Algorithms and Data Structures
Constructor University Bremen
Dr. Kinga Lipskoch

Course: CH-231-A
April 11th, 2023

Homework 9

- Submit one ZIP file per homework sheet which contains one PDF file (including pictures, computations, formulas, explanations, etc.) and your source code file(s) with one makefile and without adding executable, object or temporary files.
- The implementations of algorithms has to be done using C, C++, Python or Java.
- The TAs are grading solutions to the problems according to the following criteria: https://grader.eecs.jacobs-university.de/courses/ch_231_a/2023_1/Grading_Criteria_ADS.pdf

Problem 9.1 *Stacks & Queues* (9 points)

(a) (5 points) Implement using C++, Python or Java the data structure of a stack backed up by a linked list, that can store data of any type, and analyze the running time of each specific operation. Implement the stack such that you have the possibility of setting a fixed size but not necessarily have to (size should be -1 if unset). Your functions should print suggestive messages in cases of underflow or overflow. You can assume that if the size is passed, it will have a valid value.

```
class Stack[T]:
  private:
    struct StackNode {    // linked list
      T data;
      StackNode * next;
    };
    StackNode * top;      // top of stack
    int size;             // -1 if not set, value otherwise
    int current_size;     // unused if size = -1
  public:
    push(x : T) : void    // if size set, check for overflow
    pop() : T             // return element if not in underflow
    isEmpty() : bool      // true if empty, false otherwise
    Stack(int new_size)
    Stack()
```

(b) (4 points) Implement a queue which uses two stacks to simulate the queue behavior.

Problem 9.2 *Linked Lists & Rooted Trees* (10 points)

(a) (3 points) Write down the pseudocode for an in-situ algorithm that reverses a linked list of n elements in $\Theta(n)$. Explain why it is an in-situ algorithm.

(b) (4 points) Implement an algorithm to convert a binary search tree to a sorted linked list and derive its asymptotic time complexity.

(c) (3 points) Implement an algorithm to convert a sorted linked list to a binary search tree and derive its asymptotic time complexity. The search time complexity of the resulting binary search tree should be lower than the one for the equivalent sorted linked list.

How to submit your solutions

You can submit your solutions via *Grader* at **`https://grader.eecs.jacobs-university.de`** as a generated PDF file and/or source code files.
If there are problems with *Grader* (but only then), you can submit the file by sending mail to `k.lipskoch@jacobs-university.de` **with a subject line that starts with CH-231-A**.

Please note, that after the deadline it will not be possible to submit solutions. It is useless to send solutions by mail, because they will not be graded.

This homework is due by Monday, April 17th, 23:00.