Data Structures based on Pointers

1. A programmer made a mistake, so the pointer of the last element of a (singly) linked list now points to an element of this list. Construct an algorithm that fixess this issue and uses $O(1)$ RAM.

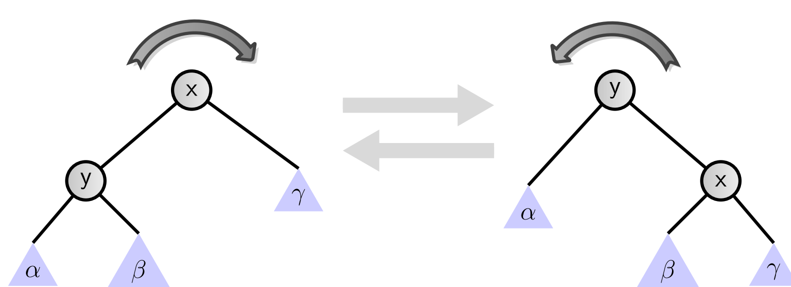
2. A binary search tree with numeric keys is stored in RAM. Construct an algorithm that receives two numbers l and r on the input and outputs all the keys k such that $l \leqslant k \leqslant r$ in the sorted order. The complexity is $O(h + m)$ where m is the number of the keys on the output.

3. An alternative method of performing an inorder tree walk of an n-node binary search tree finds the minimum element in the tree by calling `BSTMin` and then making $n - 1$ calls to `BSTNext`. Prove that this algorithm runs in $\Theta(n)$ time.

```
Function BSTNext(x) :
    if x → RightChild ≠ NULL then
        return BSTMin(x → RightChild)
    y := x → Parent;
    while  y ≠ NULL and x = y → RightChild do
        x := y;
        y := y → Parent;
    return y
```

```
Function BSTMin(x) :
    while  x → LeftChild ≠ NULL do
        x := x → LeftChild;
    return x
```

4. Rotate operations are defined via the picture below. A right rotate transforms a subtree with the root x of a binary search tree and the left rotate is the inverse transformation (of the subtree with the root y). Triangles α, β, and γ denote subtrees (maybe empty). Prove that a rotate applied to any node of a binary search (that has the corresponding child) tree results in a binary search tree.



5. You need to design a data structure PriorityQueue that stores key-priority pairs and has the following operations:

- `insert`(k, p) — adds the element with the key k and the priority p;
- `extract_max`$()$ — returns a pair (k, p) with maximal p;
- `set_priority`(k, p) sets the priority p to the key k.

Describe the implementation of PriorityQueue via Binary Search Trees so that each operation costs $O(h)$. You can use all the operations from the lecture having the complexity $O(h)$.

6 [Upgraded problem from homework]. The problem's input is numbers $n, k > 1$ and a list $a_1, \ldots, a_n$ of positie integers. Construct an $O(n \log k)$ algorithm that computes $\max\limits_{0<|i-j|\leqslant k} a_i \times a_j$, i. e. the maximal product of different elements with distance at most k. Try to construct an algorithm that uses $O(k)$ RAM (you can read the input sequence by elements).

7. Construct a data structure that supports the following queries (each in $O(h)$):

- Add key to the container;
- Delete key from the container (if there are duplicates, remove any);
- Find the k-th order statistic.

Keys are the elements of a (totally) ordered set.