Find lost element from a duplicated array

Given two arrays which are duplicates of each other except one element, that is one element from one of the array is missing, we need to find that missing element.

Examples:

```
Input:  arr1[] = {1, 4, 5, 7, 9}
        arr2[] = {4, 5, 7, 9}
Output: 1
1 is missing from second array.

Input: arr1[] = {2, 3, 4, 5}
       arr2[] = {2, 3, 4, 5, 6}
Output: 6
6 is missing from first array.
```

We strongly recommend you to minimize your browser and try this yourself first.

One **simple solution** is to iterate over arrays and check element by element and flag the missing element when an unmatch is found, but this solution requires linear time over size of array.

Another **efficient solution** is based on binary search approach. Algorithm steps are as follows:

1. Start binary search in bigger array and get mid as (lo + hi) / 2
2. If value from both array is same then missing element must be in right part so set lo as mid
3. Else set hi as mid because missing element must be in left part of bigger array if mid elements are not equal.
4. Special case are handled separately as for single element and zero element array, single element itself will be the missing element. If first element itself is not equal then that element will be the missing element./li>

Below is C++ implementation of above steps

```
// C++ program to find missing element from same
// arrays (except one missing element)
#include <bits/stdc++.h>
using namespace std;

// Funtion to find missing element based on binary
// search approach.  arr1[] is of larger size and
// N is size of it.  arr1[] and arr2[] are assumed
// to be in same order.
int findMissingUtil(int arr1[], int arr2[], int N)
{
   // special case, for only element which is
   // missing in second array
   if (N == 1)
      return arr1[0];

   // special case, for first element missing
   if (arr1[0] != arr2[0])
      return arr1[0];

   // Initialize current corner points
   int lo = 0,  hi = N - 1;

   // loop until lo < hi
   while (lo < hi)
   {
      int mid = (lo + hi) / 2;

      // If element at mid indices are equal
      // then go to right subarray
```

```
        // then go to right subarray
        if (arr1[mid] == arr2[mid])
            lo = mid;
        else
            hi = mid;

        // if lo, hi becomes contiguous, break
        if (lo == hi - 1)
            break;
    }

    // missing element will be at hi index of
    // bigger array
    return arr1[hi];
}

// This function mainly does basic error checking
// and calls findMissingUtil
void findMissing(int arr1[], int arr2[], int M, int N)
{
    if (N == M-1)
        cout << "Missing Element is "
        << findMissingUtil(arr1, arr2, M) << endl;
    else if (M== N-1)
        cout << "Missing Element is "
        << findMissingUtil(arr2, arr1, N) << endl;
    else
        cout << "Invalid Input";
}

// Driver Code
int main()
{
    int arr1[] = {1, 4, 5, 7, 9};
    int arr2[] = {4, 5, 7, 9};

    int M = sizeof(arr1) / sizeof(int);
    int N = sizeof(arr2) / sizeof(int);

    findMissing(arr1, arr2, M, N);

    return 0;
}
```

Output :

```
Missing Element is 1
```

What if input arrays are not in same order?

In this case, missing element is simply XOR of all elements of both arrays. Thanks to Yolo Song for suggesting this.

```
// C++ program to find missing element from one array
// such that it has all elements of other array except
// one.  Elements in two arrays can be in any order.
#include <bits/stdc++.h>
using namespace std;

// This function mainly does XOR of all elements
// of arr1[] and arr2[]
void findMissing(int arr1[], int arr2[], int M,
                 int N)
{
    if (M != N-1 && N != M-1)
    {
        cout << "Invalid Input";
        return;
    }

    // Do XOR of all element
    int res = 0;
    for (int i=0; i<M; i++)
      res = res^arr1[i];
    for (int i=0; i<N; i++)
      res = res^arr2[i];

    cout << "Missing element is " << res;
}

// Driver Code
int main()
{
    int arr1[] = {4, 1, 5, 9, 7};
    int arr2[] = {7, 5, 9, 4};

    int M = sizeof(arr1) / sizeof(int);
    int N = sizeof(arr2) / sizeof(int);

    findMissing(arr1, arr2, M, N);

    return 0;
}
```

Output :

```
Missing Element is 1
```