

```

1  #include <iostream>
2  using namespace std;
3
4  class Node
5  {
6  public:
7      int data;
8      Node *left;
9      Node *right;
10
11      Node(int value)
12      {
13          data = value;
14          left = right = nullptr;
15      }
16  };
17
18  class BST
19  {
20  public:
21      Node *root;
22      BST()
23      {
24          root = nullptr;
25      }
26      Node *findmin(Node *r)
27      {
28          if (r == nullptr)
29          {
30              return nullptr;
31          }
32          else if (r->left == nullptr)
33          {
34              return r;
35          }
36          else
37          {
38              return findmin(r->left);
39          }
40      }
41      Node *findmax(Node *r)
42      {
43          if (r == nullptr)
44          {
45              return nullptr;
46          }
47          else if (r->right == nullptr)
48          {
49              return r;
50          }
51          else
52          {
53              return findmax(r->right);
54          }
55      }
56      Node *Delete(Node *r, int key)
57      {
58          if (r == nullptr) // Empty tree
59          {
60              return nullptr;
61          }
62          if (key < r->data) // item is in left subtree
63          {
64              r->left = Delete(r->left, key);
65          }
66          else if (key > r->data) // item is in right subtree
67          {
68              r->right = Delete(r->right, key);
69          }
70          else
71          {
72              if (r->left == nullptr && r->right == nullptr) // Leaf node
73              {
74                  r = nullptr;
75              }
76              else if (r->left != nullptr && r->right == nullptr) // one child on the left
77              {
78                  r->data = r->left->data;
79                  delete r->left;
80                  r->left = nullptr;
81              }
82              else if (r->left == nullptr && r->right != nullptr) // one child on the right
83              {
84                  r->data = r->right->data;
85                  delete r->right;
86                  r->right = nullptr;
87              }
88              else
89              {
90                  Node *max = findmax(r->left); // two children
91                  r->data = max->data;
92                  r->left = Delete(r->left, max->data);
93              }
94          }
95          return r;
96      }
97  };

```