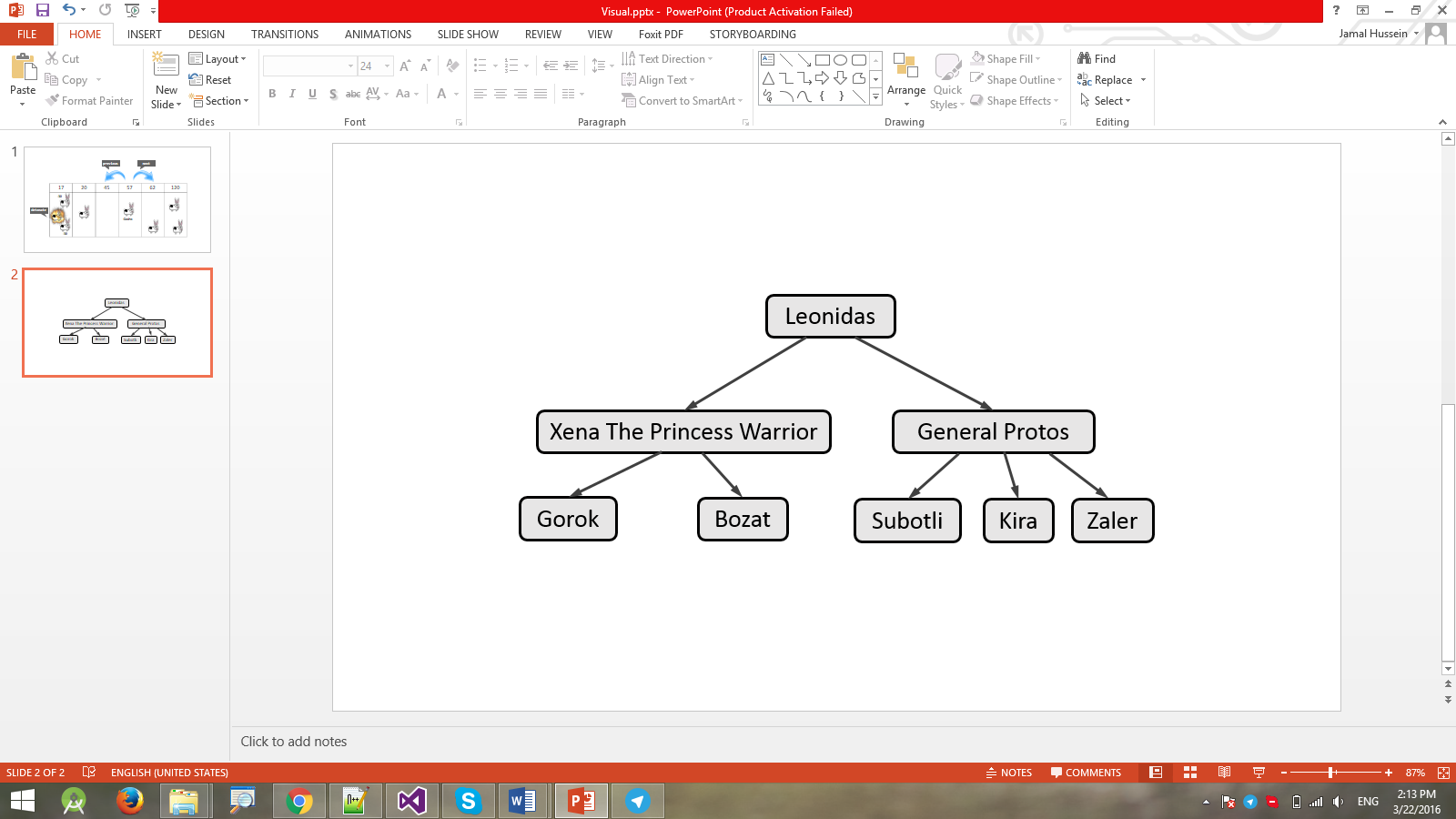
**Exercises: B-2-3 and AVL Trees**

This document defines the lab for ["Data Structures – Advanced (C#)" course @ Software University](https://softuni.bg/trainings/3420/data-structures-advanced-with-csharp-august-2021).

Please submit your solutions (**source code**) of all below described problems in [Judge](https://judge.softuni.bg/Contests/2574/01-B-Trees-2-3-Trees-and-AVL-Trees-Lab).

## Hierarchy

A **Hierarchy** is a data structure that stores elements in a hierarchical order. See the example:



It supports the following operations:

* **Add(element, child)** - adds **child** to the hierarchy as a child of **element**.
  + Throws an exception if **element** does not exist in the hierarchy.
  + Throws an exception if **child** already exists (duplicates are not allowed).
* **Remove(element)** - removes the element from the hierarchy.
  + If it has children, they become children of the element's parent.
  + If element is root node, throws an exception.
* **Count** - returns the count of all elements in the hierarchy
* **Contains(element)** - determines whether the element is present in the hierarchy.
* **GetParent(element)** - returns the parent of the element.
  + Throws an exception if **element** does not exist in the hierarchy.
  + Returns the **default value for the type** (e.g. **int** → **0**, **string** → **null**, etc.) if element has no parent.
* **GetChildren(element)** - returns a collection of all direct children of the element in order of their addition.
  + Throws an exception if **element** does not exist in the hierarchy.
* **GetCommonElements(Hierarchy other)** - returns a collection of all elements that are present in both hierarchies (order does not matter).
* **ForEach()** - enumerates over all elements in the hierarchy by levels.
  + In the image above, the elements would be enumerated as such - **Leonidas** -> **Xena the Princess Warrior** -> **General Protos** -> **Gorok** -> **Bozat** -> **Subotli** -> **Kira** -> **Zaler**.

### Input and Output

You are given a **Visual Studio C# project skeleton** (unfinished project) holding the interface **IHierarchy**, the unfinished class **Hierarchy** and **tests** covering its **functionality** and its **performance**.

Your task is to **finish this class** to make the tests run correctly.

* You are **not allowed to change the tests**.
* You are **not allowed to change the interface**.

### Interface IHierarchy

The interface **IHierarchy** in C# looks like the code below:

|  |
| --- |
| public interface IHierarchy<T> : IEnumerable<T>  {  int Count { get; }  void Add(T element, T child);  void Remove(T element);  IEnumerable<T> GetChildren(T element);  T GetParent(T element);  bool Contains(T element);  IEnumerable<T> GetCommonElements(IHierarchy<T> other);  } |

1. **Implement 2-3 Tree Insertion**

You are given a **C# project** holding the unfinished class **TwoThreeTree** and **tests** covering its **functionality**.

Your task is to implement the insertion method. You are free to create additional methods of your own choice, however only the insertion will be tested. Do **not change the ToString()** method it is used in testing.

Hint: If you have any problem with the implementation you can check the presentation from the lab, here were some graphics describing the steps while adding items.

## \*Implement AVL Tree Deletion

Extend your AVL Tree to support:

* **void DeleteMin()** deletes the minimum element (balances the tree if necessary)
* **void Delete(T** **item)** deletes the given element (balances the tree if necessary)

You are given a skeleton with additional tests that cover delete operations.