Lehrstuhl für Angewandte Informatik II
Universität Bayreuth, Universitätsstraße 30, 95447 Bayreuth
Prof. Dr. Thomas Rauber Büro: AI 2.08
Dr. Tim Werner Büro: AI 2.15
Internet: https://elearning.uni-bayreuth.de/course/view.php?id=45978
Email: rauber@uni-bayreuth.de, werner@uni-bayreuth.de

Parallel and Distributed Systems I

Exercise 1

Release date: Wednesday, 15. 10. 2025 Due date: Friday, 07. 11. 2025, 12:00

Hinweis: *Your solutions of the programming exercises including all helper files are to be uploaded to the elearning site. The Compiler* `gcc` *should be able to translate your solutions.*

Exercise 1.1 (1+3+3+1 Points)

In this exercise you have to develop a simple simulation of a Galton Board[1] with a subsequent console output using the C language:

(a) First, the simulation should ask the user to enter the amount of beads to be traced and the amount of bins via the console.

(b) Next, the simulation should trace the beads one after another through the board. In order to decide, whether a bead takes the left or the right path whenever it hits a peg, the simulation should use the *rand* function. The bins of the Galton Board are to be implemented as a one dimensional array, which also can be considered as a histogram. If a bead reaches a bin, the simulation increments the entry of the respective bin in the array by one.

(c) Now, the simulation should create a simple ASCII image from the histogram and save this image in a (two dimensional) array. For example, this image may look as follows:

```
          X
          X  X
       X  X  X  X
    X  X  X  X  X  X
 X  X  X  X  X  X  X
```

Hint: By not entering a symbol for each bead in the ASCII image, but by combining a certain fixed number of beads to a symbol, the binomial distribution in this ASCII image will become much more recognizable!

(d) Finally, the simulation should output this ASCII image on the console.

[1] https://en.wikipedia.org/wiki/Bean_machine

Exercise 1.2 (10+2 Points)

In this exercise you have to develop a binary search tree [1], which uses integers as keys, using the C language. There are already the following structures defined for this search tree:

```
typedef struct NODE Node;

struct NODE{
  int key;
  Node* smaller_keys;
  Node* larger_keys;
};

typedef struct{
    Node* root;
}Tree;
```

(a) Implement the following functions for the search tree:

- **tree_create:** Creates an empty search tree.
- **tree_insert_key:** Inserts a new key into the search tree.
- **tree_find_key_recursive:** Checks recursively whether the binary search tree contains a given key.
- **tree_find_key_iterative:** Checks iteratively whether the binary search tree contains a given key.
- **tree_is_valid:** Checks whether the search tree is valid, i.e. that for every node the partial tree containing the smaller keys actually only contains smaller keys, and that the partial tree containing the larger tree actually only contains larger keys.
- **tree_deep_copy:** Creates a deep copy of a search tree.
- **tree_delete:** Deletes a search tree including all of its nodes.

(b) Implement a main function, which creates and fills a search tree with random keys, and tests all functions implemented beforehand.

[1] `https://en.wikipedia.org/wiki/Binary_search_tree`