	2	3	4	5	6	7	8
2	9	10	11	12	13	14	15
3	16	17	18	19	20	21	22
4	23	24	25	26	27	28	29

06.00 Tourtois Method

09.00 slow = nums[0], false = nums[0]

10.00
```
do{
    slow = nums[slow]
    fast = nums[nums[fast]]
} while(slow != fast)
```
meeting condition

01.00 Actually we move One value to another via index.

02.00

* if meet, fast = num[0] → at first

03.00 do while (slow != fast)

04.00
```
{slow = nums[slow];
 fast = nums[fast]
}
return slow;
```

05.00

06.00

07.00

6) Type-1 (use XOR)

1 2 3 4 -- (X) N-1 (X) duplicate NO

If num (1 — N-1) the XOR, so we take only X ← Is our Result

XOR | 1 2 3 4 (X) - - - N-1 | (X)
    | 1 2 3 4 (X) - - - N-1 |

X^X^X^ ←

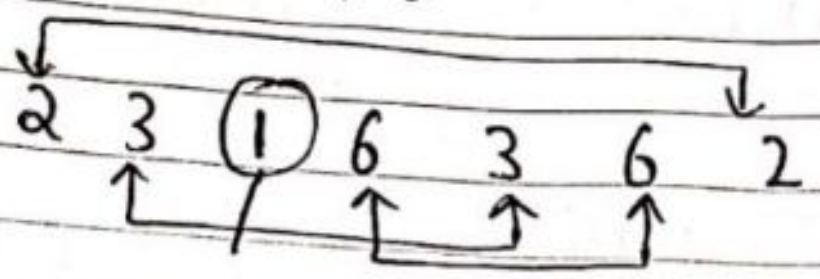
0^X = (X) our Result

MARCH

APRII

Scanned by CamScanner

Find Unique Element



1 is Unique element

Approach - 1

Count Occurance of all element

2 → 2
3 → 2
1 → (1) → those have only one occurance, that is unique element
6 → 2

2nd Approach

$2 + 3 + 1 + 6 - 3 - 6 - 2 = 1$

this could be done via XOR Method.

```
int ans = 0

for (int i = 0 ---- size; i++)
    {
        ans = ans ^ arr[i];
    }
return ans;
```


Scanned by CamScanner

Pair Sum / 2 Sum

1st Method → My Code

2nd Method → My Code

3rd Method → Here.

TC → O(n log n)

SC → O(N)

arr → | 2 | 7 | 11 | 15 | target = 2

step 1 Make pair

```
for (i = 0; i < size; i++)
{
    ans.pushback (make_pair (arr[i], i))
}
```

| (2, 0) | (7, 1) | (11, 2) | (15, 3) |

step 2 - Sort (on basis of 1st element)

ans | (2, 0) | (7, 1) | (11, 2) | (15, 3) |

i ↑ j ↑

```
while(i<j)
{
    int sum = ans[i].first + ans[j].first
    if(sum == target)
        return {ans[i].second, ans[j].second};
    else if (sum > target)
    {
        j--
    }
    else i++;
```

sum jyada aaya to kum j me -- lounge

sum kum aaya to i++ lounge

target = 9

(2,0)	(7,1)	(11,2)	(15,3)
↑ i	↑ j	↑ ✗	↑ ✗

① SUM = 2+15 > 9 j--

② SUM = 2+11 > 9 j--

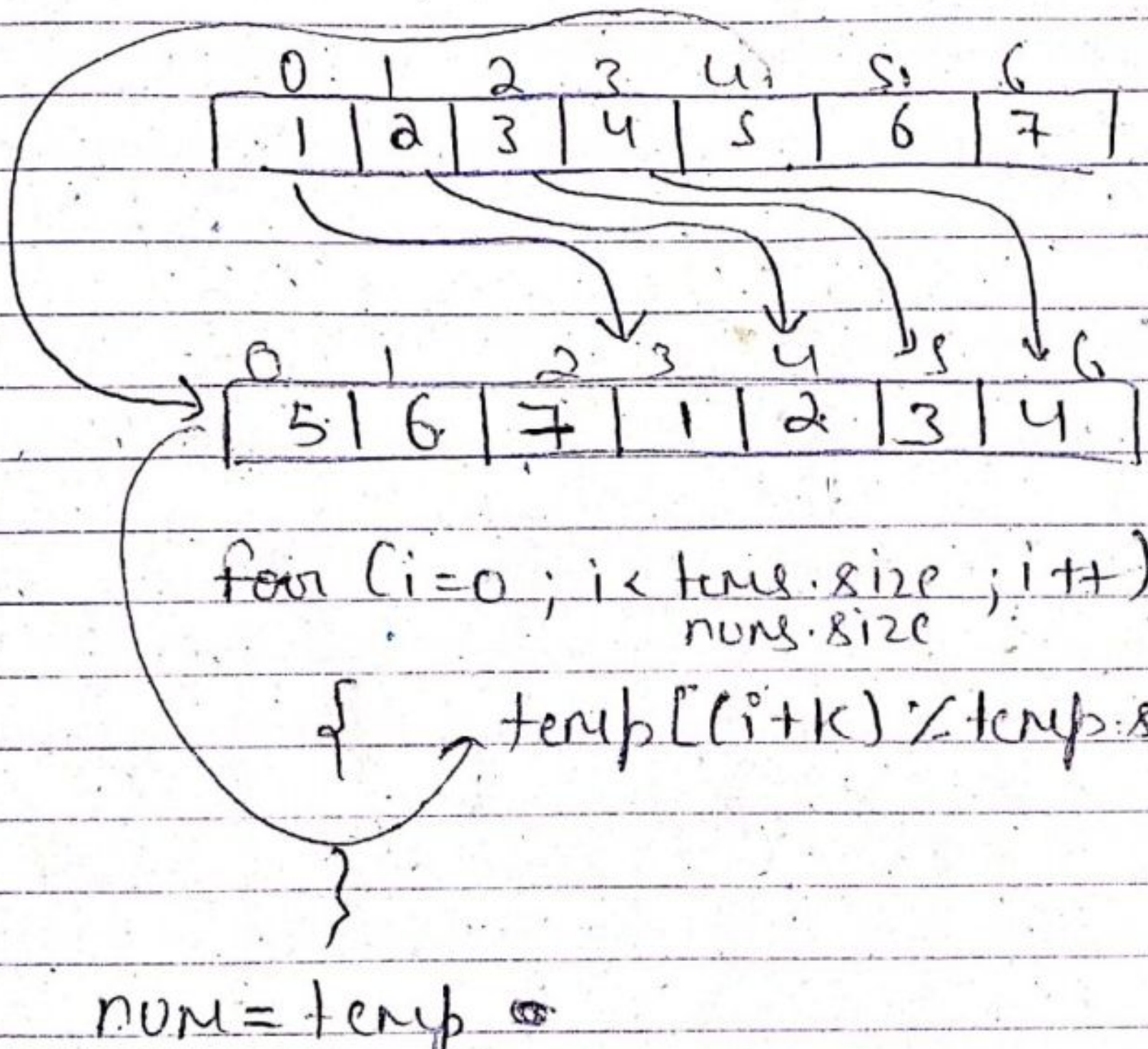
③ SUM = 2+7 == 9 return(____)

Rotate an Array by k

nums = | 1 | 2 | 3 | 4 | 5 | 6 | 7 | , k = 3

Ans → | 5 | 6 | 7 | 1 | 2 | 3 | 4 |

App-1 (Extra space)



for (i=0 ; i < nums.size ; i++)
nums.size

{ temp[(i+k) % temp.size()] = nums[i]

nums = temp ✓

① i=0 temp[(0+3)%7] = nums[0]
temp[3] = nums[0]

④ i=4 temp[(4+3)%7] = nums[4]
temp[1] = nums[4]

App-2 (without Extra space)

TC → O(N)
SC → O(1)

1	2	3	4	5	6	7

k = k % n → because

if n = 7 values, and k = 10

so if i reverse from 0 to n-k-1

n = 7, k = 10

0 to 7-10-1

-4 is invalid index [0 to -4]

① reverse (nums, 0, n-k-1)

4	3	2	1	5	6	7

n-k-1 n-k

② reverse (nums, n-k, n-1);

4	3	2	1	7	6	5

③ reverse (nums, 0, n-1)

5	6	7	1	2	3	4

2017-02-07 | 038-327 | 06

07

2017
february
tuesday

Wk	M	T	W	T	F	S	S
53/5	30	31					1
1	2	3	4	5	6	7	8
2	9	10	11	12	13	14	15
3	16	17	18	19	20	21	22
4	23	24	25	26	27	28	29

Wk	M	T	W	T	F	S	S
5				1	2	3	4
6	6	7	8	9	10	11	12
7	13	14	15	16	17	18	19
8	20	21	22	23	24	25	26
9	27	28					

Hash Map define and wants holds key / value / pair, and if find repeated value during traversing increas frequency Count

value frequency

Unordered_map <int, int > mpp;

for (int i=0; i<size; i++)
{
mpp[a[i]]++
}

or

Unordered_map <int, int > mpp

```
for( int i=0; i<size; i++)
    mpp[a[i]]++
```

```
for (auto it : mpp
{
    if (it.second > (size/2))
        return it.first
}
```

We can access via first, second

Hash Map

08.00

1) Initialization - unordered_map<int, int> mpp.

09.00

2) Find in mpp

10.00 if (mpp.find(target - nums[i]) != mpp.end())

11.00 This goona return an iterator if find and find target - nums[i] value.

12.00 return only first int value

01.00

four second value

mpp[nums[i]] = i

02.00

03.00