Data Structures & Algorithms
Lab 1: Elementary Data Structures

Due on 15.02.2018

Federico Pecora, João Salvado

João Salvado

Handing In

This lab should be completed and shown during the lab occasion on 15.02.2018. During the next lab, I will pass by your seat and evaluate each exercise. Upon successful completion of the lab, for each lab exercise, please provide a text file named ex_n.txt with the following content:

- indicate which file(s) implement the algorithm and/or data structure in the exercise;
- a brief explanation of the tests that were carried out to test the implementation;
- instructions on how to execute a test to verify the implemented code;
- answers to any theoretical questions asked in the exercise. Please submit all lab material collected into an archive (zip, rar, or tar.gz) via a Blackboard message to João Salvado and Federico Pecora.

General Hints

- Most procedures can be found in the book (pp.229). Data structures may vary.
- Use **typedef struct** to define data structures and their elements

```
typedef struct list_element_t
{
    // Define elements of structure
} ListElement;
```

- Use malloc() and free() to allocate and free memory when creating pointers:

```
ListElement* x = (ListElement*)malloc(sizeof(ListElement));
// ... use x
free(x);
```

- All tasks below assume that integers are used as the keys of the data structures.
- We provide some test files containing numbers to test your sorting algorithms, please see the directory `Labs 1 and 2/Material/Sorting Problems` on the BlackBoard page.
- We also provide a test function that loads these test files into an array, please see the directory `Labs 1 and 2/Material/Load Sorting Problems (C Code)` on the BlackBoard page.
- Test your sorting implementations at least with the smallest files (10 numbers).

Exercise 0 — Reading

Please carefully read the document `Coding Style and Testing.pdf` found in the `Labs` directory on BlackBoard.

Exercise 1 — Linked Lists

Implement a double linked list of integers and provide all of the dynamic set operations list bellow. For additional information check book on page 230.

- List (L) and Node (N) - Create a data structure List and a data structure Node, which allows the implementation of double linked list.
- isEmpty(L*) - returns true if list is empty and false otherwise;
- insert(L*, N*) - inserts the node N into the list L, returns true if node was successfully inserted and false otherwise;
- search(L*, N*) - returns a pointer to the node N if node is in the List L, and NULL if N is not on the list;
- delete(L*, N*) - returns a pointer to the node N and N is deleted from the list, and NULL if deletion was not successful;
- maximum(L*) - returns a pointer to the node with the largest key.
- minimum(L*) - returns a pointer to the node with the smallest key.
- successor(L*, N*) - returns a pointer to the next larger node, or NULL if N is the maximum;
- predecessor(L*, N*) - returns a pointer to the next smaller node, or NULL if N is the minimum.

Exercise 2 — Testing (I)

Create two lists

$$L_1 = [3, 1, 5, 10, 8, 7]$$

and

$$L_2 = [5, 2, 9, 6, 1, 2]$$

Answer the following questions using your linked list implementation.

- What are the minimum and maximum of L_1?
- What are the minimum and maximum of L_2?
- What is the successor and predecessor of the node with key 5 in L_1?
- What is the successor and predecessor of the node with key 9 in L_2?
- What is the key of the predecessor in L_2 of the maximum of L_1?
- What is the key of the predecessor in L_1 of the maximum of L_2?

Exercise 3 — Stacks

Implement a stack with an array or with a linked list.

Exercise 4 — Queues

Implement a queue with an array or with a linked list.

Exercise 5 — Testing (II)

Test your implementations by making a stack/queue of your Swedish personal number (personnummer).

Exercise 6 — Testing (III)

Once you have completed the lab, test an exercise of a colleague and report which tests you conducted and the results of these tests.