**Exercises: Trees Representation and Traversal (BFS and DFS)**

This document defines the lab for ["Data Structures – Fundamentals (C#)" course @ Software University](https://softuni.bg/trainings/3419/data-structures-fundamentals-with%20csharp-june-2021).

Please submit your solutions (source code) of all below described problems in [Judge](https://judge.softuni.bg/Contests/2468/04-Trees-Representation-and-Traversal-BFS-and-DFS-Exercise).

Write C# code for solving the tasks on the following pages.

## Tree Problems Overview

You are given a **tree of N + 1 nodes** represented as a set of **N** pairs of nodes (parent node, child node). For each problem you have specified definition of the implementation required to complete the task. You can use different approaches which you find suitable for the task you are solving.

In general, all the problems require basic tree knowledge and understanding of **DFS** and **BFS traversal algorithms**.

Because of the way you should read the tree from the console, it **cannot contain duplicate elements** in any circumstances. The tests do not include such cases as well.

For some of the exercises where the tree ordering does not matter, the tests’ input may be shuffled.

Example:

|  |  |  |  |
| --- | --- | --- | --- |
| **Input** | **Comments** | **Tree** | **Definitions** |
| 8  7 19  7 21  7 14  19 1  19 12  19 31  14 23  14 6  27  43 | N = 8  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  Leftmost longest path: 7 -> 19 -> 1 (length = 3)  Paths of sum 27: 7 -> 19 -> 1 7 -> 14 -> 6  Subtrees of sum 43: 14 + 23 + 6 |

## Create Tree

Write a program to read a tree from the console and find its **root** node.

|  |  |  |
| --- | --- | --- |
| **Input** | **Output** | **Tree** |
| 8  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. Use appropriate data structure to find and map nodes by their value. Write a logic to **find the tree node by its value or create a new node** if it does not exist. Create a mechanism to add an edge to the tree. Now you are ready to **create the tree**. You are given the **tree edges** (parent + child). Use the map to lookup the parent and child nodes by their values. Finally, you can find the root (the node that has no parent).

## Tree As String

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

|  |  |  |
| --- | --- | --- |
| **Input** | **Output** | **Tree** |
| 8  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 a tree from the console and find all **leaf** nodes (in ascending order):

|  |  |  |
| --- | --- | --- |
| **Input** | **Output** | **Tree** |
| 8  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 a tree from the console and find all **middle** nodes (in ascending order):

|  |  |  |
| --- | --- | --- |
| **Input** | **Output** | **Tree** |
| 8  7 19  7 21  7 14  19 1  19 12  19 31  14 23  14 6 | Middle nodes: 14 19 |  |

Hints

We can say that **middle** **nodes** are the once that have a **parent** **and at least one child at the same time.**

## \* Deepest Node

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

|  |  |  |
| --- | --- | --- |
| **Input** | **Output** | **Tree** |
| 8  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** |
| 8  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** |
| 8  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).

|  |  |  |
| --- | --- | --- |
| **Input** | **Output** | **Tree** |
| 8  7 19  7 21  7 14  19 1  19 12  19 31  14 23  14 6  43 | Subtrees of sum 43:  14 23 6 |  |