METU Department of Computer Engineering
CENG 213 Data Structures
Midterm II, December 13, 2012
120 min.

ID: _____

Name: _____

Section: _____

Q1 (21 pts)	
Q2 (23 pts)	
Q3 (24 pts)	
Q4 (18 pts)	
Q5 (14 pts)	
TOTAL	

Q1. (21 pts)

a) Answer the following questions:

i. What is the maximum number of levels in a binary tree of N nodes?

N

ii. What is the maximum number of nodes at level k in a binary tree?

2^{k-1}

iii. How many nodes are there in a full binary tree of k levels?

$2^{k}-1$

iv. Given a binary tree of height H, what is the minimum number of nodes that can be in this tree?

H+1

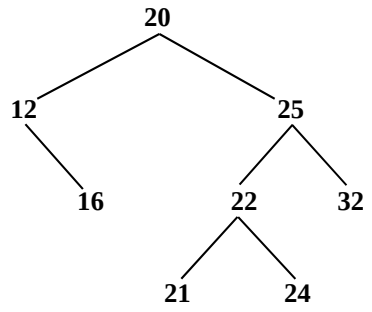
b) Construct the *Binary Search Tree* using the postorder traversal given as: **B A C F H G E I D**.
(Hint: You can deduce the inorder traversal of the corresponding binary search tree using the ordering of values in the nodes.)

Q2. (23 pts)

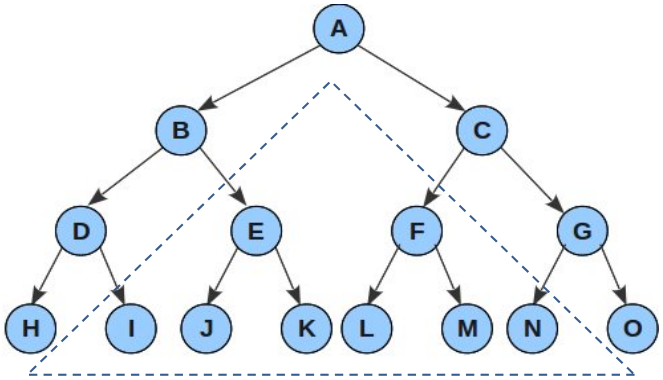
a) Consider the following sequence of numbers: **10, 11, 15, 8, 7, 9, 16, 4, 14**.
Construct an AVL-tree through successive insertions of the above sequence starting from an empty tree. Make sure that the AVL property is preserved after each insertion by applying the appropriate rotations. Draw the resulting tree after each -- one tree per box is adequate.

Insert 10, 11:	**Insert 15, 8:**	**Insert 7:**
Insert 9:	**Insert 16:**	**Insert 4, 14:**

b) Delete **16** from the following AVL tree. Draw the resulting tree in the box below.



Q3. (24 pts) When drawn on paper, a ***full binary*** tree looks like a ***triangle***. For example, the left edge of the triangle in the given figure consists of the nodes **A**, **B**, **D** and **H**. Similarly, the right edge of the triangle consists of the nodes **A**, **C**, **G** and **O**. Fill in the ***recursive*** "pne" (print-non-edges) function that takes a full binary tree and prints all of the nodes except the ones that are on the left and right edges. Do ***NOT*** declare any local variables of your own. The printed nodes should be in the same order as if the whole tree was traversed ***preorder***. For example, the output for the given figure should be **I E J K F L M N**.




```
struct Node {
  char   data  ;
  Node * left  ;
  Node * right ;
} ;

enum NodeType {
  TOP           ,
  ON_LEFT_EDGE  ,
  ON_RIGHT_EDGE ,
  NON_EDGE
} ;

void pne ( Node * tree , NodeType type ) {
  // Base case




  // Recursive part
  switch ( type ) {
    case TOP           :




    case ON_LEFT_EDGE  :




    case ON_RIGHT_EDGE :




    case NON_EDGE      :




  }
}

void main ( void ) {
  pne(root, TOP) ;
}
```

Q4. (18 pts) What is the output of the following mystery code? Assume that the stack or the queue never gets full; therefore you will not get any exceptions when you run the given code fragment.

```
void mystery ( char * array , int size ) {
  Stack S ;
  Queue Q ;

  for ( int i = 0 ; i < size ; i++ ) {
    if      ( i % 2 ) { S.push   ( array[i] ) ;                          }
    else if ( i % 4 ) { Q.enqueue( array[i] ) ;  Q.enqueue( S.pop() ) ; }
    else              { Q.enqueue( array[i] ) ;                          }
  }
  while ( ! S.isEmpty() )  {  Q.enqueue( S.pop() ) ;  }
  while ( ! Q.isEmpty() )  {  cout << Q.dequeue()  ;  }
}

void main ( void ) {
  char a[] = { 'A', 'B', 'C', 'D', 'E', 'F', 'G', 'H', 'I', 'J', 'K', 'L', 'M' } ;
  mystery(a,12) ;
}
```

Output -- ***one character per box*:**

Q5. (14 pts) Suppose you are given a queue of integers, currently holding only positive numbers. Use the fundamental queue operations below

```
bool isEmpty();
void enqueue(int element);
int  dequeue();
```

to complete the following non-recursive function that deletes all occurrences of the positive integer **X** in the queue. Other than this, the queue must remain unchanged. Do ***NOT*** declare any local variables of your own.

```
void deleteAllOccurrences(Queue & Q, int X) {
  int num;




  while(                        ) {




  }
}
```