# Fundamentals of Data Structures

## Mid-Term Exam (Full mark is 30 points)

October 12th, 2007

1. For a linked list, ______is(are) NOT correct. (1 points)
   A. Random access is always fast.
   B. No need to move elements when making insertion or deletion.
   C. No need to pre-estimate the size of space taken.
   D. Space taken is proportional to the number of elements in the list.

2. When open addressing is used to solve collisions, to delete a key from the hash table we must _______. (1 points)
   A. clear the cell that was occupied by the key
   B. find all the keys that have the same hash value of this key and shift them by 1 cell
   C. replace the key by a special value that signals the availability of the cell
   D. replace the key by the last inserted key which has the same hash value of this key

3. Suppose that five characters are pushed into a stack in the order of 'a b c d e'. The **impossible** popping sequence is _____ (1 point).
   A. e d c b a B. d e c b a C. d c e a b D. a b c d e

4. How many NULL links are there in a binary tree with *n* nodes? Prove your conclusion. (4 points)

5. Given a list of integers {4, 1, 3, 6, 5, 2, 7}. Please insert these integers according to the given order into a binary search tree which is originally empty. Please draw the resulting binary search tree.（4 points）

6. Given an array of the sequence { 70，48，24，65，35，12 }, please adjust the array into a min heap. The resulting sequence is:_______________ (3 points)

7. Find all possible binary trees of which the nodes are traversed in the exactly same order under
   (a) inorder and postorder traversals. (2 point)
   (b) preorder and postorder traversals; (2 point)
   (c) preorder and inorder traversals; (2 point)

8. Please fill in the blanks in the program which inserts **X** into a max-heap **H**. (2 points)

```
/* H->Element[ 0 ] is a sentinel */
void   Insert( ElementType   X,   PriorityQueue   H )
{    int   i;
     if ( IsFull( H ) ) {
          Error( "Priority queue is full" );
          return;
     }
     for ( i = ++H->Size; ①______________________; i /= 2 )
          ②_______________________________;
     H->Elements[ i ] = X;
}
```

9. Please write an algorithm of constructing a binary tree with given postorder and inorder sequences. The inputs are the inorder and postorder sequences stored in arrays **inorder[ ]** and **postorder[ ]**, and the number of elements is **n**. The output is the pointer to the root of the constructed tree. (8 points)

```
Tree_ptr Construct ( element_type inorder[ ], element_type postorder[ ], int n )
```

# Answer Sheet

1.A

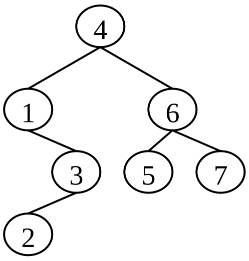
2.C

3.C

4. n+1

Total links = 2n
Non-NULL links = n-1
-> NULL links = 2n – (n-1)

5. {4, 1, 3, 6, 5, 2, 7}



6. { 70，48，24，65，35，12 }

<u>12, 35, 24, 65, 48, 70</u>

12, 35, 24, 70, 65, 48 ✕

7.
(in = post)
(a) The tree is empty or is skewed to the left

(pre=post)
(b) The tree is empty or has a single node only

(pre=in)
(c) The tree is empty or is skewed to the right

8.

① **H->Elements[ i / 2 ] < X**

② **H->Elements[ i ] = H->Elements[ i / 2 ]**

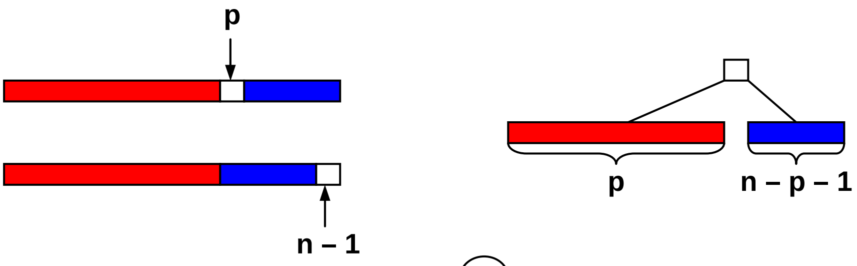
9.
```
Tree_ptr Construct ( element_type inorder[ ], element_type postorder[ ], int n )
{
        if (!n)
                return NULL;

        create the root with data postorder[n-1];

        p = position of the root data in inorder[ ];

        root->left_child = Construct( inorder, postorder, p );
        root->right_child = Construct( inorder+p+1, postorder+p, n – p – 1 );

        return root;
}
```



Example:

Inorder = { 1, 2, 3, 4, 5, 6, 7 }
Postorder = { 2, 3, 1, 5, 7, 6, 4 }