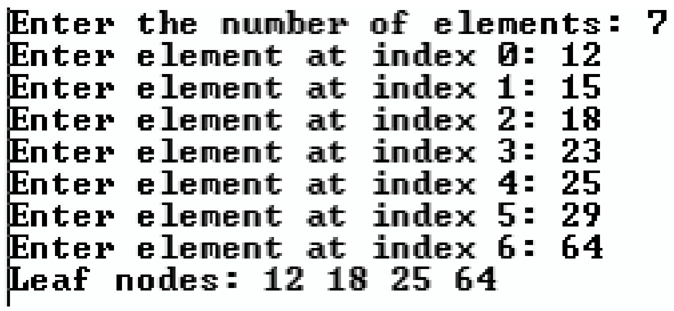
```cpp
#include <iostream>
using namespace std;

class BTNode{

    private:
        int nodeid;
        int data;
        int levelNum;
        BTNode* leftChildPtr;
        BTNode* rightChildPtr;

    public:

        BTNode(){}

        void setNodeId(int id){
            nodeid = id;
        }

        int getNodeId(){
            return nodeid;
        }

        void setData(int d){
            data = d;
        }

        int getData(){
            return data;
        }

        void setLevelNum(int level){
            levelNum = level;
        }

        int getLevelNum(){
            return levelNum;
        }

        void setLeftChildPtr(BTNode* ptr){
            leftChildPtr = ptr;
        }

        void setRightChildPtr(BTNode* ptr){
            rightChildPtr = ptr;
        }

        BTNode* getLeftChildPtr(){
            return leftChildPtr;
        }

        BTNode* getRightChildPtr(){
            return rightChildPtr;
        }

        int getLeftChildID(){
            if (leftChildPtr == 0)
                return -1;

            return leftChildPtr->getNodeId();
        }

        int getRightChildID(){
            if (rightChildPtr == 0)
                return -1;

            return rightChildPtr->getNodeId();
        }
};



class BinarySearchTree{

    private:
        int numNodes;
        BTNode* arrayOfBTNodes;
        int rootNodeID;


    public:

        BinarySearchTree(int n){
            numNodes = n;
            arrayOfBTNodes = new BTNode[numNodes];

            for (int index = 0; index < numNodes; index++){

                arrayOfBTNodes[index].setNodeId(index);
                arrayOfBTNodes[index].setLeftChildPtr(0);
                arrayOfBTNodes[index].setRightChildPtr(0);
                arrayOfBTNodes[index].setLevelNum(-1);

            }
        }


        void setLeftLink(int upstreamNodeID, int downstreamNodeID){
            arrayOfBTNodes[upstreamNodeID].setLeftChildPtr(&arrayOfBTNodes[
            downstreamNodeID]);
        }

        void setRightLink(int upstreamNodeID, int downstreamNodeID){
            arrayOfBTNodes[upstreamNodeID].setRightChildPtr(&arrayOfBTNodes[
            downstreamNodeID]);
        }


        void constructBSTree(int* array){

            int leftIndex = 0;
            int rightIndex = numNodes-1;
            int middleIndex = (leftIndex + rightIndex)/2;

            rootNodeID = middleIndex;
            arrayOfBTNodes[middleIndex].setData(array[middleIndex]);

            ChainNodes(array, middleIndex, leftIndex, rightIndex);

        }


        void ChainNodes(int* array, int middleIndex, int leftIndex, int rightIndex){


            if (leftIndex < middleIndex){
                int rootIDLeftSubtree = (leftIndex + middleIndex-1)/2;
```

```cpp
                setLeftLink(middleIndex, rootIDLeftSubtree);
                arrayOfBTNodes[rootIDLeftSubtree].setData(array[rootIDLeftSubtree]);
                ChainNodes(array, rootIDLeftSubtree, leftIndex, middleIndex-1);
            }


            if (rightIndex > middleIndex){
                int rootIDRightSubtree = (rightIndex + middleIndex + 1)/2;
                setRightLink(middleIndex, rootIDRightSubtree);
                arrayOfBTNodes[rootIDRightSubtree].setData(array[rootIDRightSubtree]);
                ChainNodes(array, rootIDRightSubtree, middleIndex+1, rightIndex);
            }


        }


        void printLeafNodes(){

            for (int id = 0; id < numNodes; id++){

                if (arrayOfBTNodes[id].getLeftChildPtr() == 0 && arrayOfBTNodes[id].getRightChildPtr() == 0)
                    cout << arrayOfBTNodes[id].getData() << " ";
            }

            cout << endl;
        }



};


int main(){

    int numElements;
    cout << "Enter the number of elements: ";
    cin >> numElements;

    BinarySearchTree bsTree(numElements);

    int array[numElements];

    for (int index = 0; index < numElements; index++){
        cout << "Enter element at index " << index << ": ";
        cin >> array[index];
    }

    bsTree.constructBSTree(array);

    cout << "Leaf nodes: ";
    bsTree.printLeafNodes();
    cout << endl;

    return 0;
}
```



```
Enter the number of elements: 7
Enter element at index 0: 12
Enter element at index 1: 15
Enter element at index 2: 18
Enter element at index 3: 23
Enter element at index 4: 25
Enter element at index 5: 29
Enter element at index 6: 64
Leaf nodes: 12 18 25 64
```