```cpp

//Name - Gourav..SID-21104066..Branch-EE..Assignment-Binary Search Tree.
#include <bits/stdc++.h>
using namespace std;

// 1. creating a BST from an array.
class Node
{
public:
    int data;
    Node *left;
    Node *right;
};

Node *createNode(int data)
{
    Node *node = new Node;
    node->data = data;
    node->left = node->right = NULL;
    return node;
};

Node *bSTconstruct(Node *root, int data)
{
    if (root == NULL)
    {
        return createNode(data);
    }

    if (data < root->data)
    {
        root->left = bSTconstruct(root->left, data);
    }

    else
    {
        root->right = bSTconstruct(root->right, data);
    }

    return root;
};


void inOrder(Node *root){
    if(root == NULL){
        return;
    }
    inOrder(root->left);
    cout<<root->data<<" ";
    inOrder(root->right);

}
Node* inOrderSucc(Node* root){
    Node* curr = root;
    while(curr && curr->left!=NULL){
        curr = curr->left;
    }
    return curr;

}

//2. Function to delete the node from the BST
Node* bSTDelete(Node* root, int data)
{
    if(data < root->data){
        root->left = bSTDelete(root->left,data);
```

```
    }
    else if(data > root->data){
        root->right = bSTDelete(root->right,data);
    }
    else{
        if(root->left==NULL){
            Node* temp = root->right;
            free(root);
            return temp;

        }
        else if(root->right==NULL){
            Node* temp = root->left;
            free(root);
            return temp;
        }

        Node* temp = inOrderSucc(root->right);
        root->data = temp->data;
        root->right = bSTDelete(root->right,temp->data);
    }
    return root;
}

// function to delete the element from the array.
int deleteElement(int arr[],int n, int val){
    for(int i=0;i<n;i++){
        if(arr[i]==val){
            break;
        }

        if(i<n){
            n = n-1;
            for(int j = i; j<n; j++){
                arr[j] = arr[j+1];
            }

        }

    }
    return n;



}


int main()
{
    int arr1[] = {10, 16, 52, 36, 25, 71};
    Node *root = NULL;
    root = bSTconstruct(root, 10);
    bSTconstruct(root, 16);
    bSTconstruct(root, 52);
    bSTconstruct(root, 36);
    bSTconstruct(root, 25);
    bSTconstruct(root, 71);

    inOrder(root);
    cout<<endl;
    cout << deleteElement(arr1,6, 52);

}
// 3. Space Complexity of Both BST and Array is O(n) where n is the number of nodes
```