Bit Manipulation Challenges

Challenge 1

Write a program to find a unique number in an array where all numbers except one, are present twice.

Hint: A ⊕ B ⊕ B ⊕ A ⊕ C = C. All those numbers which occur twice will get nullified after ⊕ operation.

Sample Test Case:

Input:

1	2	3	4	1	2	3

Output: 4

Code

```
#include<iostream>
using namespace std;
int unique(int arr[], int n) {
    int xorsum = 0;
    for (int i = 0; i < n; i++) {
        xorsum = xorsum ^ arr[i];
    }
    return xorsum;
}
int main() {
    int arr[] = {1, 2, 3, 4, 1, 2, 3};
    cout << unique(arr, 7) << endl;
    return 0;
}
```

Challenge 2

Q2. Write a program to find 2 unique numbers in an array where all numbers except two, are present twice.

Logic

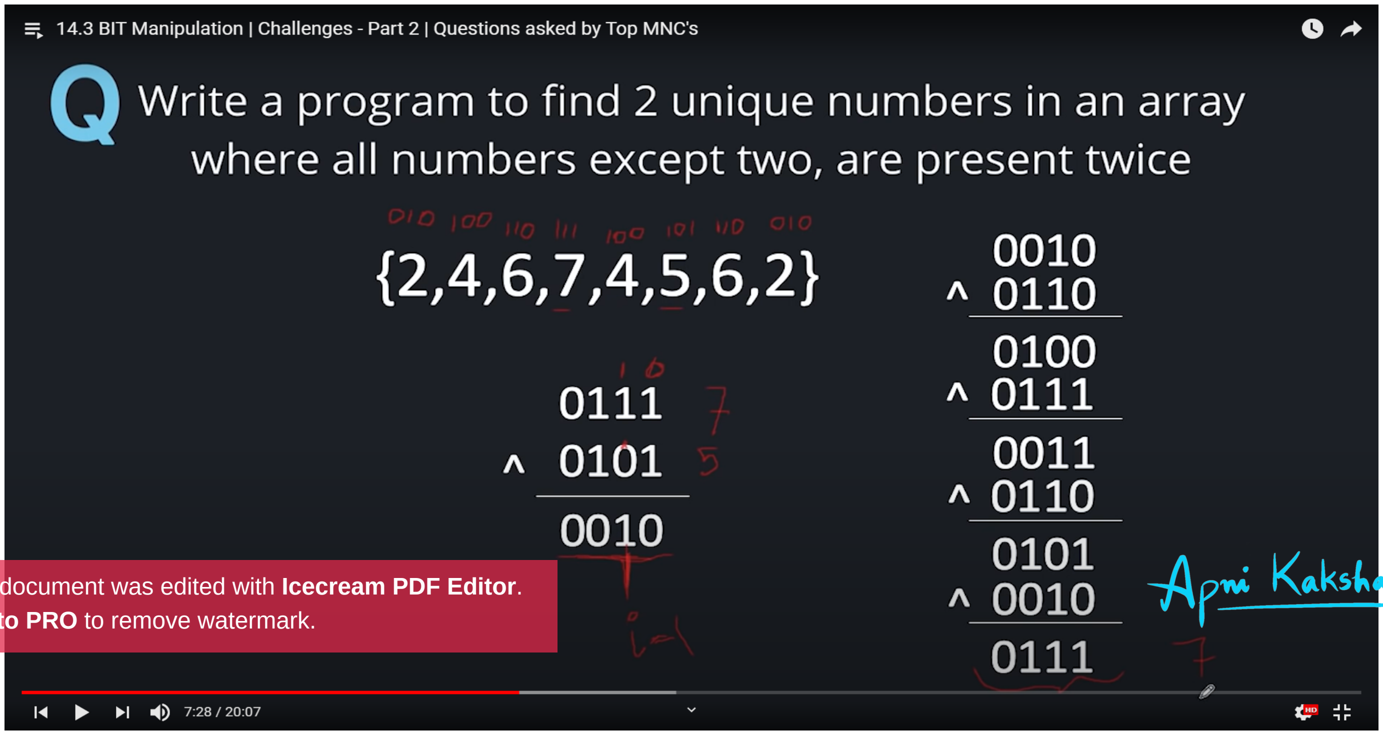
1. Take XOR of all the elements and let that xor value be x. All the repeating elements will get nullified and xor of only two unique elements will last. (as a⊕a = 0).
2. There will be at least one bit set in x. Using that set bit, divide the original set of numbers into two sets
   a. First set which contains all the elements with that bit set.
   b. Second set which contains all the elements with that bit unset.
3. Take xor of both the sets individually, and let those xor values be x1 and x2.
4. Voila, x1 and x2 are our unique numbers. Reason: both the above sets contain one of the unique elements and rest elements of the sets occur twice which will get nullified after ⊕ operation.

Sample Test Case:

Input:

2	4	6	7	4	5	6	2

Output: 5 7



14.3 BIT Manipulation | Challenges - Part 2 | Questions asked by Top MNC's

Q Write a program to find 2 unique numbers in an array where all numbers except two, are present twice

```
010 100 110 111 100 101 110 010
{2,4,6,7,4,5,6,2}                         0010
                                        ^ 0110
                                          ----
          0111  7                         0100
        ^ 0101  5                       ^ 0111
          ----                            ----
          0010                            0011
                                        ^ 0110
                                          ----
                                          0101
                                        ^ 0010
                                          ----
                                          0111  7
```

7:28 / 20:07

Code

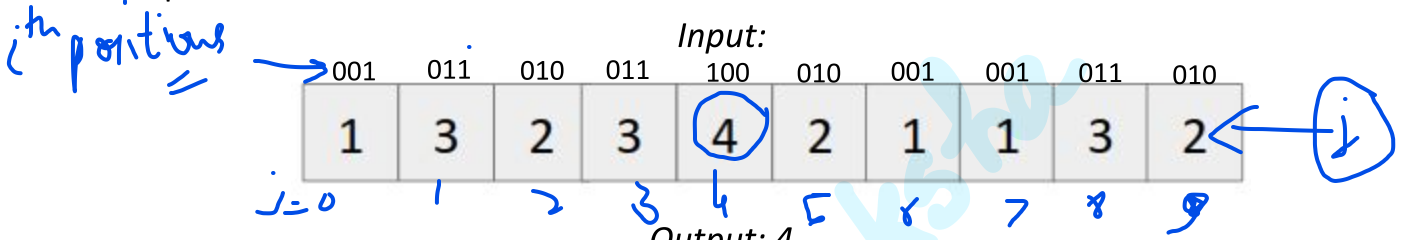
```
#include<iostream>
using namespace std;
int setBit(int n, int pos) {
    return ((n & (1 << pos)) != 0);
}
void unique(int arr[], int n) {
    int xorsum = 0;
    for (int i = 0; i < n; i++) {
        xorsum = xorsum ^ arr[i];
    }
    int tempxor = xorsum;
    int setbit = 0;
    int pos = 0;
    while (setbit != 1) {
        setbit = xorsum & 1;
        pos++;
        xorsum = xorsum >> 1;
    }
    int newxor = 0;
    for (int i = 0; i < n; i++) {
        if (setBit(arr[i], pos - 1)) {
            newxor = newxor ^ arr[i];
        }
    }
    cout << newxor << endl;
    cout << (tempxor ^ newxor) << endl;

}
int main() {
    int arr[] = {1, 2, 3, 1, 2, 3, 5, 7};
    unique(arr, 8);
    return 0;
}
```

Challenge 3

Q3. Write a program to find a unique number in an array where all numbers except one, are present thrice.

Sample Test Case:

Input:

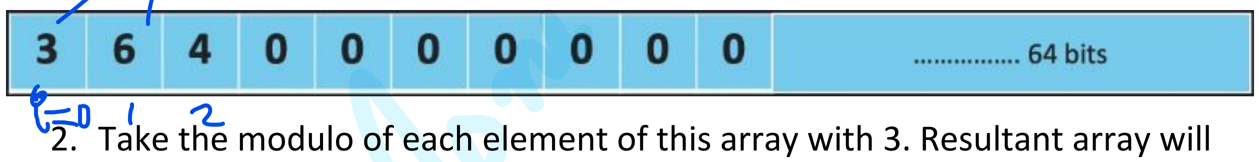


Output: 4

Logic

Multiples of 3 of no use.

1. We will maintain an array of 64 size which will store the number of times i th bit has occurred in the array.



2. Take the modulo of each element of this array with 3. Resultant array will represent the binary representation of the unique number.
3. Convert that binary number to decimal number and output it.

For $\boxed{i=2}$ → sum=4.

→ This will give ans.

int result=0

result = setbit(result, i);

return result;

Code

```
#include<iostream>
using namespace std;

bool getBit(int n, int pos) {
    return ((n & (1 << pos)) != 0);
}
int setBit( int n, int pos) {
    return (n | (1 << pos));
}
int unique(int arr[], int n) {
    int result = 0;
    for (int i = 0; i < 64; i++) {
        int sum = 0;
        for (int j = 0; j < n; j++) {
            if (getBit(arr[j], i)) {
                sum++;
            }
        }
        if (sum % 3 != 0) {
            result = setBit(result, i);
        }
    }
    return result;
}
int main() {
    int arr[] = {1, 2, 3, 4, 1, 2, 3, 1, 2, 3};
    cout << unique(arr, 10) << endl;
    return 0;
}
```