Data Structures (IT205) 2013-14
First Midsemester-semester Exam
27^{th} August, 2013

Time: 2 hours **marks: 60**

This question paper consists of 3 questions printed on a single page. Check that your question paper is complete

1. Consider an array based implementation of a new linear list data structure which can behave at any time instant as either a queue or a stack. Thus, before performing an operation of insert or delete, it can change from a queue to a stack or vice-versa at will and also remain the same. This data structure has 4 elementary operations, PUSH, POP, ENQUEUE, DEQUEUE. The error conditions of empty and full are defined as usual, assuming an array capacity of n elements.

   (a) What is the running time of these four operations in the best possible algorithm?

   (b) Write an efficient routine for exchanging a pair of elements stored in this type of a data structure, using **only one** extra version of a similar data structure. (You are not allowed to use two extra arrays). What is the running time of the routine?

   (c) Write an efficient routine for reversing a subarray of elements between two specified positions, in this type of data structure using **only one** extra version and analyse the running time of the routine.

2. Write a routine, **almost** identical to insertion sort such that the resulting sequence is in alternating order. That is if the input sequence is $a_1, \ldots, a_n$ then the output permutation $b_1, \ldots, b_n$ should be such that $a_1 < a_2$, $a_2 > a_3$, $a_3 > a_4$ etc. In symbolic notation for all odd values of i, $a_{i-1} > a_i < a_{i+1}$ and for even values of i, $a_{1-1} < a_i > a_{i+1}$.

   The expected structure off the pseudocode is that an outer for loop starts from the second element of the sequence and using a while loop with an appropriate condition the i^{th} element is inserted into an appropriate position extending the sequence of the first $1 - i$ element alternating sequence by 1. Thus your routine should be provbly correct using the **loop invariant** concept. At the end of the i^{th} iteration, the first i elements must form an alternating sequence. You may assume that all elements have distinct key values.

   What is the running time of your routine?

3. Write a routine to **merge** two sorted sequences of n elements each into a single alternating sequence of length $2n$. You may assume all elements in the union of both lists are distinct. The procedure should resemble the merge routine of merge sort.

   Can you extend this process to achieve an alternating sequence with merge routine without the assumption that the two individual sequences are in sorted order? If so, notice that you can construct an alternating sequence in $\theta(n)$ time although sorting requires $\theta(n \log n)$ time.