Department of Computer Science and Engineering
National Sun Yat-sen University
Data Structures - Final Exam., Jan. 9, 2017

1. Multiple choices (There may be zero or more correct answers. If there is no correct answer, you should write down "None".) (16%)

   (a) Which statement(s) is correct for an optimal binary search tree? (A) The tree height is minimized. (B) The elements stored in the left subtree of the root are all less than or equal to the root. (C) The most frequently searched element is stored in the root. (D) The least frequently searched element is stored in a leaf node.

   (b) Which statement(s) is correct for a red-black tree? (A) The color of the root is black. (B) The height difference of two subtrees of any node is no more than 1. (C) All leaf nodes are on the same level. (D) In the path from the root to each external node, there are no two consecutive red nodes.

   (c) Which statement(s) is correct for the hashing technique? (A) Hashing is a fast sorting method. (B) If the chaining method is used, then the data elements with the same position index are chained by the linked lists. (C) The purpose for designing a dynamic hash table to solve the deletion problem. (D) Linear probing is to search the next available when a hash collision occurs.

   (d) Which statement(s) is correct for a B tree or a B+ tree? (A) All leaves in a B tree are on the same level. (B) All leaves in a B+ tree are on the same level. (C) 2-3 tree is a B tree. (D) 2-3 tree is a B+ tree.

2. Assume there are the following sorting methods: (A) selection sort (B) insertion sort (C) quick sort (D) heap sort (E) merge sort (F) radix sort. Answer the following questions with ABCDEF. (16%)

   (a) Which sorting method(s) is stable?

   (b) Which sorting method(s) has time complexity O($n$log$n$) in the worst case, where n denotes the number of input elements?

   (c) Which sorting method(s) requires additional arrays or linked lists when it is implemented on a computer?

   (d) An array is usually used for implementing a sorting method. However, some sorting methods can also be implemented by using linked lists. Please give the sorting methods which can be implemented with linked lists.

3. The *period* p of a string $A=a_1a_2...a_n$ is defined as the minimum positive integer satisfying that $a_i = a_{i+p}$, for all i, $i+p \in [1, n]$. If there is no such integer, then $p=n$. The *exponent* e of a string A is defined as $\frac{n}{p}$. A string A is a *repetition* if e is an

integer greater than 1. Please give *p* and *e* of each of the following strings, and answer whether it is a repetition or not. (9%)

(a) `atatat`

(b) `atata`

(c) `attt`

4. Starting with an empty AVL tree, suppose the key insertion sequence is 20, 10, 5, 30, 40, 57, 3. Draw the AVL tree after each insertion. (10%)

5. Huffman algorithm is a variable-length encoding method, in which a Huffman tree is constructed. Suppose seven symbols A, B, C, D, E, F, G with frequencies 10, 7, 6, 3, 15, 1, 6, respectively, are given. You have to obey the following rules for constructing the Huffman tree:

   (a) The label of each leaf node is a single symbol. The label of each internal node is the concatenation of the labels of its left child and right child.

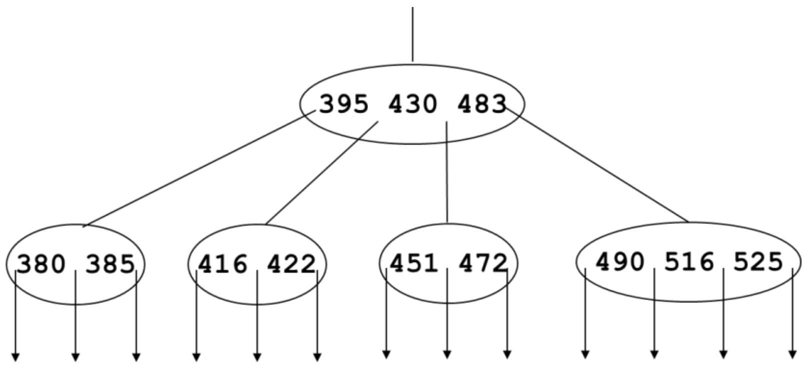
   (b) When two nodes are merged, always set the node of label with less lexical order as the left child, and the other as the right child.

   (c) If more than two nodes have the same least frequency, then you have to merge the two that have the least and the second least label in lexical order.

   Please draw the full Huffman tree. (10%)

6. Suppose that the division method (mod 11) is used in the hashing function. The linear probing method is applied when a collision occurs. Given input numbers: 19, 18, 7, 13, 3, 2, 14, please give the hash table after all numbers have been inserted into the table according to the input order. (10%)

7. Please draw the tree after 509 and 521 are inserted into the following B-tree of order 5. (7%)



8. The *internal path length I* is the sum of all paths from the root to all internal nodes. The *external path length E* is the sum of all paths from the root to all external nodes. For a strictly binary tree, in which every node has either no or two children, show that $E = I + 2n$, where *n* is # of internal nodes. (Hints: by induction) (10%)

9. 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 value *y*, and |*x*-*y*| ≤ 3, then the occurrence count is increased by 1. Write a *recursive* C++ function to do this work. (12%)

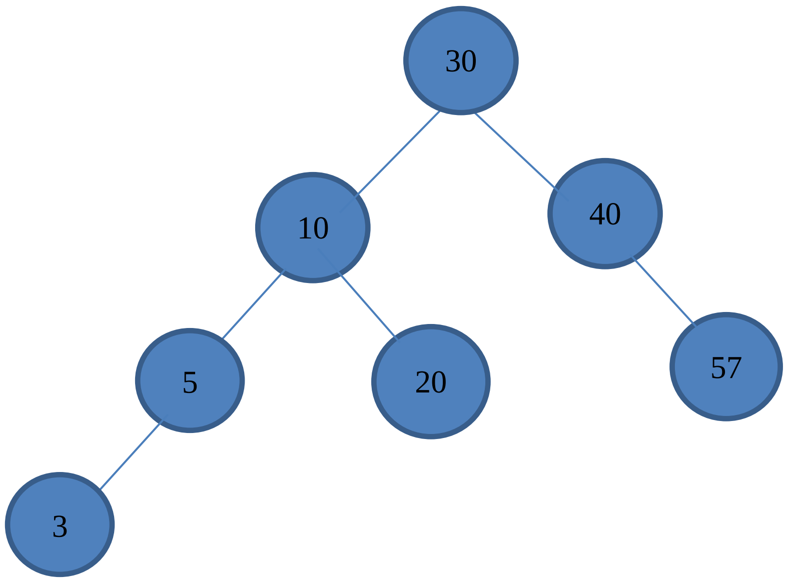
```
class TreeNode {
   int value;
   TreeNode *leftChild, *rightChild;;
};
int count(TreeNode *root, int x)
{
```

Please write the body of the function.

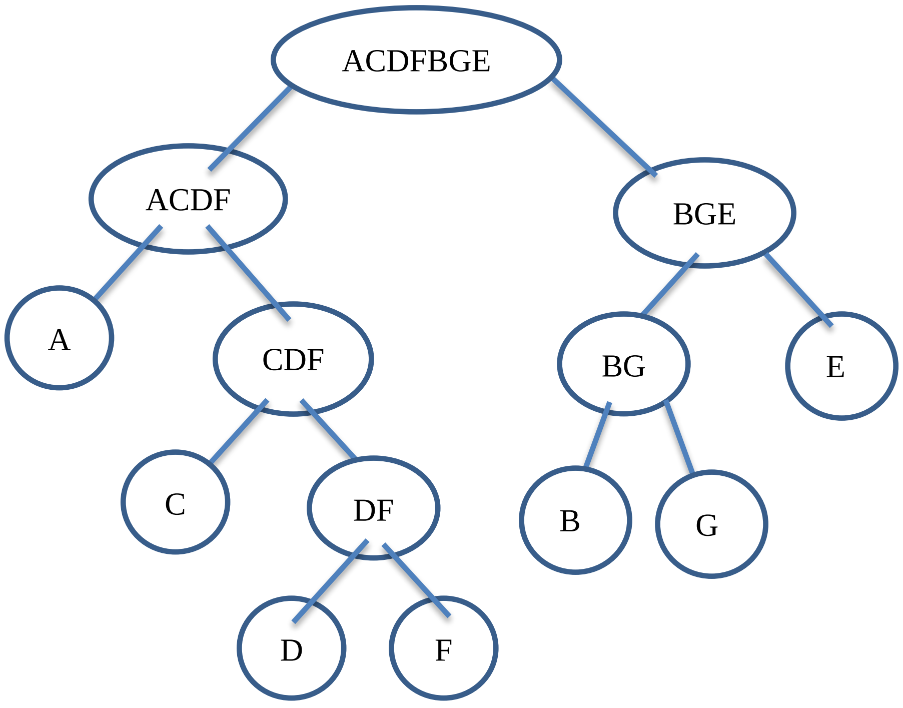
```
} // end of count( )
```

Answer:

1. B, AD, BD, ABC
2. Sorting methods (a) BEF (b) DEF or DE (c) EF (d) ABEF
3. String
   (a) p=2, e=3, repetition
   (b) p=2, e=2.5, not repetition
   (c) p=4, e=1, not repetition
4. AVL tree。

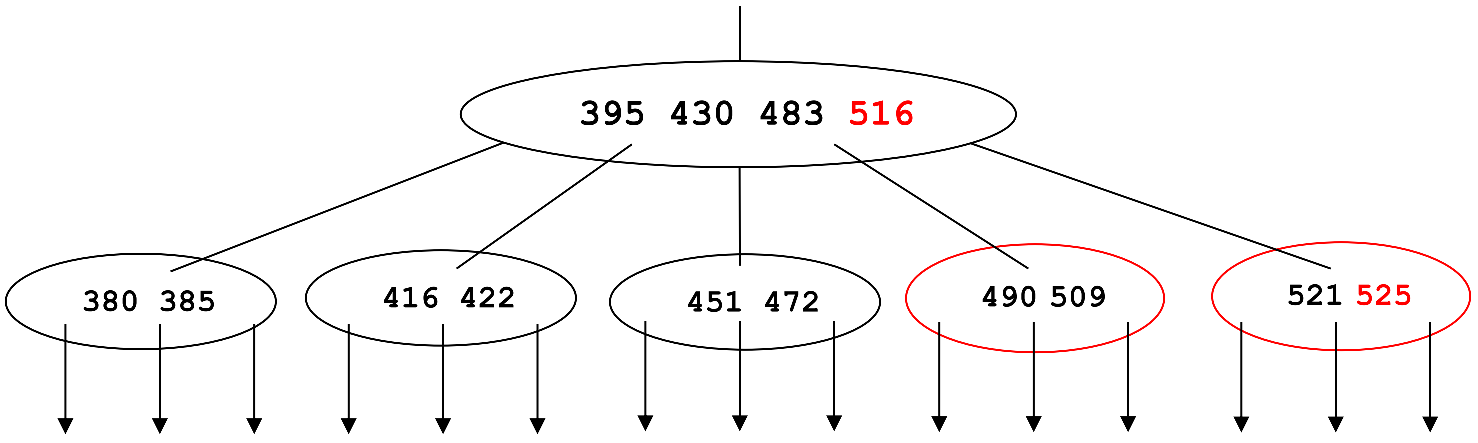


5. Huffman tree



6. Hash a[2]=13, a[3]=3, a[4]=2, a[5]=14, a[7]=18, a[8]=19, a[9]=7

7. B tree



8. 用數學歸納法來證明 binary tree 符合公式$E = I + 2n$。(題目敘述的 strictly binary tree 屬於 binary tree 的一種)
假設 n 為 tree 的 internal node 數量。

(1) 當 n=1 時，tree T_1只會有 1 個 internal node(root)，2 個 external node。此時，$E_1 = 2$，$I_1 = 0$，$n = 1$， $E = I + 2n$公式成立。

(2) 假設 n=k 時，$E_k = I_k + 2k$成立。

(3) 當 n=k+1 時，將 tree T_{k+1}其中一個深度為 d 的 leaf node 移除，成為只有 k 個 node 的 tree T_k。原先被移除的 leaf node 底下的 2 個 external node 的深度為 d+1，移除之後，該 leaf node 的位置變成一個 external node。因此，$E_k = E_{k+1} - 2(d + 1) + d = E_{k+1} - d - 2$。而該 leaf node 被移除了，因此，$I_k = I_{k+1} - d$。透過(2)的假設，$E_{k+1} - d - 2 = I_{k+1} - d + 2k$成立，得出$E_{k+1} = I_{k+1} + 2(k + 1)$，得證。

9.
```
class TreeNode {
    int value;
    TreeNode *leftChild, *rightChild;;
};
int count(TreeNode *root, int x)
{
    int leftM,rightM;
    if(root==0)
        return(0);      // Return 0 if the binary tree is empty.
    leftM=count(tree->left);
    rightM=count(tree->right);
    if ((abs(tree->value - x) <= 3)      // abs: absolute value
        return(1+leftM+rightM);          // count 1 for root
    else
        return(leftM+rightM);      // count 0 for root
} // end of count( )
```