CS212 (MT2) Name _____ ID _____

King Saud University
College of Computer and Information Sciences
Department of Computer Science
CS 212 Midterm 2 – 1st Semester 2011
Exam Duration : 2 Hours

Question 1 (30 Marks)

A. The following is the specification for the Queue ADT.

`public void enqueue(T e)`
requires: Queue Q is not full. **input:** T e.
results: Element e is added to the queue at its tail. **output:** none.

`public T serve ()`
requires: Queue Q is not empty.
results: the element at the head of Q is removed and its value assigned to e. **output**: T e.

`public int length ()`
results: The number of element in the Queue Q is returned. **output**: length.

`public boolean full ()`
results: If Q is full then flag is set to true, otherwise flag is set to false. **output**: flag.

During this course you have studied the *Stack* ADT, Using the above *Queue* ADT *Complete* the operations of the *Stack* ADT developed in the following class "StackQueue".

```
Public Class StackQueue<T> {
                Private LinkedQueue<T> Stack;

// the constructor of the LinkedQueue Class
Public  StackQueue(){
            Stack=new LinkedQueue<T>();
}
```

```
// If the Stack is full then this method will return
//true, otherwise it will return  false.
public boolean full(){

    return Stack.full();

}
```

```
// If The Stack is empty then flag is true, otherwise false.
public boolean empty(){

    return  Stack.length()==0;


}
```

```
// Element e is added to the stack as its most recently added
//elements.
public void push(T e){

    Stack.enqueue(e);




}
```

```
// the most recently arrived element in S is removed and its
//value is returned
public T pop(){

    for(int i=0; i<Stack.length()-1; i++)
        Stack.enqueue(Stack.serve());

    return Stack.serve();


}
}
```

B. Write a method named *Symbol_Balancer* which checks a string of character for matching Tokens, assume that your method only checks for parenthesis of these two types "( )" "[ ]". Your method will receive the String to be checked and will return a Boolean that will indicate if the string have any unmatched parenthesis or not.

(Note: the function "public char charAt(int i)" in ADT String returns the i^{th} character in the string)

```
String S ="Ahmed";
char x = S.charAt(0);// x value is 'A'
```

Example:

If your method receives this string :

This string have is (OK). ➔ **it will return true.**

There are problems) with this [String] ➔ **it will return false**

```
public Boolean Symbol_Balancer (String S){
    StackSS = new Stack ();
    for (int i=0; i < S.length(); i++)
    {  if (S.charAt(i) == '(' || S.charAt(i)=='[')
            SS.Push(S.charAt(i))
       else if (S.charAt(i) == ')') { char s = SS.Pop();
                                      if (s != '(') return false;}
       else if (S.charAt(i)==']') {char s = SS.Pop();
                                   if(s!='[') return false;}
    }
    if (SS.empty) return false;
    else
        return True;
}
```

Question 2 (35 Marks)

(A) Write a <u>recursive</u> *member* method in the ADT *Binary Tree,* named `findMax`, this method will move the *current* pointer to the tree node that have the largest data element.<u>(The Binary Tree ADT specifications is in Appendix A)</u>

```
Public void findMax (BTNode P)
{
    if (P != Null)
        if (P.Data > Current.Data)
            current = P;
        findMax (P.left);
        findMax (P.right);
}
```

(B) Write a recursive *member* method in the ADT *Binary Search Tree,* named `IsAVL` , this method will Check if the BST is an AVL and will return either true of false, Assume that you have the method:

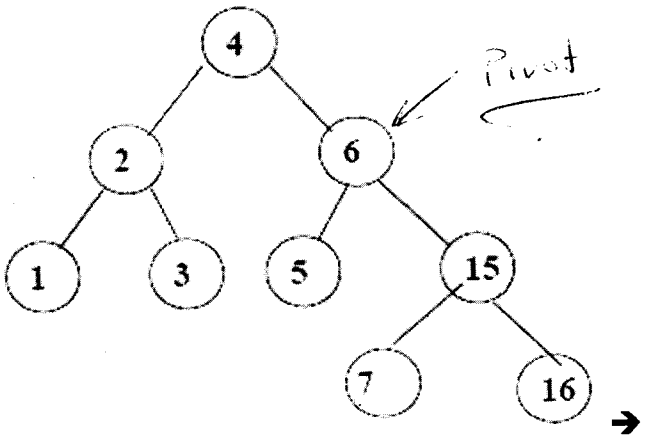
```
Public int getLength (BSTNode bt);
```

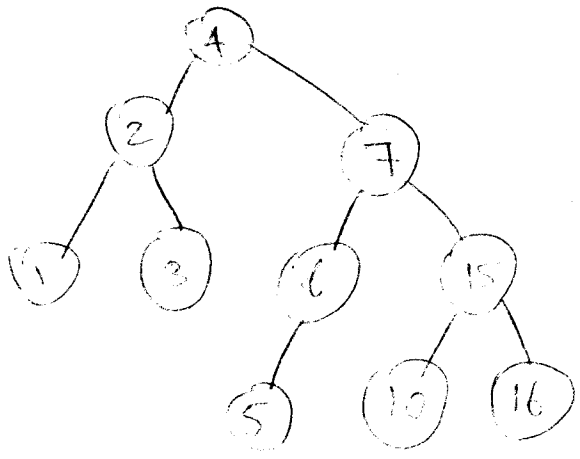
This method will return the length of the tree rooted with the BSTNode bt. (The Binary Search Tree ADT specifications is in Appendix A)

```
Public boolean isAVL (BSTNode P)
{
   if (P == null) return True;
   else
   { int balance = getLength(P.left) - getLength(P.right)
     boolean X = (balance >= -1) && (balance <= 1);

     return X && isAVL(P.left) && isAVL(P.right);

   }
}
```

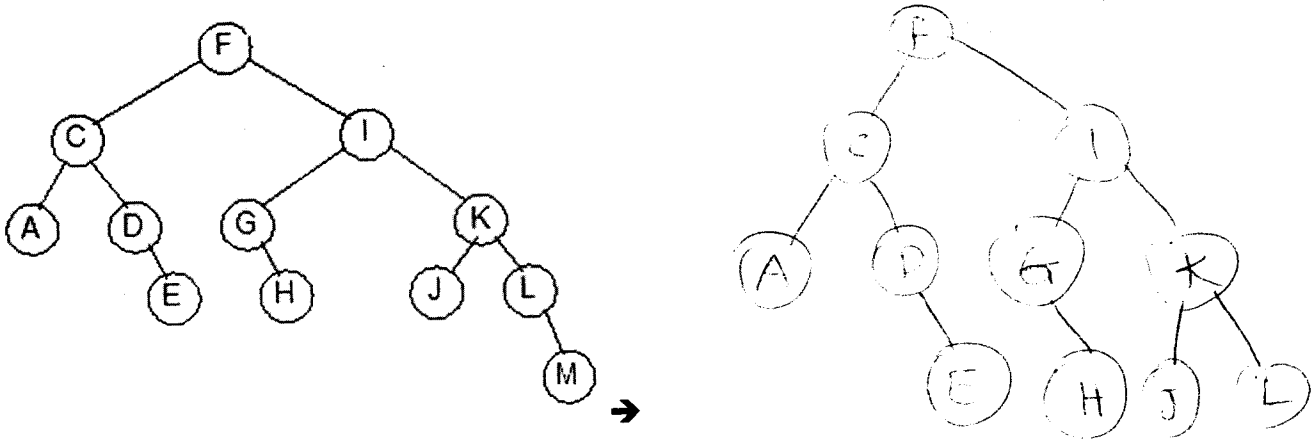
Question 3 (35 Marks)

A. Insert key '10' into the AVL tree shown below. After insertion indicate the pivot node and type of rotation needed to restore balance if any.

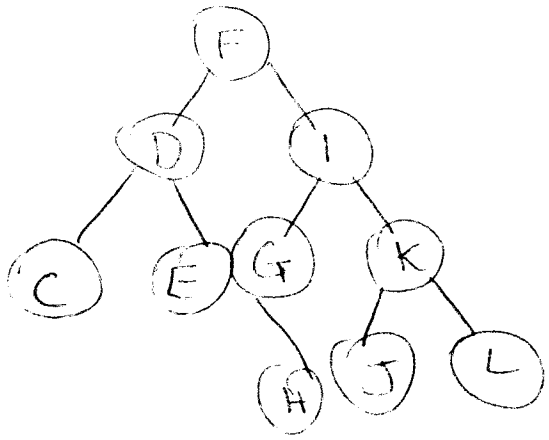




B. Delete key 'M' from the following AVL tree. Draw Tree after deletion.

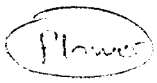


C. From the resulting tree in C (after deleting key 'M')Delete Key 'A'

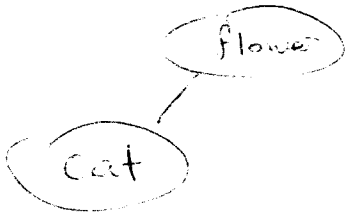


D. Insert following Keys in a Binary Search Tree. The tree is initially empty.
flower, cat, ink, ball, elephant, dust, king, mango
Note: {A,B,C,D,E,F,G,H,I,J,K,L,M,N,O,P,Q,R,S,T,U,V,W,X,Y,Z}

Step1

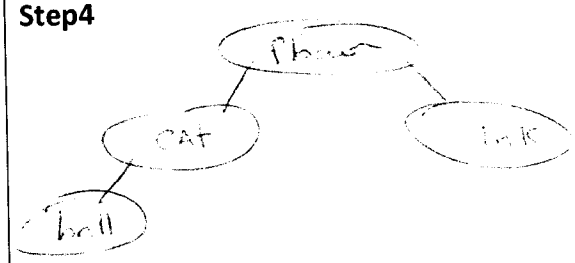


Step2

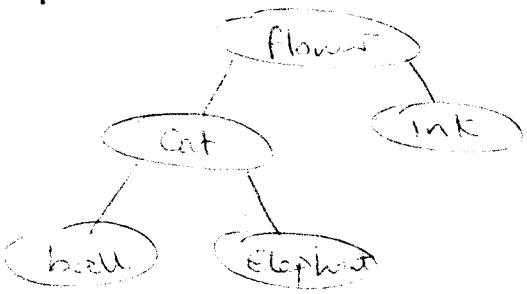


Step3

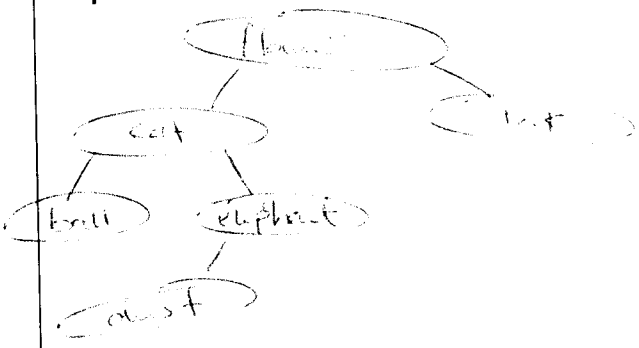
Step4



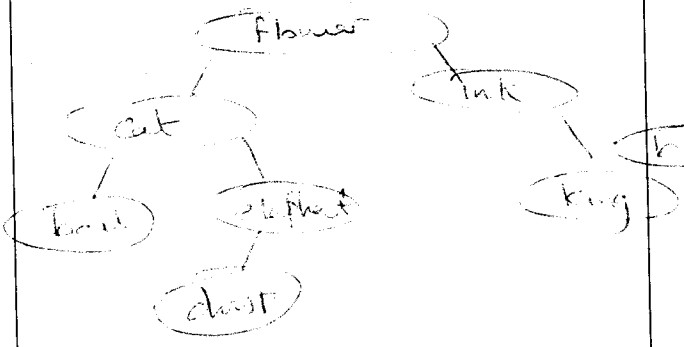
Step5



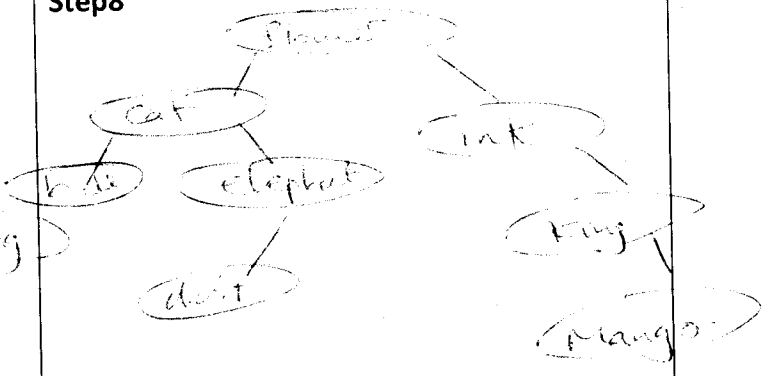
Step6



Step7



Step8



E. Delete 50 from the following BST and show the resulted tree.