# Homework: Basic Tree Data Structures

This document defines the **homework assignments** for the ["Data Structures" course @ Software University](https://softuni.bg/trainings/1147/Data-Structures-June-2015).

## Introduction

You are given a **tree of N nodes** represented as a set of N-1 pairs of nodes (parent node, child node). Below are the operations that you are going implement.

|  |  |  |  |
| --- | --- | --- | --- |
| **Input** | **Comments** | **Tree** | **Definitions** |
| 9  7 19  7 21  7 14  19 1  19 12  19 31  14 23  14 6  27  43 | N = 9  Nodes: 7🡪19, 7🡪21, 7🡪14, 19🡪1, 19🡪12, 19🡪31, 14🡪23, 14🡪6  P = 27  S = 43 |  | Root node: 7  Leaf nodes: 1, 6, 12, 21, 23, 31  Middle nodes: 14, 19  Leftmost deepest node: 1  Longest path: 7 -> 19 -> 1 (length = 3)  Paths of sum 27: 7 -> 19 -> 1 7 -> 14 -> 6  Subtrees of sum 43: 14 + 23 + 6 |

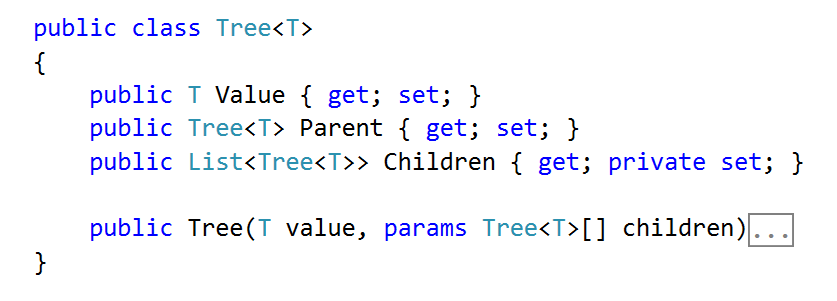
## Root Node

Write a program to read the tree and find its **root** node:

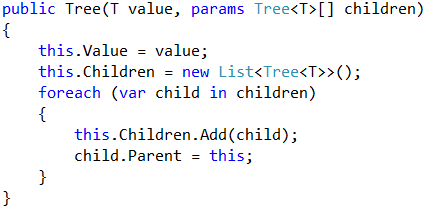
|  |  |  |
| --- | --- | --- |
| **Input** | **Output** | **Tree** |
| 9  7 19  7 21  7 14  19 1  19 12  19 31  14 23  14 6 | Root node: 7 |  |

Hints

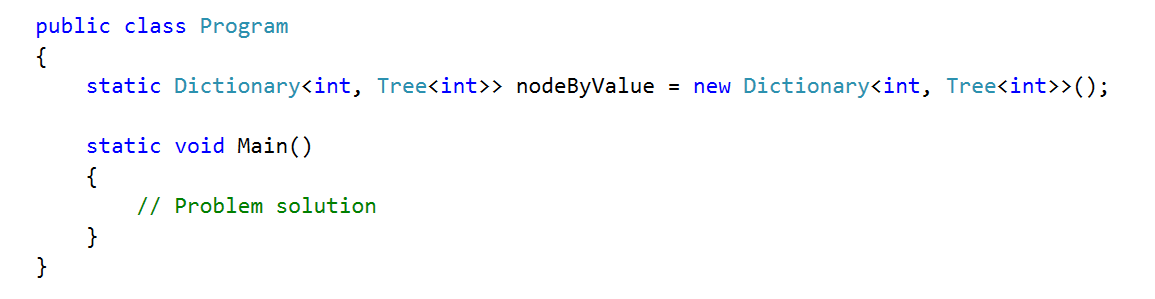
Use the recursive Tree<T> definition. Keep the **value**, **parent** and **children** for each tree node:



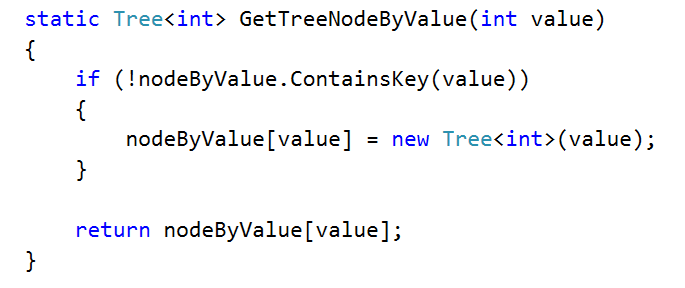
Modify the Tree<T> **constructor** to **assign a parent** for each child node:



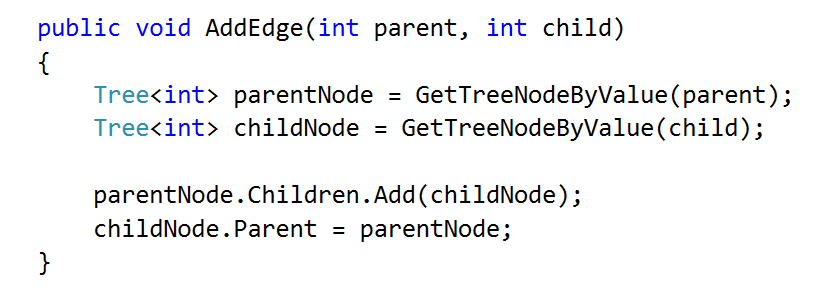
Use a **dictionary** to map nodes by their value. This will allow you to find the tree nodes during the tree construction (when you read the input data, you get the node values):



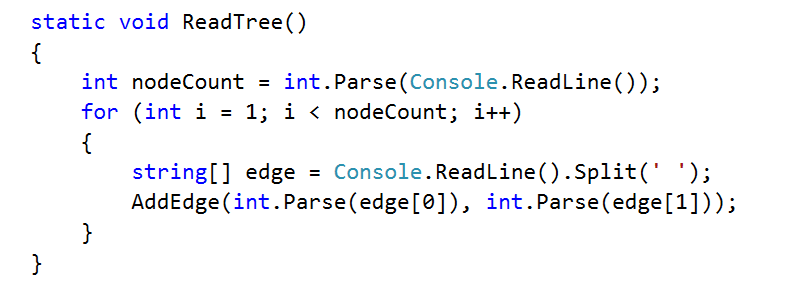
Write a method to **find the tree node by its value or create a new node** if it does not exist:



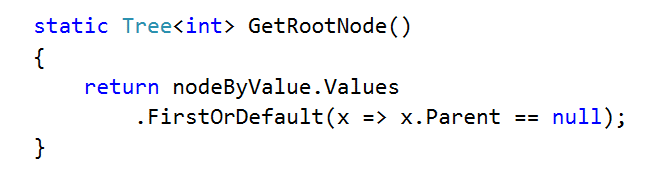
Create a method for adding an edge to the tree



Now you are ready to **create the tree**. You are given the **tree edges** (parent + child). Use the dictionary to lookup the parent and child nodes by their values:



Finally, you can find the root (the node that has no parent)



## Print Tree

Write a program to read the tree from the console and print it in the following format (each level indented +2 spaces):

|  |  |  |
| --- | --- | --- |
| **Input** | **Output** | **Tree** |
| 9  7 19  7 21  7 14  19 1  19 12  19 31  14 23  14 6 | 7  19  1  12  31  21  14  23  6 |  |

Hints

Find the root and recursively print the tree

## Leaf Nodes

Write a program to read the tree and find all **leaf** nodes (in increasing order):

|  |  |  |
| --- | --- | --- |
| **Input** | **Output** | **Tree** |
| 9  7 19  7 21  7 14  19 1  19 12  19 31  14 23  14 6 | Leaf nodes: 1 6 12 21 23 31 |  |

Hints

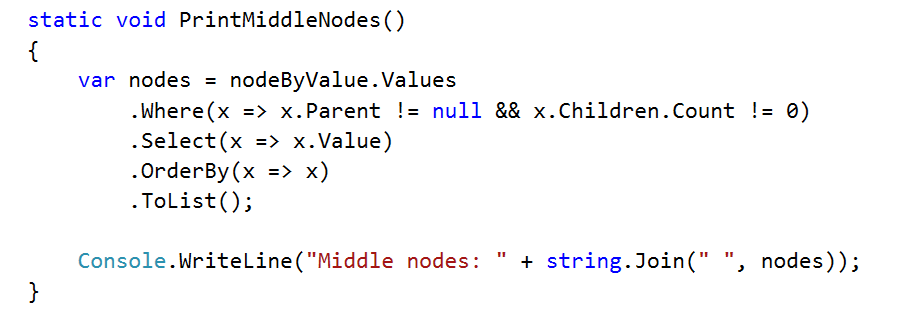
Find the all nodes that have no children

## Middle Nodes

Write a program to read the tree and find all **middle** nodes (in increasing order):

|  |  |  |
| --- | --- | --- |
| **Input** | **Output** | **Tree** |
| 9  7 19  7 21  7 14  19 1  19 12  19 31  14 23  14 6 | Middle nodes: 14 19 |  |

Hints



## \* Deepest Node

Write a program to read the tree and find its deepest node (leftmost):

|  |  |  |
| --- | --- | --- |
| **Input** | **Output** | **Tree** |
| 9  7 19  7 21  7 14  19 1  19 12  19 31  14 23  14 6 | Deepest node: 1 |  |

## Longest Path

Find the **longest path** in the tree (the leftmost if several paths have the same longest length)

|  |  |  |
| --- | --- | --- |
| **Input** | **Output** | **Tree** |
| 9  7 19  7 21  7 14  19 1  19 12  19 31  14 23  14 6 | Longest path: 7 19 1 |  |

## All Paths With a Given Sum

Find all paths in the tree with **given sum** of their nodes (from the leftmost to the rightmost)

|  |  |  |
| --- | --- | --- |
| **Input** | **Output** | **Tree** |
| 9  7 19  7 21  7 14  19 1  19 12  19 31  14 23  14 6  27 | Paths of sum 27:  7 19 1  7 14 6 |  |

## \* All Subtrees With a Given Sum

Find all **subtrees with given sum** of their nodes (from the leftmost to the rightmost). Print subtrees in **pre-order** sequence

|  |  |  |
| --- | --- | --- |
| **Input** | **Output** | **Tree** |
| 9  7 19  7 21  7 14  19 1  19 12  19 31  14 23  14 6  43 | Subtrees of sum 43:  14 23 6 |  |