Written Assignment 2: Sorting, Trees and Graphs

1. **(15') Divide and Conquer**
   We are given an array of *n* items and for any two items *A* and *B*:
   - We can check if A and B are equal.
   - We cannot check which one is greater and which one is smaller, so we cannot sort them.

   For example, the array may be {♥,♥,☆,○,△,♥,♥,♥,♥,♫,☆,■,♥}. Our task is to find the *majority* of the array, if it has one. The *majority* of an array is defined as the item that appears strictly more than n/2 times. For example, in the sample array above, the array size is 13 and item ♥ appears 7 times, so ♥ is the majority.
   Use a divide-and-conquer algorithm to solve the problem in O(*n*log.n*). Describe your algorithm in *pseudo-code*.

2. **(21') Heaps and Heap Sort**
   a) *(5')* Does the array *{2, 8, 3, 18, 11, 4, 17, 16, 15, 7}* represent a minimum heap? Justify your answer.
   b) *(16')* Given the following input array:

   2 | 6 | 3 | 8 | 7 | 1 | 5 | 4

   you are required to sort the numbers into an incremental sequence using *heap sort*. Further, you are not allowed to use any extra arrays during the sorting procedure. Write down the content of the array whenever
   - an *insert* operation
   - or a *deleteMax* operation

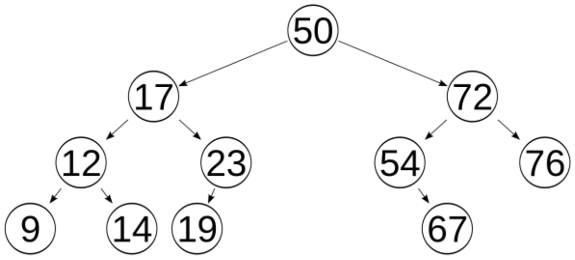
   of the heap is done. The first two steps are done for you. Please complete the rest.

   Initial state: 2 | 6 | 3 | 8 | 7 | 1 | 5 | 4

   Step 1 - insert(2): 2 | 6 | 3 | 8 | 7 | 1 | 5 | 4

   Step 2 - insert(6): 6 | 2 | 3 | 8 | 7 | 1 | 5 | 4

3. **(15') Binary Search Trees**
   a) *(5')* Show the final tree of inserting a sequence of numbers *{13, 27, 38, 93, 45, 82, 16, 65, 29, 77, 8*} into an empty *binary search tree*. You are *not* required to write intermediate trees.
   b) *(5')* Based on the following BST, draw the two possible binary search trees after deleting the root.



   c) *(5')* Show the result of a pre-order traversal of the tree in 3(b).

4. **(15') AVL Trees**

   Construct an AVL tree by inserting the input array *{13, 27, 38, 93, 45, 82, 16, 65, 29, 77, 8*}. Draw the state of the tree when each of the last three elements, 29, 77, and 8, are inserted, respectively.

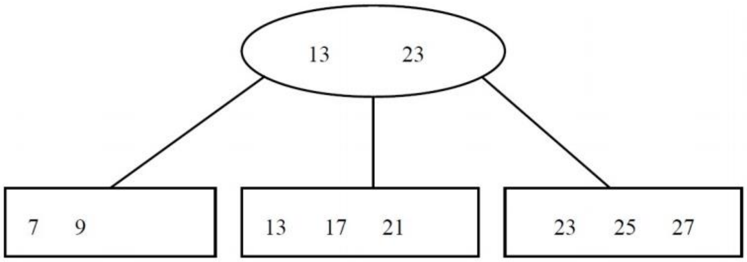
5. **(19') B+ Trees**

   a) *(4')* Suppose you are managing employee records on a computer with the following setting:

      - Computer hard disk access is block based and the size of one block is *2048 bytes*.
      - The size of each employee record is *256 bytes* (including the primary key).
      - The primary key for an employee record is of type *integer (4 bytes).*

      You decide to store the data using a B+ tree. Propose a best setting for *M* and *L*. Note: The definition of *M* and *L is* as described in the lecture slides.

   b) In this question, we use a B+ tree with *M=L=3*. The initial B+ tree in shown below

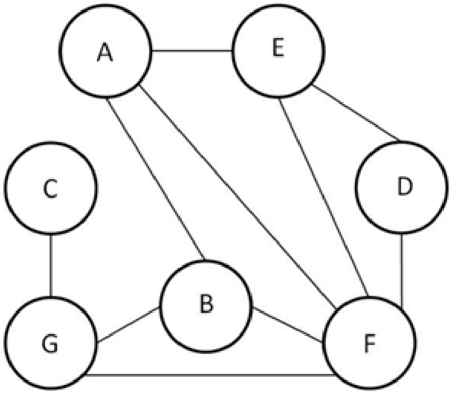


      i) *(5')* Given an insertion sequence *{11, 31, 19, 8, 4}*, show the B+ tree after each insert.

      ii) *(10')* Given an deletion sequence *{7, 9, 13, 23, 25}*, show the B+ tree after each deletion. Please start from the initial B+ tree shown in 5(b).

6. **(15') Graphs**

   Given the following graph:



   a) *(5')* Represent the graph using an adjacency matrix.

   b) *(5')* Represent the graph using an adjacency list.

   c) *(5')* Give the output sequence if the graph is visited using breath first search, starting from vertex *A*.

      Note: If multiple vertices are available during the search, process them alphabetically.