JOHANNES KEPLER UNIVERSITÄT LINZ

Algorithms and Data Structures 1
Summer term 2021
Dari Trendafilov,
Stefan Grünberger

Assignment 05

Deadline: **27.05.2021, 23:59**
Submission via: **Moodle**

Elaboration time

*Remember the time you need for the elaboration of this assignment and document it in the file **time.txt** according to the structure illustrated in the right box. Please do not pack this file into an archive but upload it as a **separate file**.*

```
#Student ID
K12345678
#Assignment number
05
#Time in minutes
190
```

Priority Queue

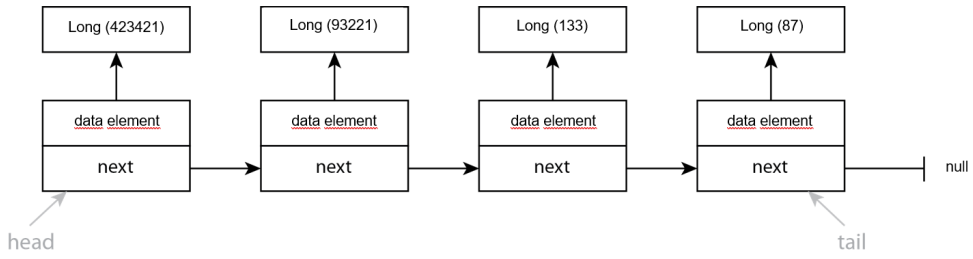
Please keep the provided skeletons unchanged, and complete your implementations as described below. The sources of **My_LinkedList.py**, **My_ListPriorityQueue.py** and **My_HeapPriorityQueue.py** must be submitted.

Always check the validity of arguments (*None* must not be inserted into the list). *Comment* those parts of your code, that are difficult to understand.

1. Priority Queue with Single Linked List — 10 points

Implement the abstract data type **Priority Queue** using a sorted (descending) **Single Linked List**.

a) The figure below shows an example of a single linked list with a head and a tail pointing at the beginning and at the end of the list, respectively.



Implement **My_LinkedList.py** which stores **integer** values sorted in **descending** order, based on the provided skeleton. For the list nodes use the provided class **My_ListNode.py**.

b) Implement the **Priority Queue** class in **My_ListPriorityQueue.py** based on the provided skeleton. The implementation should use the corresponding methods implemented in **My_LinkedList.py**.

2. Priority Queue with MaxHeap — 14 points

Implement the abstract data type **Priority Queue** using a **MaxHeap** (where the largest key is placed in the root) in **My_HeapPriorityQueue.py** based on the provided skeleton. For implementing the **MaxHeap** use a python list. Heap data shall be stored in a list (as shown below) and indexed as explained in the exercise material.

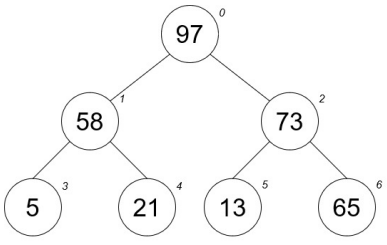


Fig.1a.: *Heap in tree structure*

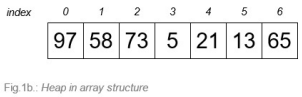


Fig.1b.: *Heap in array structure*

To make your code more readable, we recommend using methods as suggested below.

```
up_heap(index)
down_heap(index)
parent(index)
left_child(index)
right_child(index)
swap(index1, index2)
```