HOMEWORK 7

ALGORITHMS AND DATA STRUCTURES (CH08-320201)

Spring 2018

Prof. Dr. Michael Sedlmair
Computer Science & Electrical Engineering
Jacobs University

Due on Monday, Apr 9, 2018, 23:55.

Problem 1: Stacks & Queues *(7+3=10 points)*

(a) Implement in Python or C++ the data structure of a stack backed up by a linked list, that can work for whatever type, and analyze the running time of each specific operation. In order to achieve this in C++, make use of $template < class T >$, on which you can find more information at Templates. Implement the stack such that you have the possibility of setting a fixed size but not necessarily have to (size is -1 when unset). Check for the correctness of your functions and throw exceptions with suggestive messages in cases of underflow or overflow(C++,Python). You can assume that if the size is passed, it always has a valid value.

```
class Stack[A]():
  private:
      struct StackNode(){ //linked list
        A data;
        StackNode *next;
        };
      StackNode *top;      //top of stack
      int size;            //-1 if not set, value otherwise
      int current_size;    //unused if size = -1
  public:
      fun push(x:A):void   //if size set, check for overflow
      fun pop():A          //return element if not in underflow
      fun isEmpty():bool   //1 is empty, 0 otherwise
      fun Stack(int new_size)
      fun Stack()
```

Checking will be done by case-tests alone. The tests will cover overflow and underflow cases, as well as tests for different types of variables.

(b) Show how a queue can be implemented with two stacks.

Problem 2: Linked Lists & Rooted Trees *(4+5+3=12 points)*

(a) Program an in-situ algorithm that reverses a linked list of n elements in $\Theta(n)$. Add an explanation to why it is an in-situ algorithm in the code.

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

(c) Program an algorithm to convert a sorted linked list to a binary search tree and derive its asymptotic time complexity.

Remarks

- Solutions have to be handed in via Moodle by the due date. For late submissions you need to get in contact with the TAs directly. You need to upload one zip-file that contains a PDF-file for the theoretical parts and source files (no executables or object files) for the programming assignments. The source files need to include a makefile. Programming assignments need to be handed in as **C++, or Python** code. **Please write your own code.** It is ok to take snippets from online resources, but they need to be clearly marked.
- Exercises marked with a * are bonus problems. These count towards your number of points for this homework. The maximum number of official points can not be exceeded.