A review will be held in 113 HRBB:

Sunday, October 18th & 25th (3:00–5:00 pm)

Topics Covered by Test 2

1. Stacks and queues implemented based on expandable/resizable arrays. Amortized cost for push and enqueue operations, p. 231.
2. Singly and doubly linked list data structures.
   (a) Write C++ functions for a linked list: insert a new node, delete a node, copy constructor, copy assignment operator, destructor, search for a given item, get the number of items in a list.
   (b) Provide running times for operations listed above.
3. Divide and conquer algorithms.
   (a) Write a running time recurrence relation based on the description of an algorithm.
   (b) Provide a description of an algorithm based on a given running time recurrence relation.
   (c) Solve a running time recurrence relation using the iterative and/or recursive tree method.
   (d) Use the Master theorem, if it is applicable, to obtain a running time of a recurrence relation.
4. Recursive sorting algorithms.
   (a) Merge sort
      i. How is the division of an array/subarray done by this algorithm during the partition step?
      ii. How many recursive steps are required by this algorithm to reach the base case (= an array with one element)?
      iii. Use the Big-O notation to estimate the number of comparisons done by this algorithm at each level of the recursive tree.
      iv. Is merge sort an in-place algorithm?
      v. What are recurrence relations for the best, average and worst cases of this algorithm?
      vi. Use the Big-O notation to provide running times for the best, average and worst cases of this algorithm.
   (b) Quick sort
      i. What are recurrence relations for the best, average and worst cases of running time of the algorithm?

ii. Use the Big-O notation to provide running times for the best, average and worst cases of the algorithm.
iii. Provide an input and select the pivot point to get the worst case.
iv. Provide an input and select the pivot point to get the best case.

5. Applications of the stack and queue ADT.

   (a) Stack and Queue ADT and their implementations.
   (b) Convert an algebraic expressions from infix form to postfix form using queue and stack. For example, $(x + y)^2 + (x - 4)/3$.
   (c) Use postfix form to evaluate an expression for $x = 2$ using a stack.
   (d) Build a binary expression tree starting from its postfix form using a stack. Illustrate all the merging steps during the construction of the tree.
   (e) Use the obtained binary tree to evaluate the expression for $x = 2$.

6. Binary search tree (BST).

   (a) Definitions and properties of proper and extended binary trees
   (b) Traversal operations: in-order, pre-order and post-order.
   (c) Complexity of the worst and best cases for insert, search, find, Min/Max and delete operations for balanced and unbalanced binary search trees.
   (d) Create a tree; insert and delete a value to an existing binary search tree; find max value in a tree; remove max value from a tree.
   (e) What is the complexity of building a balanced and unbalanced binary search trees?
   (f) Provide a sorting algorithm based on BST.
   (g) What is complexity of the sorting algorithms based on a balanced and unbalanced binary search tree.
   (h) Provide running times for all the above operations.