CSCD320 Homework4, Winter 2012, Eastern Washington University. Cheney, Washington.

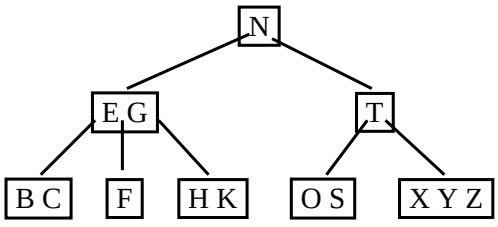
Name: **EWU ID:** **Due:** 11:59pm, Feb. 12, 2012 (Sunday)

Please follow these rules strictly:

1. Write your name and EWUID on **EVERY** page of your submission.
2. Verbal discussions with classmates are encouraged, but each student must independently write his/her own solutions, without referring to anybody else's solution.
3. The deadline is sharp. Late submissions will **NOT** be accepted (it is set on the Blackboard system). Send in whatever you have by the deadline.
4. Submission must be computer typeset in the **PDF** format and sent to the Blackboard system. I encourage you all to use the LaTeX system for the typesetting, as what I am doing for this homework as well as the class slides. LaTeX is a free software used by publishers for professional typesetting and nearly all the computer science and math professionals for paper writing.
5. Your submission PDF file must be named as: **firstname_lastname_EWUID_cscd320_hw4.pdf**
   (1) We use the underline '_' not the dash '-'.
   (2) All letters are in the lower case including your name and the filename's extend.
   (3) If you have middle name(s), you don't have to put them into the submission's filename.
6. Sharing any content of this homework and its keys in any way with anyone who is not in this class of this quarter is NOT permitted.

Problem 1 (25 points)**.** *Trace the ONE-PASS construction of the B-tree for the sequence* $\{M, C, P, H, G, X, S, U, O, A, W\}$*. Draw the configuration of the B-tree after inserting each letter. We use* $t = 2$ *as the branching degree threshold of the B-tree, so that: (1) all the non-leaf node must have at least* $t - 1 = 1$ *key and at most* $2t - 1 = 3$ *keys; and (2) The root node of an non-empty B-tree must have at least one key and at most* $2t - 1 = 3$ *keys.*

Problem 2 (25 points)**.** *Trace the deletion the sequence of keys* $\{G, S, Y, T, F\}$ *from the B-tree below. Draw the configuration of the B-tree after each deletion. We use* $t = 2$ *as the branching degree threshold of the B-tree, so that: (1) all the non-leaf node must have at least* $t - 1 = 1$ *key and at most* $2t - 1 = 3$ *keys; and (2) The root node of an non-empty B-tree must have at least one key and at most* $2t - 1 = 3$ *keys.*



Problem 3 (25 points)**.** *We know binary search trees support the operations of finding (1) the minimum and maximum node of a given subtree; and (2) the successor and predecessor of a given node in the tree.* **Now**

CSCD320 Homework4, Winter 2012, Eastern Washington University. Cheney, Washington.

Name: **EWU ID:** **Due:** 11:59pm, Feb. 12, 2012 (Sunday)

you are asked to present the algorithmic idea and the pseudocode of the operations below for B-trees. Give the time cost of your algorithms in the big-oh notation. *(Note: You can assume all the keys in the B-tree are distinct; Don't forget the DISK_READ operations.)*

- `B_tree_max(root)`
  - *input:* `root`.
    "`root`" is the reference to the root of the B-tree.
  - *output:* `(node, i)` *or* `NULL`.
    If the tree is not empty, then the "`i`"th key in the node "`node`" is the maximum key; Otherwise return `NULL`.
- `B_tree_predecessor(node, i)`
  - *input:* `node`, `i`
    A node referenced by "`node`" and its "`i`"th key.
  - *output:* `(predecessor_node, j)` *or* `NULL`.
    If the predecessor of the "`i`"th key of "`node`" exists, return the node referenced by "`predecessor_node`" whose "`j`"th key is the predecessor of the "`i`"th key of "`node`"; Otherwise, return `NULL`.
  - *Notes: (1) You can use the* `B_tree_max` *as a subroutine for* `B_tree_predecessor` *if needed; (2) You can assume that each node has a parent link that points to the node's parent, and the parent link at the B-tree's root points to NULL; (3) you can assume the given "`node`" and "`i`" are valid, meaning that they exist in the tree. (4) Make sure you consider all the possibilities.*

`B_tree_min` *and* `B_tree_successor` *are asymmetry to* `B_tree_max` *and* `B_tree_predecessor` *respectively, so you don't have to work on them.*

Problem 4 (25 points)**.** *We have raised the new challenges in external memory based sorting* [1]*. Suppose we need to sort a massive data set, which unfortunately cannot fit into the RAM of the machine. Propose any good idea for sorting this data set on external memory such as the hard disks, so that the performance will still be acceptable. Note that the guiding rule for external algorithm design is to make full use of and minimize the I/O operations, while the processing at the RAM is still maintained as efficient as possible. It is certain that you can find answers from the Internet. You are encouraged to do so and learn from it, but you must present what you learned as well as any other ideas from other sources or by yourself if you have in your own language. Provide the source of the information if they are not your own.*

[1] Page 5 of the slides of *B-trees (I)*, January 31, 2012.