Assignment I: University Organisation Tree
Deadline: March 29, 2024, 23:59

Please Read This Page Carefully

Submission Rules:

- **You should provide your name, surname, and ID, as well as a comment on the first line of your files**, stating that you read the rules specified here and the submission is your own work. **Submissions without this statement will not be graded.** An example statement can be provided as follows:

```
-- Zekican Budin – 1234567
-- I read and accept the submission rules and the extra rules specified
in each question. This is my own work that is done by myself only.
```

- Please refer to the syllabus[1] provided for CNG 242 for the measures in place in case of any academic dishonesty[2,3].

- The instructors or TAs may ask for demo sessions for any of the submissions.

- You cannot share this worksheet with any third parties. Upon doing so, any detected action will directly be sent to the disciplinary committee.

- You need to submit a single .hs file **named with your student id** only. For example, **1234567.hs . Any other files or attachments will not be graded.**

- You should read the questions fully and follow the directions listed in there**.**

- You can only get full marks if your file fully compiles, runs, follows rules and generates correct output for all possible cases and inputs.

- You should only submit one solution per question. Your solution might have multiple lines or include self-written helper functions. However, **only the functions with the same name** will be graded. If you submit multiple solutions, your lowest scoring solution will be considered.

- The assignment should not be shared publicly in any manner, at any time. The assignment cannot be disclosed or disseminated to anyone before, during, or after the submission.

- **You cannot import libraries**; you cannot use anything that we did not cover in the first 5 lab sessions and in classes. Additionally, you are not allowed to use following keywords; print, putStrLn, putStr, putChar. Everything you need is either in slides or in lab worksheets.

---

[1] Page 3&4 (Course rules, #1,2,3)
[2] Taking unfair advantage in assessment is considered a serious offence by the university, which will take action against any student who contravenes the regulation through negligence or deliberate intent.
[3] For a comprehensive cheating definition, please refer to: https://ncc.metu.edu.tr/res/academic-code-of-ethics . When a breach of the code of ethics occurs (cheating, plagiarism, deception, etc.), the student will be added to the BLACKLIST.

Learning Outcomes: On successful completion of this assignment, a student will:

- Have practiced how to program using functional programming languages
- Have used different approaches to define data types in Haskell
- Have practised how to use recursive definitions in functional programming paradigms

In this assignment, we are going to work with tree structures. The case study we are going to consider is an "Organisation Tree" for a university. Organisation trees are often used in various ways, such as hierarchical databases, to store and analyse information related to organisations in a hierarchical way.

For simplicity, in this assignment, we are going to consider a **university organisation tree** where we have the **rectorate** as the root, there are multiple faculties, and there are departments that belong to these faculties. Vertices of the tree are used to represent organisational components, and edges are used to represent the relationships of these components in different hierarchical levels.

The terminology we will use for the analysis will be the one adopted from disciplines such as botany and genealogy [1]. Considering a rooted tree *T*, if *v* is a vertex in *T* other than the root, the unique vertex *u* with a directed edge from u to v is called *the* **parent** of *v*. Vertices with the same parent are called **siblings**. If a vertex of a rooted tree has no children, it is called a **leaf** and all the other vertices are called **internal vertices**.

University Organisation Tree [50 points]

In this assignment,

- a list of organisational component names,
- a list of integer values for the number of sub-divisions
- a list of the number of employees in each section

will be provided as an input. In turn, your program will establish an organisational tree. Please consider the tree of an imaginary university presented in Figure 1 as an example. This example is used for the following sample run. Additionally, you need to design a disjoint union that fits to this example. Please note that, the tree children count can be infinite, therefore, **an organisational component can have more than three children (eg. 10, 20, 30,... children)**

Input: a list of tuples where each tuple contains organisational components and number of employees in that organisation, presented in (organisation name, employee count) format, a list of integer values for the number of sub-divisions.

Output: Your own university organisation tree data type

Sample run:
unitree [("Rectorate", 2), ("General Secretary", 6), ("Social Sciences", 2), ("Economics", 4), ("Political Science", 5), ("Education", 2), ("Pre School", 3), ("Foreign Languages", 4), ("Engineering", 4), ("Computer Engineering", 10), ("Mechanical Engineering", 5), ("Electrical Engineering", 6)] [3, 0, 3, 0, 0, 2, 0, 0, 3, 0, 0, 0]

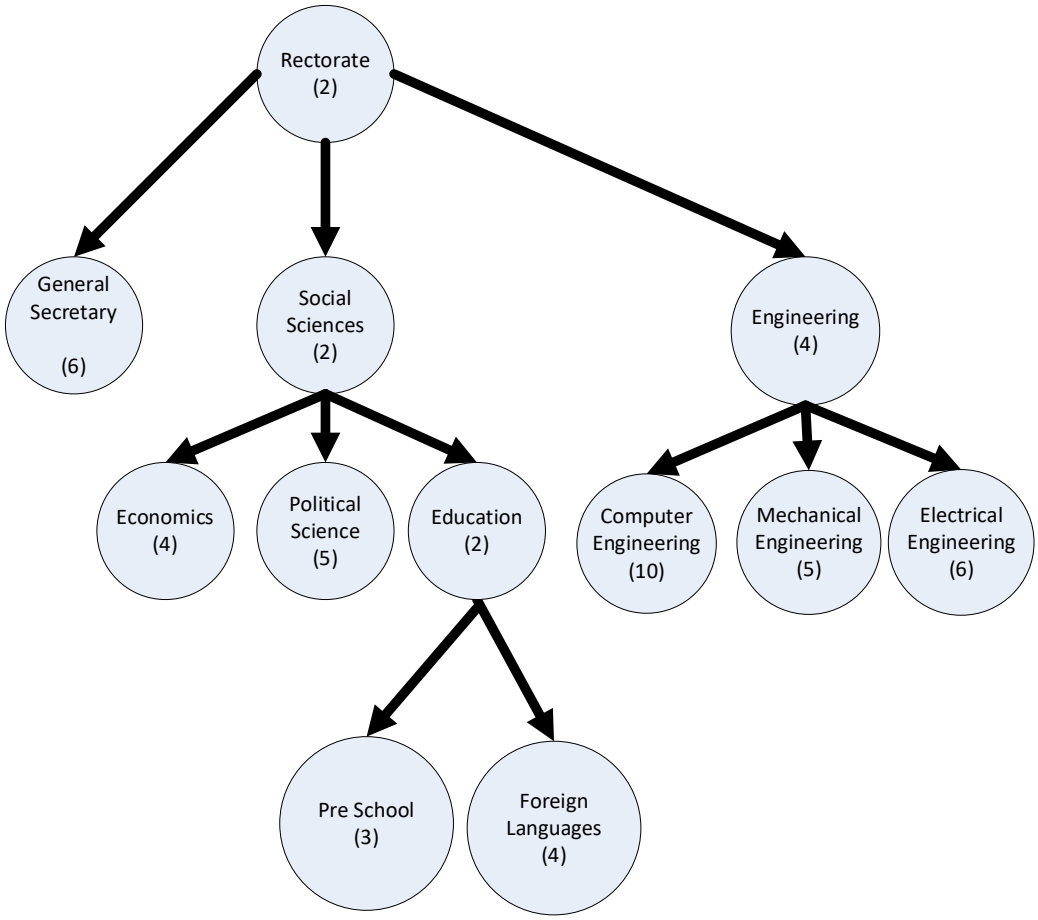

Figure 1: Example university organisation tree.

Tree Analysis [50 points]

In this part of the assignment, you will also introduce some functions in order to analyse the university organisation tree as follows:

sectionsize: This function will return the total number of employees in a section and all its sub-divisions if there are any.

Input:	tree, name of the section
Output:	Total number of employees
Sample run-1:	sectionsize unitree "Engineering" 25
Sample run-2:	sectionsize unitree "Education" 9

managingentity: This function will return the name of the immediate section responsible from the management of the given subdivision.

Input:	tree, name of the section
Output:	name of the managing section
Sample run:	managingentity unitree "Education" "Social Sciences"

managelist: This function will return the names of the all managing entities for the given section until the root.

Input:	tree, name of the section
Output:	list of all the sections on the path to the root
Sample run:	managelist unitree "Preschool" ["Education", "Social Sciences", "Rectorate"]

Assessment Criteria

The assignment will be marked as follows:

Grading Item	Marks (Total 100)
Implementation of Unitree and a relevant disjoint union (data)	50
sectionsize	15
managingentity	15
managelist	20

References

1. Rosen, Kenneth. Discrete Mathematics and Its Applications 7th edition. McGraw-Hill, 2011.
2. Wenger, Rephael. Isosurfaces: Geometry, Topology, & Algorithms. CRC Press, 2013.