

Homework: Dictionaries, Hash Tables and Sets

This document defines the **homework assignments** for the ["Data Structures" course @ Software University](#). Please submit a single **zip / rar / 7z** archive holding the solutions (source code) of all below described problems.

Problem 1. Dictionary

Implement a **dictionary** using a **hash table**. Refer to the [lab document](#) for detailed steps on how to implement the data structure.

Solve the next problems by using the **implemented dictionary**. You are **NOT** allowed to use the built-in **Dictionary** or **SortedDictionary** classes!

Problem 2. Count Symbols

Write a program that reads some text from the console and counts the occurrences of each character in it. Print the results in **alphabetical** (lexicographical) order. Examples:

Input	Output	Input	Output
SoftUni rocks	: 1 time/s S: 1 time/s U: 1 time/s c: 1 time/s f: 1 time/s i: 1 time/s k: 1 time/s n: 1 time/s o: 2 time/s r: 1 time/s s: 1 time/s t: 1 time/s	Did you know Math.Round rounds to the nearest even integer?	: 9 time/s .: 1 time/s ?: 1 time/s D: 1 time/s M: 1 time/s R: 1 time/s a: 2 time/s d: 3 time/s e: 7 time/s g: 1 time/s h: 2 time/s i: 2 time/s k: 1 time/s n: 6 time/s o: 5 time/s r: 3 time/s s: 2 time/s t: 5 time/s u: 3 time/s v: 1 time/s w: 1 time/s y: 1 time/s

a. Phonebook

Write a program that receives some info from the console about **people** and their **phone numbers**.

You are free to choose the manner in which the data is entered; each **entry** should have just **one name** and **one number** (both of them strings).

After filling this simple phonebook, upon receiving the **command "search"**, your program should be able to perform a search of a contact by name and print her details in format **"{name} -> {number}"**. In case the contact isn't found, print **"Contact {name} does not exist."** Examples:

Input	Output
Nakov-0888080808 search Mariika Nakov	Contact Mariika does not exist. Nakov -> 0888080808
Nakov-+359888001122 RoYaL(Ivan)-666 Gero-5559393 Simo-02/987665544 search Simo simo RoYaL RoYaL(Ivan)	Simo -> 02/987665544 Contact simo does not exist. Contact RoYaL does not exist. RoYaL(Ivan) -> 666

Problem 3. Ordered Set

Implement the **ordered set** data structure. It should store **unique elements** in a **binary search tree**. The elements should be kept **sorted at all times**. The **ordered set** should be **generic** and support the following operations:

- **Add(T element)** - adds the element to the set
- **Contains(T element)** - determines whether the element is present in the set
- **Remove(T element)** - removes the element from the set. Its place should be taken by the **bigger child node**.
- **Count** - property that returns the number of unique elements in the set
- The set should be **foreach**-able (just like arrays, lists and other data structures). Implement the **IEnumerable<T>** interface to achieve this. The set should yield all elements, **sorted**, in ascending order.

Tip: Use [in-order traversal](#).

Sample Code	Internal Binary-Tree
<pre>var set = new OrderedSet<int>(); set.Add(17); set.Add(9); set.Add(12); set.Add(19); set.Add(6); set.Add(25); foreach (var item in set) { Console.WriteLine(item); }</pre>	<pre> graph TD 17((17)) --> 9((9)) 17 --> 19((19)) 9 --> 6((6)) 9 --> 12((12)) 19 --> 25((25)) </pre>

Problem 4. ** Balanced Ordered Set

Extend the ordered set from the previous problem by making the internal binary tree **self-balancing**. Balance the tree each time an item is inserted or deleted. Implement it using a binary search tree such as an [AVL tree](#), [AA tree](#) or [Red-Black Tree](#).