American University of Armenia
CS121, Data Structures
PSS 7

Positional lists (LinkedPositionalList)

1) Implement a method, that for given 2 positions *p1* and *p2* checks if *p1* represents a position before *p2*, or vice versa. Note, in this problem it is not guaranteed that *p1* comes before *p2*.

2) Implement a method, that for given position *p* over a sorted PositionalLinkedList and given value *diff*, returns pair of such positions *p1* and *p2*, that value of *p1* is less than value of *p* by at most *diff*, and value of *p2* is greater than value of *p* by at most *diff*. The result positions *p1* and *p2* should be as far away from each other, as possible.

3) Implement bubble sort only over some range of PositionalLinkedList, that is specified with positions *[p1, p2)*.

4) Implement a method, that for given positions *p1* and *p2*, checks if content inside *[p1, p2]* is a palindrome.

Iterators (ArrayList)

1) Implement a method, that for given 2 iterators *it1* and *it2*, and given distance *D*, checks if *it1* and *it2* are at most *D* steps away from each other.

2) Implement a method that given an ArrayList and an iterator pointing to one of it's elements, returns if that iterator is closer to begin of ArrayList, or to the end of it.

3) Implement method that given 2 iterators *it1* and *it2* of same ArrayList, returns longest common substring, starting from that positions *it1* and *it2*.

If time permits

1) Implement such iterator over an ArrayList, which when traversed, will skip all 0 values. So it will traverse only over non-zero values of entire ArrayList.

2) Implement an iterator over a 2D array of predefined sizes (MxN). Iterators should traverse the matrix in row-major order, from left to right, and all rows from top to bottom.