Dept. of Computer Science and Engineering, undergraduate
National Sun Yat-sen University
Data Structures - Final Exam., Jan. 14, 2013

1. Explain each of the following terms. (24%)
   (a) *almost complete binary tree*
   (b) *stable sorting*
   (c) *internal sorting*
   (d) *indexed sequential search*
   (e) *AVL tree*
   (f) *B-tree*

2. The following two methods can increase the performance of the sequential search. Please explain these two methods and give examples to illustrate the methods.
   (a) move-to-front method. (5%)
   (b) transposition method. (5%)

3. (a) What is *hashing*? Give an example to explain it. (5%)
   (b) What are the advantages and disadvantages of hashing? (5%)

4. Define a recursive function F as follows:
   $F(n) = 0$ if $n = 0$,
   $F(n) = n\%10$ if $(n\%10) > 0$,
   $F(n) = F(n/10)$ otherwise.
   You should write down how you calculate in the following two problems.
   (a) What is the value of $F(2)+F(3)+F(4)+\cdots+F(43)$? (5%)
   (b) What is the value of $F(23)+F(24)+F(25)+\cdots+F(1234)$? (5%)

   Answer: (a) 195 (b) 6044

5. (a) In a *buddy system*, what is the *buddy* of a memory block of size 2^i? (5%)
   (b) In a *buddy system*, what method is used for allocating a new memory block? (5%)

6. In memory management, there are three methods to put a segment of program into free memory: *first-fit*, *best-fit* and *worst-fit*. What are their meanings? What is the reason (advantage) for using each of them? (12%)

7. The *weight* of a node u, denoted as $w(u)$, in a binary tree is defined as the number of nodes contained in the subtree rooted at u, including u itself. A node u is *heavy* if $w(u) > \frac{w(p(u))}{2}$, where $p(u)$ is the parent of u. Note that a single node or the root of a tree is not counted as a heavy node. Write a *recursive* C function to calculate the number of heavy nodes in a binary tree. (12%)

```
struct nodetype {
    int info;
    struct nodetype *left;
    struct nodetype *right;
}
int heavy(struct nodetype *tree)
/*  *tree: root */
```

8. 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 na, int nb)
/* a[ ] and b[ ] are input sorted arrays */
/* c[ ] is the output sorted array after a[ ] and b[ ] are merged */
/* na and nb are the lengths of a[ ] and b[ ], respectively */
```