main.cpp

```cpp
#include <bits/stdc++.h>
using namespace std;

class BTreeNode
{
    int *keys;
    BTreeNode **child;
    int t;
    int n;
    bool leaf;
    public:
        BTreeNode(int t, bool leaf);
        void traverse();
        void insertNonFull(int k);
        void splitChild(int i, BTreeNode *y);
        friend class BTree;
};

class BTree
{
    BTreeNode *root;
    int t;
    public:
        BTree(int _t){
            root = NULL;
            t = _t;
        }
```

main.cpp

```
        void insert(int k);
};

BTreeNode::BTreeNode(int t1, bool leaf1)
{
    t = t1;
    leaf = leaf1;
    keys = new int[2*t-1];
    child = new BTreeNode *[2*t];
    n = 0;
}


void BTree::insert(int k)
{
    if(root == NULL)
    {
        root = new BTreeNode(t, true);
        root->keys[0] = k;
        root->n = 1;
    }
    else{
        if(root->n == 2*t-1)
        {
            BTreeNode *s = new BTreeNode(t, false);
            s->child[0] = root;
            s->splitChild(0, root);
```

main.cpp

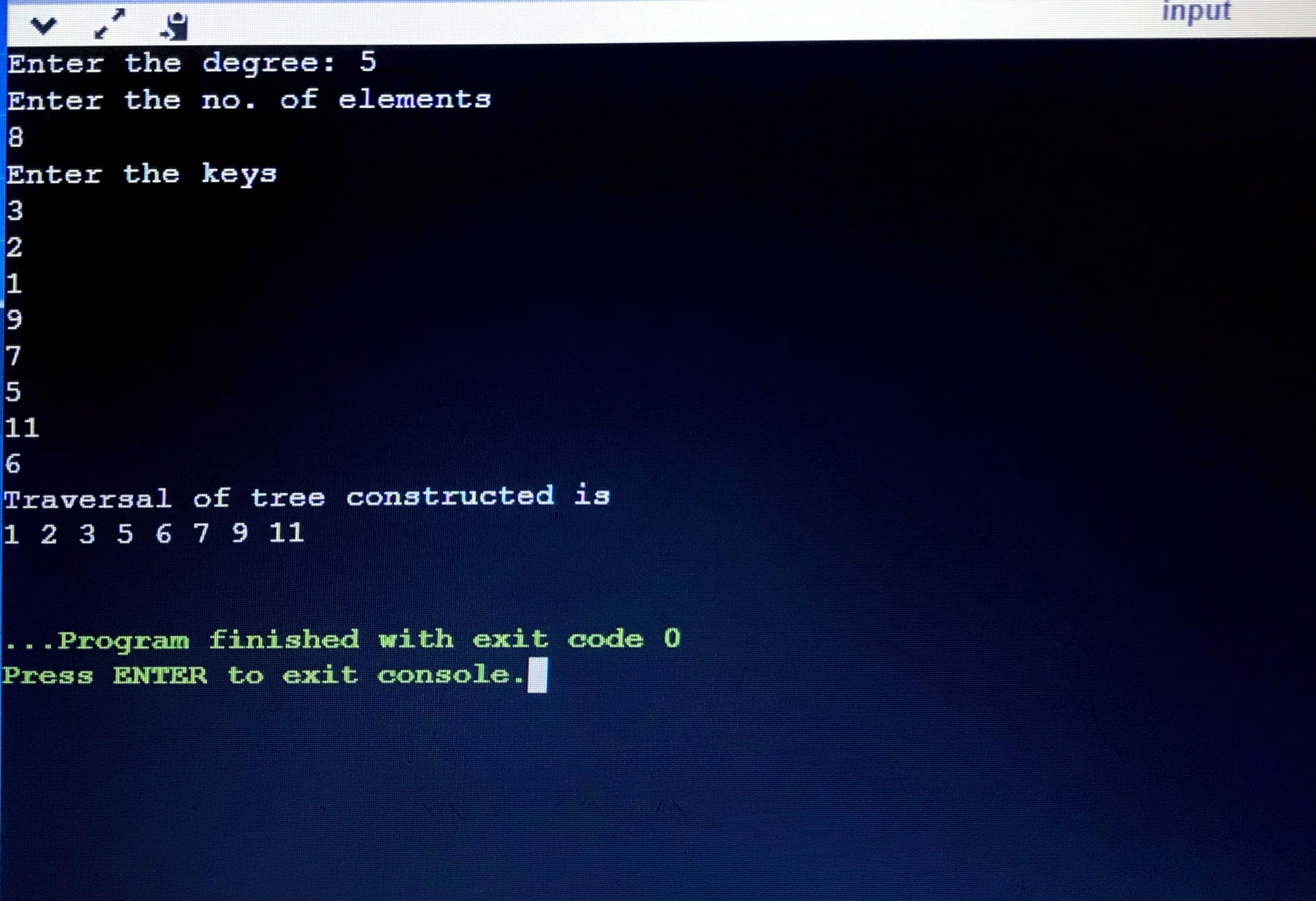
```
66              else
67                  root->insertNonFull(k);
68          }
69      }
70
71      void BTreeNode::insertNonFull(int k)
72      {
73          int i = n-1;
74          if(leaf == true)
75          {
76              while(i>=0 && keys[i] > k)
77              {
78                  keys[i+1] = keys[i];
79                  i--;
80              }
81              keys[i+1] = k;
82              n = n + 1;
83          }
84          else{
85              while(i>=0 && keys[i]>k)
86                  i--;
87              if(child[i+1]->n == 2*t-1)
88              {
89                  splitChild(i+1, child[i+1]);
90                  if(keys[i+1]<k)
91                      i++;
92              }
```

main.cpp

```
100         z->n = t-1;
101         for (int j = 0; j < t-1; j++)
102                 z->keys[j] = y->keys[j+t];
103
104         if(y->leaf == false)
105         {
106             for(int j=0; j<t; j++)
107                 z->child[j] = y->child[j+t];
108         }
109         y->n = t-1;
110         for(int j=n; j>=i+1; j--)
111             child[j+1] = child[j];
112         child[i+1] = z;
113         for (int j = n-1; j >= i; j--)
114                 keys[j+1] = keys[j];
115
116             keys[i] = y->keys[t-1];
117
118             n = n + 1;
119 }
120
121 void BTreeNode::traverse()
122 {
123     //cout<<endl;
124     int i;
125     for(i=0; i<n; i++)
126     {
```

main.cpp

```
132             child[i]->traverse();
133
134         //cout<<endl;
135     }
136
137
138     int main()
139     {
140         int d;
141         cout<<"Enter the degree: ";
142         cin>>d;
143             BTree t(d);
144         int n,k;
145         cout<<"Enter the no. of elements"<<endl;
146         cin>>n;
147         cout<<"Enter the keys"<<endl;
148         for(int i=0; i<n; i++)
149         {
150             cin>>k;
151             t.insert(k);
152         }
153         cout << "Traversal of tree constructed is\n";
154             t.traverse();
155         cout<<endl;
156         return 0;
157
158     }
```


input
Enter the degree: 5
Enter the no. of elements
8
Enter the keys
3
2
1
9
7
5
11
6
Traversal of tree constructed is
1 2 3 5 6 7 9 11
...Program finished with exit code 0
Press ENTER to exit console.