```
/ CPP implementation of the approach
#include<bits/stdc++.h>
using namespace std;
// Shows function to return the missing element
int findMissing(int array[], int n1){
  int low = 0, high = n1 - 1;
  int mid1;
  while (high > low){
    mid1 = low + (high - low) / 2;
    // Verify if middle element is consistent
    if (array[mid1] - mid1 == array[0]){
      // Here, no inconsistency till middle elements
      // When missing element is just after
      // the middle element
      if (array[mid1 + 1] - array[mid1] > 1)
        return array[mid1] + 1;
      else{
        // Go right
        low = mid1 + 1;
      }
    }
    else{
      // Here inconsistency found
      // When missing element is just before
      // the middle element
      if (array[mid1] - array[mid1 - 1] > 1)
        return array[mid1] - 1;
      else{
        // Go left
        high = mid1 - 1;
      }
    }
  }
  // Here, no missing element found
  return -1;
}
// Driver code
int main(){
  int array[] = { -9, -8, -6, -5, -4, -3, -2, -1, 0 };
  int n1 = sizeof(array)/sizeof(array[0]);
  cout <<"The Missing Element:" <<(findMissing(array, n1));
}
```