```
// Simple binary tree

#include <cassert>
#include <cstdlib>

class Node {
private:
    int data;

    Node *left;
    Node *right;
    Node *parent;

public:
    // Constructor
    Node(int data)
    {
        this->data = data;

        left   = 0;
        right  = 0;
        parent = 0;
    }

    int getData()     { return data;   }

    Node *getLeft()   { return left;   }
    Node *getRight()  { return right;  }
    Node *getParent() { return parent; }

    void setLeft(Node *left)
    {
        this->left = left;
    }

    void setRight(Node *right)
    {
        this->right = right;
    }

    void setParent(Node *parent)
    {
        this->parent = parent;
    }
};

class Tree {
private:
    Node *root;


    Node *search(int data)
    {
        Node *n = root;
        while (n != 0)
        {
            if (data < n->getData())
                n = n->getLeft();
            else if (data > n->getData())
                n = n->getRight();
            else
                return n;  // n == n->getData()
        }

        return 0; // not found
    }

public:
    // Constructor
    Tree()
    {
        root = 0;
    }
```

```
    // Search data
    bool contains(int data)
    {
        Node *n = search(data);
        return (n != 0);
    }

    // Insert element
    void add(int data)
    {
        Node *parent = 0;
        Node *n = root;
        while (n != 0)
        {
            parent = n;
            if (data < n->getData())
                n = n->getLeft();
            else if (data > n->getData())
                n = n->getRight();
            else
                break;  // data == n->getData()
        }

        Node *newNode = new Node(data);
        newNode->setParent(parent);

        if (parent == 0)
        {
            root = newNode;
        }
        else if (data < parent->getData())
        {
            parent->setLeft(newNode);
        }
        else if (data > parent->getData())
        {
            parent->setRight(newNode);
        }
        else
        {
            // Already in tree
            delete newNode;
        }
    }
};


// Tester
const int size = 2000;

int main(int argc, char *argv[])
{
    Tree *t = new Tree();

    int x[size];
    for (int i = 0; i < size; i++)
    {
        x[i] = random() * 2;
    }

    for (int i = 0; i < size; i++)
    {
        t->add(x[i]);
    }

    for (int i = 0; i < size; i++)
    {
        assert(t->contains(x[i]));
    }

    for (int i = 0; i < size; i++)
    {
        int r = random() * 2 + 1;
        assert(!t->contains(r));
```

```
    }
}
```