

```

1  include <iostream>
2  using namespace std;
3
4  class Node {
5      int *keys;
6      int t;
7      Node **C;
8      int n;
9      bool leaf;
10
11 public:
12     Node(int tt, bool lleaf);
13
14     void insertNonFull(int k);
15     void splitChild(int i, Node *y);
16     void traverse();
17
18     friend class BTree;
19 ;
20
21 class BTree {
22     Node *root;
23     int t;
24
25 public:
26     BTree(int tt) {
27         root = NULL;
28         t = tt;
29     }
30
31     void traverse() {
32         if (root) root->traverse();
33     }
34
35     void insert(int k);
36 ;
37
38 Node::Node(int tt, bool lleaf) {
39     t = tt;
40     leaf = lleaf;
41
42     keys = new int[2 * t - 1];
43     C = new Node *[2 * t];
44
45     n = 0;
46
47
48 void Node::traverse() {
49     int i;
50     for (i = 0; i < n; i++) {
51         if (leaf == false) C[i]->traverse();
52         cout << " " << keys[i];
53     }
54
55     if (leaf == false) C[i]->traverse();
56
57
58 void BTree::insert(int k) {
59     if (root == NULL) {
60         root = new Node(t, true);
61         root->keys[0] = k;
62         root->n = 1;
63     } else {
64         if (root->n == 2 * t - 1) {
65             Node *s = new Node(t, false);
66
67             s->C[0] = root;
68
69             s->splitChild(0, root);
70
71             int i = 0;
72             if (s->keys[0] < k) i++;
73             s->C[i]->insertNonFull(k);
74
75             root = s;
76         } else
77             root->insertNonFull(k);
78     }
79
80
81 void Node::insertNonFull(int k) {
82     int i = n - 1;
83
84     if (leaf == true) {
85         while (i >= 0 && keys[i] > k) {
86             keys[i + 1] = keys[i];
87             i--;
88         }
89
90         keys[i + 1] = k;
91         n = n + 1;
92     } else {
93         while (i >= 0 && keys[i] > k) i--;
94
95         if (C[i + 1]->n == 2 * t - 1) {
96             splitChild(i + 1, C[i + 1]);
97
98             if (keys[i + 1] < k) i++;
99         }
100         C[i + 1]->insertNonFull(k);
101     }
102
103
104 void Node::splitChild(int i, Node *y) {
105     Node *z = new Node(y->t, y->leaf);
106     z->n = t - 1;
107
108     for (int j = 0; j < t - 1; j++)
109         z->keys[j] = y->keys[j + t];
110
111     if (y->leaf == false)
112         for (int j = 0; j < t; j++)
113             z->C[j] = y->C[j + t];
114
115     y->n = t - 1;
116     for (int j = n; j >= i + 1; j--)
117         C[j + 1] = C[j];
118
119     C[i + 1] = z;
120
121     for (int j = n - 1; j >= i; j--)
122         keys[j + 1] = keys[j];
123
124     keys[i] = y->keys[t - 1];
125     n = n + 1;
126
127
128 int main() {
129     int o;
130     cout << "Enter B Tree Order: ";
131     cin >> o;
132     BTree t(o);
133
134     int k, ele;
135     cout << "Number of elements to insert: ";
136
137     cin >> k;
138     cout << "Enter " << k << " elements \n";
139     while (k--) {
140         cin >> ele;
141         t.insert(ele);
142     }
143
144     cout << "The B-tree is: ";
145     {
146         // * In a different block because it doesn't
147         t.traverse();
148         cout << endl;
149     }
150

```



Terminal



Enter B Tree Order: 3

Number of elements to insert: 10

Enter 10 elements

2

1

4

5

6

7

8

9

10

11

The B-tree is: 1 2 4 5 6 7 8 9 10 11