```
 1: /*-------Binary Search Tree-------*/
 2:
 3: #include<iostream>
 4: using namespace std;
 5:
 6: struct node
 7: {
 8:     int data;
 9:     node * left;
10:     node * right;
11: };
12:
13: class SearchTree
14: {
15:     public:
16:         node * root;
17:         SearchTree()
18:         {
19:             root = NULL;
20:         }
21:
22:         void insert(int d)
23:         {
24:             node * pointer, * last_node, * temp = new node();
25:
26:             temp -> data = d;
27:             temp -> left = NULL;
28:             temp -> right = NULL;
29:
30:             pointer = root;
31:             if(root == NULL)
32:             {
33:                 root = temp;
34:                 return;
35:             }
36:             while(pointer != NULL)
37:             {
38:                 last_node = pointer;
39:                 if(pointer -> data > d)
40:                 {
41:                     pointer = pointer -> left;
42:                 }
43:                 else
44:                 {
45:                     pointer = pointer -> right;
46:                 }
```

```
47:             }
48:             if(last_node -> data > d)
49:             {
50:                 last_node -> left = temp;
51:             }
52:             else
53:             {
54:                 last_node -> right = temp;
55:             }
56:         }
57:         void search(int y)
58:         {
59:             node * pointer;
60:             pointer = root;
61:             while(pointer != NULL )
62:             {
63:                 if(pointer -> data == y)
64:                 {
65:                     cout << y << " is present. \n";
66:                 }
67:                 else if(pointer -> data > y)
68:                 {
69:                     pointer = pointer -> left;
70:                 }
71:                 else
72:                 {
73:                     pointer = pointer -> right;
74:                 }
75:             }
76:             if(pointer == NULL)
77:             {
78:                 cout << y << " is absent. \n";
79:             }
80:         }
81:         void print(node *k)
82:         {
83:             if(k != NULL)
84:             {
85:                 print(k -> left);
86:                 cout << k -> data << " ";
87:                 print(k -> right);
88:             }
89:         }
90: };
91:
92: int main()
```

```
 93: {
 94:     SearchTree tree;
 95:     int num;
 96:     cout << "How many nodes you want to add? \n";
 97:     cin >> num;
 98:     int data;
 99:     cout << "Enter the nodes: \n";
100:     for(int i = 0; i < num; i++)
101:     {
102:         cin >> data;
103:         tree.insert(data);
104:     }
105:
106:     tree.print(tree.root);
107:
108:     return 0;
109:
110: }
```