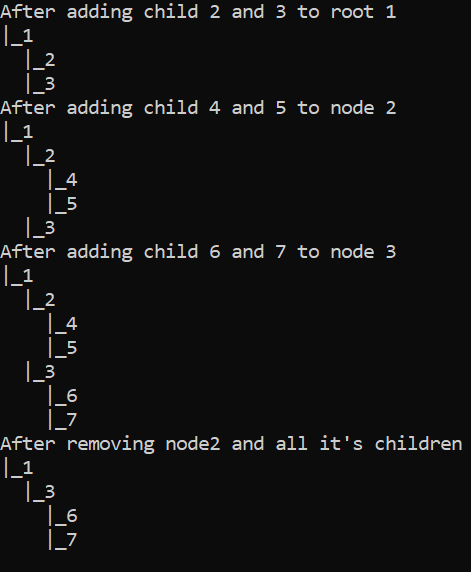
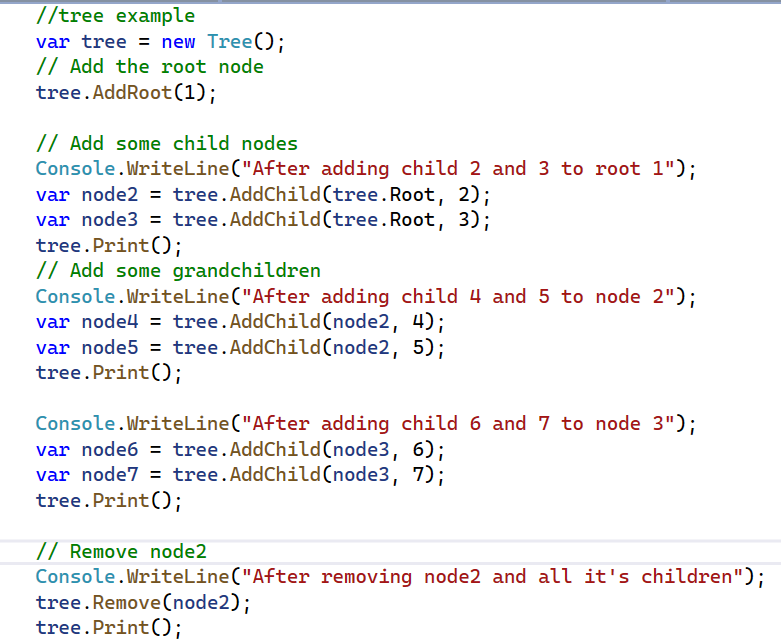
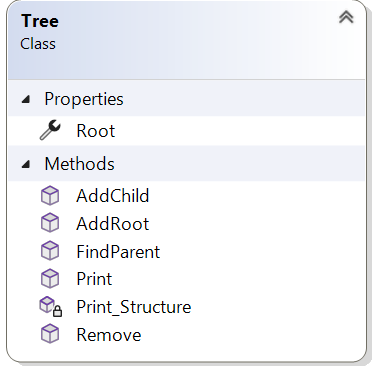
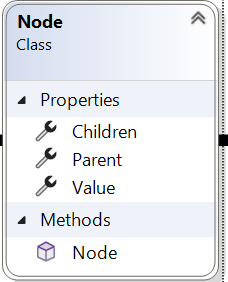
Labb Create your own Tree data structure in C#

After running this code in Programs.cs main method, the output should be as below



Create these two classes:



1. Define a **Node class**: A tree is made up of nodes, so start by defining a Node class. Here, **Value** is an integer that represents the value of the node, **Parent** is a reference to the parent node, and **Children** is a list of references to the child nodes.
   1. In the constructor of the Node class, initialize the Value property and create an empty Children list:
2. Define the **Tree class**: A tree is essentially a collection of nodes, so create a Tree class that contains the root node. Here, **Root** is a reference to the root node of the tree. **AddRoot** adds a new root node with the given value if one does not already exist. **AddChild** adds a new child node to the given parent node with the given value. **Remove** removes the given node and all its children from the tree. **FindParent** returns the parent node of the given node, and **Print** prints the tree structure to the console.It uses the private method **Print\_Structure**

**Explanations for each method in Tree class**

1. public Node AddRoot(int value)
   1. The method takes an integer value as its input parameter.
   2. The first thing the method does is check whether the root of the tree already exists. It does this by checking whether the "Root" variable (presumably an instance variable of the class containing this method) is null or not. If it's not null, then the root already exists, and the method throws an InvalidOperationException.
   3. If the root does not already exist, the method creates a new node with the input value and assigns it to the "Root" variable.
   4. Finally, the method returns the new root node.
   5. Overall, this method ensures that the tree has only one root and throws an exception if an attempt is made to add a new root when one already exists.
2. public Node AddChild(Node parent, int value)
   1. The method takes two parameters - the parent node to which the new child node will be added, and the value that the new child node will hold.
   2. The first thing the method does is check whether the parent node is null. If it is null, then the method throws an ArgumentNullException with a message indicating that the parent node cannot be null.
   3. If the parent node is not null, the method creates a new node with the input value and assigns it to a "child" variable.
   4. The method then adds the new child node to the "Children" collection of the parent node.
3. public void Remove(Node node)
   1. The method takes a single parameter - the node to be removed from the tree.
   2. The first thing the method does is check whether the input node is null. If it is null, then the method throws an ArgumentNullException with a message indicating that the node cannot be null.
   3. If the input node is not null, the method checks whether the input node is the root node. If it is, the method sets the root node to null, effectively removing the entire tree.
   4. If the input node is not the root node, the method removes the input node from its parent's "Children" collection by calling the "Remove" method on the parent's "Children" collection, passing in the input node as the argument.
   5. The method then sets the input node's parent to null, effectively detaching the input node from the tree.
   6. Overall, this method removes a node from the tree and ensures that the tree structure is updated properly, including removing the root node if necessary.
      1. Finally, the method sets the parent of the new child node to be the input parent node and returns the new child node.
   7. Overall, this method ensures that the new child node is properly linked to its parent node and throws an exception if an attempt is made to add a child node to a null parent.
4. public void Print()
   1. The method first checks whether the tree is empty, i.e., whether the root node is null. If the root node is null, the method prints a message to the console indicating that the tree is empty.
   2. If the root node is not null, the method calls a recursive helper function named "Print\_Structure", passing in the root node and an empty string as arguments. This function will recursively traverse the tree and print out its structure.
   3. The "Print\_Structure" function takes two parameters - the current node being processed and a string that contains the prefix to be printed before each node. The prefix is used to visually indicate the level of each node in the tree.
   4. The function first prints the prefix followed by the value of the current node to the console.
   5. The function then recursively calls itself on each child of the current node, passing in the child node and a modified prefix that includes a "|" character and additional spacing to indicate the level of the child node in the tree.
   6. The recursive calls continue until all nodes in the tree have been printed.
   7. Overall, this method prints out the structure of the tree in a visually informative way, showing the relationships between nodes and indicating the level of each node in the tree.
5. private void Print\_Structure(Node node, string prefix)
   1. This is a recursive helper method that is called by the "Print" method to print out the structure of a tree data structure. Here's a breakdown of what the code is doing:
   2. The method takes two parameters - the current node being processed and a string that contains the prefix to be printed before each node.
   3. The method first prints the prefix followed by a "|" character and the value of the current node to the console. This is the visual representation of the current node in the tree structure.
   4. The method then iterates over each child node of the current node using a for loop.
   5. For each child node, the method recursively calls itself, passing in the child node as the new "node" parameter and a modified prefix that includes an additional two spaces to visually indicate that the child node is one level deeper in the tree than the current node.
   6. The recursive calls continue until all nodes in the tree have been printed.
   7. Overall, this method is a helper function used to print out the structure of a tree data structure in a visually informative way. It recursively traverses the tree and prints out the value of each node, along with a visual indication of its level in the tree.

Beware of plagiarism:

In short, plagiarism is passing off someone else's work as your own. This can in many cases

happen unconsciously and that is why you are asked to read the university's information about it.

**By your active act** of submitting this examination**, you confirm/certify** that **You have read, understood and followed the instruction**s at this link: [References and copyright](https://www.du.se/sv/bibliotek/referera/) → Plagiarism (References and copyright), including all content in the links "Refero - The Anti-Plagiarism Guide".

ChatGPT:

When **using chatgpt, list the questions you asked it**. And [reference](https://libguides.kpu.ca/c.php?g=713337&p=5281556) in this manner:

-OpenAI. (2023, March 5). [ChatGPT's response to the question "What is important for effective intercultural communication in an organization?"]. Accessed 2023, 5 March  [https://chat.openai.com](https://chat.openai.com/chat).

Hand in:

* **Lab report containing:** What have you learned from the lab? What problems did you encounter and how did you solve them? What do you want to learn more about in this area of Hash tables? List questions you asked chatgpt and [reference](https://libguides.kpu.ca/c.php?g=713337&p=5281556) in this manner:
  + -OpenAI. (2023, March 5). [ChatGPT's response to the question "What is important for effective intercultural communication in an organization?"]. Accessed 2023, 5 March [https://chat.openai.com](https://chat.openai.com/chat).
* **Video recording** for the lab where you run and demonstrate the functionality of the code, as well as explaining key concepts and their implementations in the code. Run and explain your unit tests. Your face/faces should be in the recording.
  + If you for some reason can’t do recording together in the group. Then each member do their own recording and hand it in.
* **Zipped visual studio project,** that includes commented source code.