Dept. of Computer Science and Engineering, undergraduate
National Sun Yat-sen University
Data Structures - Final Exam., Jan. 10, 2011

1. Multiple choices (There may be zero or more correct answers. If there is no correct answer, you should write down "None".) (28%)
   Answer: (a) AB (b) BD (c) ABC (d) BC (e) BC (f) D (g) AD

   (a) Which sorting algorithm(s) needs only $n-1$ comparisons in the best case, where n is the size of the input data? (A) Bubble sort (B) Insertion sort (C) Quicksort (D) Heapsort.

   (b) Which sorting algorithm(s) is of time complexity $O(n \log n)$ in the worst case, where n is the size of the input data? (A) Selection sort (B) Merge sort (C) Quicksort (D) Heapsort.

   (c) Which statement(s) is correct for a B^+-*tree*? (A) All leaves are on the same level. (B) All elements are maintained in leaf nodes. (C) The leaf nodes are linked as a list, so that finding the next element needs only $O(1)$ time. (D) Each nonleaf node stores only one element.

   (d) Which statement(s) is correct for the storage allocation method? (A) The best-fit method is to find the smallest memory block for the requirement. (B) The worst-fit method is to find the larget memory block for the requirement. (C) The first-fit method always spends no more time than the best-fit method, if the sizes of freed memory blocks are not sorted. (D) The worst-fit method wastes storage more than the best-fit and first-fit methods.

   (e) Which statement(s) is correct for the *hashing* method? (A) Hashing is a fast sorting method. (B) The memory allocated should not be less than the requirement for inserting data elements. (C) Hash collisions may occur when data elements are inserted. (D) An element deletion can be accomplished by first finding its position with the hashing function and then removing the element directly.

   (f) Which statement(s) is correct for a *binary tree*? (A) The *preorder* traversal in a binary tree is: root, right subtree, left subtree. (B) With the *postorder* traversal in a binary search tree, we can get a sorted sequence. (C) An *ordered tree* in which each node has at most two children is a binary tree. (D) An AVL tree is a binary tree.

   (g) Which statement(s) is correct for *binary search*? (A) Data elements should be sorted. (B) Binary search can be performed in a linked list. (C) Binary search can be performed in a disk file. (D) Binary search can be performed in an array.

2. Explain the *minimax* method on a game tree. (5%)

3. Define the *Fibonacci binary tree of order n* as follows: If $n = 0$ or $n = 1$, the tree consists of a single node. If $n > 1$, the tree consists of a root with the Fibonacci tree of order $n-1$ as the left subtree and the Fibonacci tree of order $n-2$ as the right subtree.

   (a) Draw the Fibonacci tree of order 4. (3%)

   (b) How many leaves are there in the Fibonacci tree of order n? (4%)

   (c) What is a *strictly binary tree*? Is a Fibonacci tree strictly binary? Why? (4%)

   (d) Is a Fibonacci tree an AVL tree? Why? (4%)

4. (a) What is the *radix sort*? Use 11, 27, 19, 25, 14, 34, 15, 35, 41, 21, 17, 24 as the input elements to illustrate the algorithm. (5%)

   (b) Why is the radix sort correct? Please explain. (5%)

5. An *optimal binary search tree* is to build a binary search tree according to the searching count of each key. Suppose we are given seven keys: 1, 2, 3, 4, 5, 6, 7 and their searching counts are 2, 10, 3, 1, 4, 8, 9, respectively. Please design a method to construct a near optimal binary serach tree. Use the above keys and counts to illustrate how you construct the tree. Note that you would get no score if you only draw the trees, but do not give the explanation. (10%)

6. Answer the following questions for a *buddy system*?

   (a) What method is used when a memory block is freed? (5%)

   (b) How do you determine whether a memory block of size 2^i is the first half or second half of a memory block of size 2^{i+1}? (5%)

7. A set of numbers are stored in a binary tree, but they are not ordered. For a given input element x, our job is to find the occurrence count of nodes whose value y has difference at most 3 compared with x. That is, if a node stores a number y, and $|x-y| \leq 3$, then the occurence count is increased by 1. Write a *recursive* C function to do this work. (10%)

```
struct nodetype {
    int info;
    struct nodetype *left;
    struct nodetype *right;
}
int count(int x, struct nodetype *tree)
```

8. The *height balance* of each node in a binary tree is defined as the height of its left subtree subtracted by the height of its right subtree. Write a *recursive* C function to calculate the

value of height balance of each node in a binary search tree. Note that the height of a tree containing only one single node is 1. (12%)

```
struct nodetype {
    int balance; /* height balance of this node */
    struct nodetype *left;
    struct nodetype *right;
}
int hbl(struct nodetype *tree)
/*  *tree: root */
```