Data Structures - Assignment no. 5

Remarks:

- **Write both your name and your ID number very clearly on the top of the exercise. Write your exercises in pen, or in clearly visible pencil. Please write *very* clearly.**
- **Give correctness and complexity proofs for every algorithm you write.**
- **For every question where you are required to write pseudo-code, also explain your solution in words.**

1. On an initially empty binomial heap, carry out the following sequence of operations: insert(27), insert(17), insert(19), insert(20), insert(24), insert(12), insert(11), insert(10), insert(14), insert(18), deletemin, decreasekey(19, 7), delete (17), decreasekey(24,5), deletemin.

   After each operation, draw the resulting structure of the binomial heap.

2. (a) Suppose that x is a node in a binomial tree within a binomial heap. Let sibling[x] be the successor of x in the doubly linked list containing it. If x is not a root, how does $degree[sibling[x]]$ compare to degree[x]? How about if x is a root? (degree[x] is the number of children of x)

   (b) If x is a non-root node in a binomial tree within a binomial heap, how does degree[$p[x]$] compare to $degree[x]$?

3. We wish to augment a binomial heap H to support an additional operation BINOMIAL-HEAP-SECOND-MIN(H) that returns the second smallest value in the heap. The operation should take O(1) (Note that the operation does not change the heap, it only return a value). The new structure should still support all the binomial heap operations with the same time complexity. Describe the required changes of the the binomial heap operations and how to implement the new operation. Prove the correctness of your implementation and analyze its complexity.

4. In class you have seen an implementation of delete for binomial heaps which moves the item to be deleted to the root by switching it with its parent and then continue like in deletemin. This implementation changes the nodes containing the items which were switched with the deleted item, thereby making it harder for applications to track where items reside. Describe an alternative implementation which does not change the locations of the items. (I.e. each item stays in the same node but nodes can move around.) Hint: Break apart the tree containing the deleted item and then combine the pieces.

5. We wish to implement the operation BINOMIAL-HEAP-SPLIT(H, r) on a binomial heap. The operation gets as an input a binomial heap H with n items and an integer r. The operation splits the binomial heap into two binomial heaps H_1 and H_2 such that H_1 contains r items and H_2 contains $n - r$ items. The partition of items is arbitrary. Suggest an implementation for this operation and prove its correctness and complexity.