Exercise 7

1. Some Linked List Data Structures

Using struct, declare the data structure of the following structures :

2. a linked list
3. a doubly linked list
4. a linked binary tree
5.

Pointer Manipulation

Define a data structure that corresponds to the sketch in Fig.1, and implement the function void swap_ptr() allowing to swap the two top elements as showed in Fig.1.

Figure 1: A data structure before and after 'swap_ptr();'

Figure 1: A data structure before and after 'swap_ptr();

Remark. To simplify the implementation of swap_ptr(), we assume that the data structure contains at least 2 nodes, i.e. no error checking has to be performed. Beware that your solution should also work when the structure contains only 2 nodes.

3. Queue

Study the linked_list_stack.c source code and transform it to implement a queue. A queue is a data structure in which objects are accessed in FIFO (First In First Out) order. New objects are inserted at the end of the queue (enqueue). Object is removed from the beginning (dequeue).

Hint. to easily access the end of the queue, you need to store an additional pointer to the last element of your linked list. Create a structure struct Queue which will store the head node and tail node of the list.

4. Graph

Study the minimal_tree.c source code. Suggest a new data structure to be able to express directed graphs instead of trees:

- Modify the struct node structure and the newNode function.
- Write an additional function connect which allows to connect two arbitrary nodes in your graph.

5. Project P01: Linked Data In-Memory Store

Think about the data structure you want to use for your storage. Describe it, its advantages and potential pitfalls.

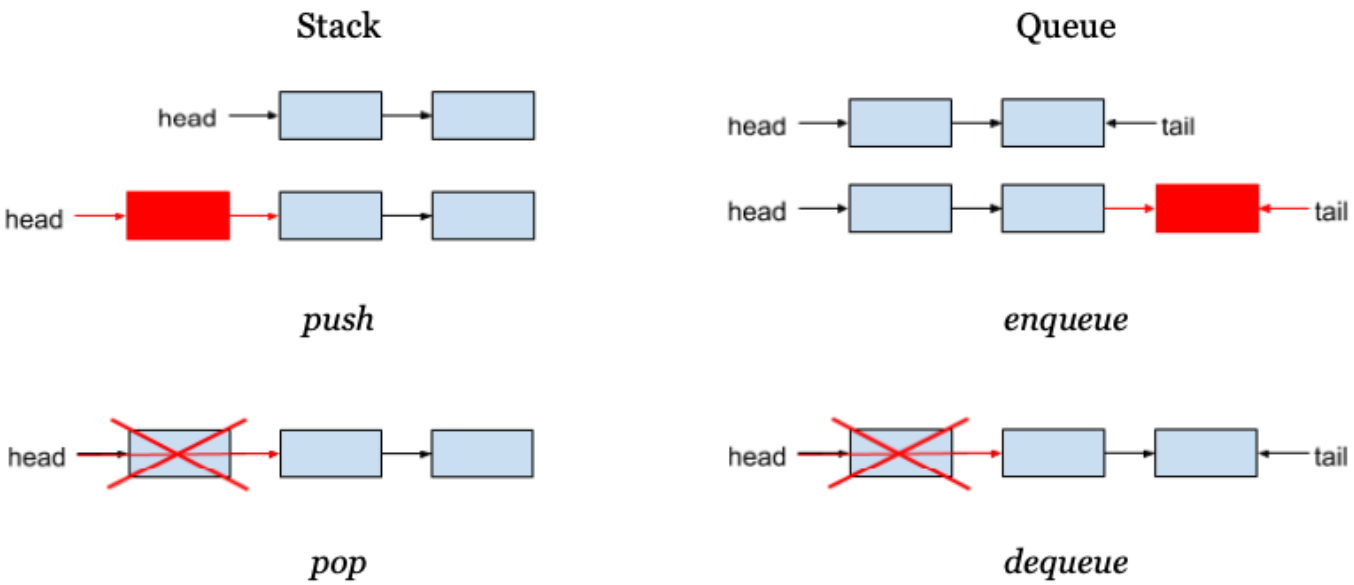

Figure 2: Stack and queue implemented with a linked list