Dept. of Computer Science and Engineering, undergraduate
National Sun Yat-sen University
Data Structures - Final Exam., Jan. 19, 2009

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) BCD (b) BC (c) BD (d) ACD (e) D (f) C (g) BD

   (a) Which sorting algorithm(s) is a *stable* sorting method? (A) Selection sort (B) Insertion sort (C) Merge sort (D) Radix sort.

   (b) Which statement(s) is correct for an *almost complete binary tree*? (A) It is a *strictly binary tree*. (B) All leaf nodes can appear only on the two lowest levels. (C) At each node, the height of the left subtree is always greater than or equal to that of the right subtree. (D) The Huffman tree used in the Huffman encoding algorithm is an *almost complete binary tree*.

   (c) Which statement(s) is correct for an AVL tree? (A) The absolute value of height difference of two subtrees of each node is at most 1, but it may be more than 1 at the root. (B) When a node is added, it is always inserted as a leaf node. (C) After a rotation is performed, the sequence of preorder traversal is preserved. (D) When an insertion is performed, if there exists one unbalanced node and some rotations have been done for it, then the tree height is not changed.

   (d) Which statement(s) is correct for a *B-tree*? (A) All leaves are on the same level. (B) The number of elements stored in the root of a *B-tree* of order n is between $\lfloor \frac{n-1}{2} \rfloor$ and $n-1$. (C) The number of elements stored in a nonroot node of a *B-tree* of order n is between $\lfloor \frac{n-1}{2} \rfloor$ and $n-1$. (D) When a newly added element is inserted into a leaf, if the leaf is full, then the leaf will be split into two nodes.

   (e) Which statement(s) is correct for the *hashing* method? (A) We can get a sorted sequence by linearly scanning the hash table. (B) When we search an element k with the hashing function, suppose k is not at the position whose value is m. We can further search k in the lower part recursively if $k < m$, and in the upper part if otherwise. (C) We can delete one element already in the hash table by first finding its position with the hashing function and then deleting it. (D) Linear probing is a method for resolving hash collisions.

   (f) Which statement(s) is correct for the storage allocation method? (A) The first-fit method may be effective because it is based on the concept that it should waste the least storage. (B) The best-fit method is always better than the worst-fit method. (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.

(g) Which statement(s) is correct? (A) $O(n^d)$ is of exponential order, where d is a constant. (B) The best sorting algorithm based on comparisons requires $O(n \log n)$ time. (C) *Internal sorting* is to sort the elements in a file. (D) *Binary search* can be applied only when the elements are sorted.

2. (a) What is the *Quicksort*? Use 26, 5, 37, 1, 61, 11, 59, 15, 48, 19 as the input elements to illustrate the algorithm. (6%)

   (b) What is the time complexity of the algorithm in the best case? Give your analysis. (5%)

3. In an *optimal binary search tree*, the expected number of comparisons is minimized when the frequencies for searching the keys in the tree are given. Suppose the set of keys are $1, 2, 3, 4, 5, 6, 7$, and their corresponding searching frequencies are $2, 10, 3, 1, 4, 8, 9$. Explain the following two methods for constructing the near optimal binary search tree. In addition, draw the trees obtained by the methods with the above data. Note that you would get no score if you only draw the trees, but do not give the explanation.

   (a) The *balancing method*. (6%)

   (b) The method of *median split tree*. (6%)

4. (a) What is an *indexed sequential file*? How do we perform searching on that structure? (5%)

   (b) Why don't we use the binary search on that structure instead of sequential search? (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 5-block starts at address 544, what is the starting address of its buddy? How do you calculate? (3%)

   (b) Suppose 9-list=$\{0\}$, 7-list=$\{512\}$ and 6-list=$\{832\}$. How many blocks have been allocated? Please give their sizes and their starting addresses, respectively. (6%)

   (c) Suppose 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 change of the i-lists? (6%)

Answer:
(a) 512
(b) 7-block at 640 and 896, 6-block at 768
(c) 6-list={704, 768}, 7-list={512}, 8-list=empty
after that, 6-list={768}, 7-list=empty, 8-list={512}

6. Write a C function to delete the nth element from a linearly linked list, which is implemented by an array. You can assume that the length of the list is greater than n. Note that the first element is deleted when $n = 1$. (12%)

```
struct nodetype {
    int info;
    int next;
}
struct nodetype node[100];
void delete(int *list, int n)
```

7. Write a *recursive* C function to determine if a binary tree is *strictly binary*. Note that in a strictly binary tree, each nonleaf node has nonempty left and right subtrees. (12%)

```
struct nodetype {
    int info;
    struct nodetype *left;
    struct nodetype *right;
}
int strict(struct nodetype *tree)
/*   *tree : root
return 1 if the tree is strictly binary
return 0 if it is not */
```