

```

1
2 //Name - Gourav..SID-21104066..Branch-EE..Assignment-Binary Search Tree.
3 #include <bits/stdc++.h>
4 using namespace std;
5
6 // 1. creating a BST from an array.
7 class Node
8 {
9 public:
10     int data;
11     Node *left;
12     Node *right;
13 };
14
15 Node *createNode(int data)
16 {
17     Node *node = new Node;
18     node->data = data;
19     node->left = node->right = NULL;
20     return node;
21 };
22
23 Node *bSTconstruct(Node *root, int data)
24 {
25     if (root == NULL)
26     {
27         return createNode(data);
28     }
29
30     if (data < root->data)
31     {
32         root->left = bSTconstruct(root->left, data);
33     }
34
35     else
36     {
37         root->right = bSTconstruct(root->right, data);
38     }
39
40     return root;
41 };
42
43
44 void inOrder(Node *root){
45     if(root == NULL){
46         return;
47     }
48     inOrder(root->left);
49     cout<<root->data<<" ";
50     inOrder(root->right);
51 }
52
53 Node* inOrderSucc(Node* root){
54     Node* curr = root;
55     while(curr && curr->left!=NULL){
56         curr = curr->left;
57     }
58     return curr;
59 }
60
61
62 //2. Function to delete the node from the BST
63 Node* bSTDelete(Node* root, int data)
64 {
65     if(data < root->data){
66         root->left = bSTDelete(root->left,data);

```

```

67     }
68     else if(data > root->data){
69         root->right = bSTDelete(root->right,data);
70     }
71     else{
72         if(root->left==NULL){
73             Node* temp = root->right;
74             free(root);
75             return temp;
76         }
77     }
78     else if(root->right==NULL){
79         Node* temp = root->left;
80         free(root);
81         return temp;
82     }
83
84     Node* temp = inOrderSucc(root->right);
85     root->data = temp->data;
86     root->right = bSTDelete(root->right,temp->data);
87 }
88 return root;
89 }
90
91 // function to delete the element from the array.
92 int deleteElement(int arr[],int n, int val){
93     for(int i=0;i<n;i++){
94         if(arr[i]==val){
95             break;
96         }
97     }
98     if(i<n){
99         n = n-1;
100         for(int j = i; j<n; j++){
101             arr[j] = arr[j+1];
102         }
103     }
104 }
105
106 }
107 return n;
108
109
110
111 }
112
113
114 int main()
115 {
116     int arr1[] = {10, 16, 52, 36, 25, 71};
117     Node *root = NULL;
118     root = bSTconstruct(root, 10);
119     bSTconstruct(root, 16);
120     bSTconstruct(root, 52);
121     bSTconstruct(root, 36);
122     bSTconstruct(root, 25);
123     bSTconstruct(root, 71);
124
125     inOrder(root);
126     cout<<endl;
127     cout << deleteElement(arr1,6, 52);
128
129 }
130 // 3. Space Complexity of Both BST and Array is O(n) where n is the number of nodes

```