# Dept. of Computer Science and Engineering, undergraduate
# National Sun Yat-sen University
# Data Structures - Final Exam., Jan. 11, 2010

1. (a) What is an *almost complete binary tree*? (5%)

   (b) How can an almost complete binary tree be implemented by a one-dimensional array? Hints: Point out the relationship between fathers and sons. (5%)

2. (a) What is the meaning of *stable sorting*? (4%)

   (b) Please give the stability of each of the following sorting algorithms: bubble sort, insertion sort, Shell sort, quicksort, radix sort. (10%)

3. The *odd-even transposition sort* proceeds as follows. Pass through the file several times. On the first pass, compare $x[i]$ with $x[i+1]$ for all odd $i$. On the second pass, compare $x[i]$ with $x[i+1]$ for all even $i$. Each time that $x[i] > x[i+1]$, interchange the two. Continue alternating in this fashion until the file is sorted. Use the following elements to explain how this algorithm works: 25, 48, 37, 12, 57, 86, 33, 92. (10%)

4. (a) In a *B-tree* of order $n$, how many elements are there in each node? (5%)

   (b) What is the difference between a *B-tree* and a $B^+$*-tree*? (5%)

5. Assume we use the *buddy system* to manage memory. In the buddy system, a block of memory size $2^i$ is called an $i$-block, and the $i$-list consists of starting addresses of free $i$-blocks. Suppose that the total memory size in our computer system is 1024 bytes, whose starting address is 0.

   (a) If a 6-block starts at address $p$, what is the starting address of its buddy? (5%)

   (b) Suppose 9-list=$\{0\}$, 8-list=$\{512\}$ and 7-list=$\{768\}$. Which block has been allocated? Please give its size and its starting address. (5%)

   (c) Under the allocation situation of the above problem, if we request two more blocks, with sizes 300 and 50, what is the content of each $i$-list, $5 \leq i \leq 9$ ? (5%)

   (d) Suppose initially 9-list=$\{0\}$, and 7-list=$\{512\}$. And the following blocks are allocated: $B_1$ starts at 896 with size 128, $B_2$ starts at 832 with size 64, $B_3$ starts at 768 with size 64, $B_4$ starts at 704 with size 64, $B_5$ starts at 640 with size 64. Now, $B_3$ and $B_4$ are freed. What is the content of each $i$-list, $6 \leq i \leq 8$? After that, if $B_5$ is further freed, what is the content of each $i$-list, $6 \leq i \leq 8$? (5%)

6. A set of numbers are stored in a binary tree, but they are not ordered. For a given input element $x$, we can find a nearest number by comparing $x$ with each number in the tree. That is, if $y$ is

the answer, then $|x-y|$ is the minimum. Write a *recursive* C function to do this work. (12%)

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

7. Write a *recursive* C function to perform the *merge sort*. To implement your merge sort, you can call the following *2-way merge* function as a basic function, which merges two sorted arrays into a single one. In other words, you need not write the 2-way merge function. (12%)

```
void twoway(int a[ ], int b[ ], int c[ ], int sizea, int sizeb)
/* a[ ] and b[ ] are input sorted arrays */
/* c[ ] is the output array after a[ ] and b[ ] are merged */
/* sizea and sizeb are the lengths of a[ ] and b[ ], respectively */
```

8. Write a C function for a binary search tree to insert a new key which is known not to exist in the tree. (12%)

```
struct nodetype {
    int key;
    struct nodetype *left;
    struct nodetype *right;
}
struct nodetype *insert(struct nodetype *tree, int k)
/*  *tree: root, k: new key
return: the root of the tree */
```