This assignment is **due on April 9**. All submitted work must be *done individually* without consulting someone else's solutions in accordance with the University's "Academic Dishonesty and Plagiarism" policies.

Problem 1. We are planning a board games event and we're using one of the shelves in my office to store the games. Unfortunately the shelf only has a certain amount of space S, so we need to carefully pick which games we want to bring. Every game takes some space s_i and has a fun factor f_i that indicates how much fun it is to play that game (for $1 \leq i \leq n$).

We want to maximize the amount of fun we'll have, so we want to maximize the sum of the fun factors of the games we pick (i.e., $\max \sum_{\text{picked game } i} f_i$), while making sure that the games fit on my shelf, so the sum of the space the games we pick take should be at most S (i.e., $\sum_{\text{picked game } i} s_i \leq S$). For simplicity, you can assume that all f_i, s_i, and S are (positive) integers.

We suggest two algorithms to compute the maximum amount of fun we can have, within the given space limit. The strategy of PickLargest is to always pick the game with the highest fun factor until my shelf is full: it sorts the games by their fun factor f_i in decreasing order and adds a game when its required space is less than the remaining space on the shelf.

The strategy of PickLargestRatio is to compare the games based on their fun/space ratio, to avoid picking a single very fun but also very big game too early: it sorts the games by the ratio of fun factor and space f_i/s_i in decreasing order and adds a game when its required space is less than the remaining space on the shelf.

```
1: function PickLargest(all f_i and s_i, S)
2:     currentSpace ← 0
3:     currentFun ← 0
4:     Sort games by f_i and renumber such that f_1 ≥ f_2 ≥ ... ≥ f_n
5:     for i ← 1; i ≤ n; i++ do
6:         if currentSpace + s_i ≤ S then
7:             currentSpace ← currentSpace + s_i          ▷ Pick the ith game
8:             currentFun ← currentFun + f_i
9:     return currentFun
```

```
1: function PickLargestRatio(all f_i and s_i, S)
2:     currentSpace ← 0
3:     currentFun ← 0
4:     Sort games by f_i/s_i and renumber such that f_1/s_1 ≥ f_2/s_2 ≥ ... ≥ f_n/s_n
5:     for i ← 1; i ≤ n; i++ do
6:         if currentSpace + s_i ≤ S then
7:             currentSpace ← currentSpace + s_i          ▷ Pick the ith game
8:             currentFun ← currentFun + f_i
9:     return currentFun
```

a) Show that PICKLARGEST doesn't always return the correct solution by giving a counterexample.

b) Argue whether PICKLARGESTRATIO always returns the correct solution by either arguing its correctness (if you think it's correct) or by providing a counterexample (if you think it's incorrect).

Problem 2. Suppose you have k non-empty *sorted* doubly-linked lists $L_1, \ldots, L_k$, where $n = |L_1| + \ldots + |L_k|$. We want to support both REMOVE-MIN and REMOVE-MAX in $O(\log k)$ time. Note that k is not a constant and may change over time, for example when all elements of a list are removed. We will only be removing elements from the lists, so you don't have to handle insertions. You are allowed $O(n)$ time to initialize the data structure. For full marks you need to meet *all* time requirements.

a) Describe your data structure, i.e., the structure and its operations.

b) Briefly argue the correctness of your operations.

c) Analyse the running time of your operations.

Problem 3. The median of a set of n distinct integers is the middle element when the integers are sorted. For simplicity, we will assume that n is odd, i.e., the median is at position $(n+1)/2$ in sorted order.

Example:
$\{-4, 1, 3, 5, 7\}$, median is 3.

In this problem we are interested in finding the median of a set of distinct integers stored in *two* AVL trees. You cannot assume anything about how the integers are distributed between the two trees (e.g. one might contain $n-1$ elements and the other only 1). In addition to the standard pointers to the parent and two children, and the integer itself, each node of the trees also stores the size of the subtree rooted at it.

Your task is to design an algorithm that takes these two trees as input and returns the median of the elements stored in them. For full marks your algorithm needs to run in $O(\log n)$ time.

a) Design an algorithm that solves the problem.

b) Briefly argue the correctness of your algorithm.

c) Analyse the running time of your algorithm.

Advice on how to do the home work

- Assignments should be typed and submitted as pdf (no handwriting)
- When designing an algorithm or data structure, it might help you (and us) if you briefly describe your general idea, and after that you might want to develop and elaborate on details. If we don't see/understand your general idea, we cannot give you points for it.
- Be careful with giving multiple or alternative answers. If you give multiple answers, then we will give you marks only for "your worst answer", as this indicates how well you understood the question.
- Some of the questions are very easy (with the help of the lecture notes or book). You can use the material presented in the lecture or book without proving it. You do not need to write more than necessary (see comment above).
- When giving answers to questions, always prove/explain/motivate your answers.
- When giving an algorithm as an answer, the algorithm does not have to be given as (pseudo-)code.
- If you do give (pseudo-)code, then you still have to explain your code and your ideas in plain English.
- Unless otherwise stated, we always ask about worst-case analysis, worst case running times etc.
- As done in the lecture, and as it is typical for an algorithms course, we are interested in the most efficient algorithms and data structures.
- If you use further resources (books, scientific papers, the internet,...) to formulate your answers, then add references to your sources and show clearly that you understand what you're referencing by explaining the approach as you would normally.
- If you refer to a result in a scientific paper or on the web you need to explain the results to show that you understand the results, and how it was proven.