Institute of
Pervasive Computing

Algorithms and Data Structures 1
Summer term 2023
Jäger, Beck, Anzengruber

Assignment 5

Deadline CS: **Tue. 23.05.2023, 08:00**
Deadline AI: **Thu. 25.05.2023, 08:00**
Submission via: **Moodle**

Elaboration time

Remember the time you need for the elaboration of this assignment and document it in Moodle.

Priority Queue with Heaps

For this assignment, please submit the PDF of the pen-and-paper-work (example 2) and the source code of your **`min_heap.py`** implementation (example 1). As usual, don't change the given interface, but you can add auxiliary methods and reuse code where possible.

1. Priority Queue using a MinHeap 12 points

Implement the abstract data type **Priority Queue** using a **MinHeap** (where the smallest key is placed in the root) in **min_heap.py**, based on the provided skeleton. For implementing the **MinHeap**, use a python list to store and index data, as explained in the exercise material.

Make sure to implement and provide a working solution, as you need a working heap implementation for the next assignment 6 (sorting).

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)
```

Institute of
Pervasive Computing

Algorithms and Data Structures 1
Summer term 2023
Jäger, Beck, Anzengruber

Assignment 5

Deadline CS: **Tue. 23.05.2023, 08:00**
Deadline AI: **Thu. 25.05.2023, 08:00**
Submission via: **Moodle**

2. MinHeap & MaxHeap (Pen and Paper) — 8+4 points

a. Create an **array-based MinHeap** in the table below (i.e., a heap where the minimum is stored in the root) using the following sequence of numbers **[107, 79, n_1, 59, n_2, 62, 23, 47, n_3, 19, 24, n_4, 6]** from left to right, where n_1..n_4 are replaced by the corresponding parts of your student ID.

Example student ID: k $\underbrace{1\,2\,3}_{n_1}\,\underbrace{4\,5}_{n_2}\,\underbrace{6\,7}_{n_3}\,\underbrace{8}_{n_4}$

The array-based notation stores nodes „*line by line*" in the array, as presented in the exercise slides and as the following example (fig. 1) shows (for MaxHeap).

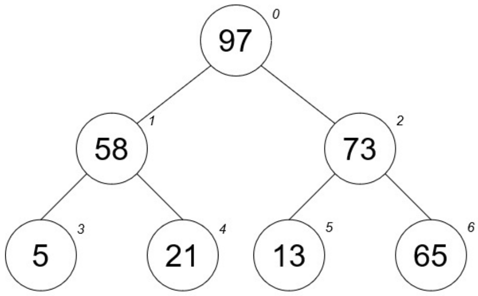

Fig.1a.: *Heap in tree structure*

index	0	1	2	3	4	5	6
	97	58	73	5	21	13	65

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

Each line in the table should represent the heap, after finishing one **insert** operation. Make sure that you always have a valid heap which fulfills the structure and the order property.

0	1	2	3	4	5	6	7	8	9	10	11	12
107												

b. Execute the `removeMax()` method on the following **MaxHeap** (i.e., a heap where the maximum is stored in the root). Each line in the table should represent one step of the algorithm (unlike 2.a., **show all up-/downheap operations stepwise**). Note briefly in the column *remarks* what you did on that line.

0	1	2	3	4	5	6	7	8	9	10	remarks
54	28	39	8	17	20	21	5	2	15	1	