King Saud University	College of Computer and Information Sciences
Data Structures CSC 212	Department of Computer Science
Date: 21/3/2019	**Quiz 4 - Spring 2019**
Student ID: [redacted]	Duration: 20 minutes
Sequential number:	Name: Saqd Ahmed Alshmibi
	Section/Instructor: Mohammed Sulmi 10-11

1	2	Total/40
30	10	40/40

Question 1 ..... 30 points

Write the **recursive** method `isMirror`, member of the class `BT`, that takes as input a binary tree and returns true if the two trees are the mirror image of each other. The method signature is **public boolean isMirror(BT<T> bt)**. This method calls the private recursive method **private boolean recIsMirror(BTNode<T> t1,BTNode<T> t2)**. Choose the correct option to complete the code of these methods:

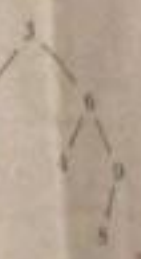
```
1 public boolean isMirror(BT<T> bt) {
2   ...
3 }
4 private boolean recIsMirror(BTNode<T> t1,BTNode<T> t2) {
5   ...
6   ...
7   ...
8   ...
9 }
```

1. Line 2:
   - (A) return isMirror(bt)&& isMirror(this);
   - (B) return isMirror(bt);
   - (C) return recIsMirror(root,bt.root);
   - (D) return recIsMirror(current,bt.current);
   - (E) None
2. Line 5:
   - (A) if (t1==null && t2==null)return true;
   - (B) if (t1==null || t2==null)return true;
   - (C) if (t1!=null && t2!=null)return false;
   - (D) if (t1==null || t2==null)return false;
   - (E) None
3. Line 6:
   - (A) if (t1!=null || t2!=null)return false;
   - (B) if (t1==null && t2==null)return true;
   - (C) if (t1==null || t2==null)return true;
   - (D) if (t1==null || t2==null)return false;
   - (E) None
4. Line 7:
   - (A) if (t1!=t2)return false;
   - (B) if (!(t1.data.equals(t2.data)))return false;
   - (C) if (t1.data.equals(t2.data))return false;
   - (D) if (t1.data.equals(t2.data))return true;
   - (E) None
5. Line 8:
   - (A) return recIsMirror(t1.left,t2.right);
   - (B) return recIsMirror(t1.right,t2.left);
   - (C) return recIsMirror(t1.left,t2.right)&& recIsMirror(t1.right,t2.left);
   - (D) return recIsMirror(t1.left,t2.left)&& recIsMirror(t1.right,t2.right);
   - (E) None

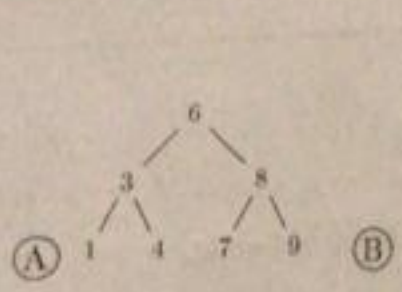
Question 2 ..... 10 points

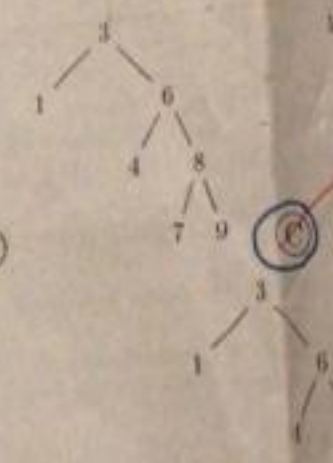
Convention: When necessary, the key must be replaced by the **smallest key in the right subtree**. Answers that do not follow this convention are considered wrong and receives the mark 0.

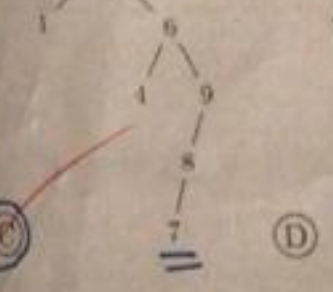
1. After inserting the key 7 in the BST

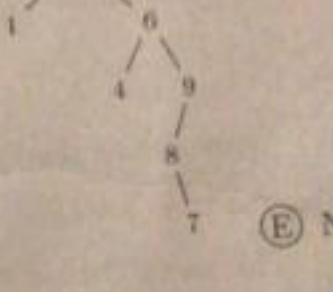


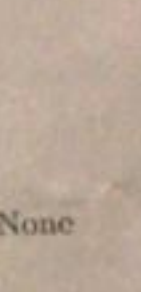
, the tree becomes:





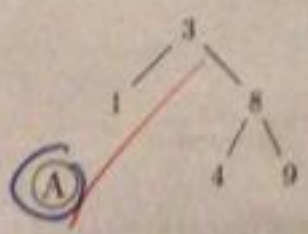


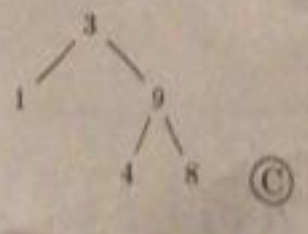


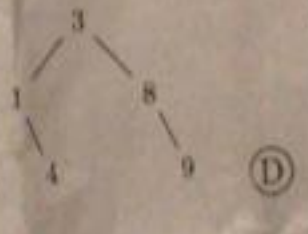


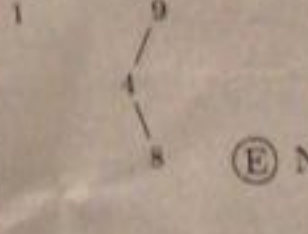
None

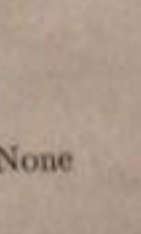
2. After deleting the key 6 from the BST











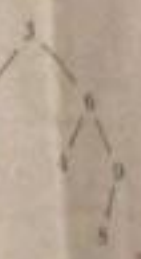
None

, the tree becomes:

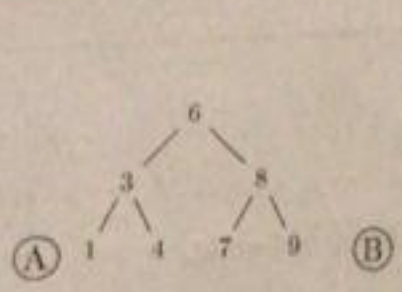
Question 2 ..... 10 points

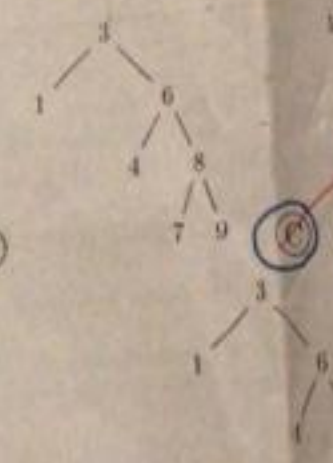
Convention: When necessary, the key must be replaced by the **smallest key in the right subtree**. Answers that do not follow this convention are considered wrong and receives the mark 0.

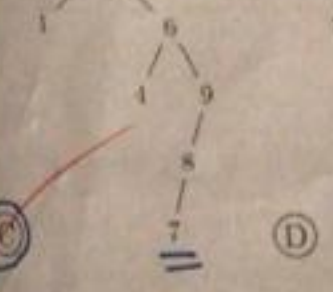
1. After inserting the key 7 in the BST

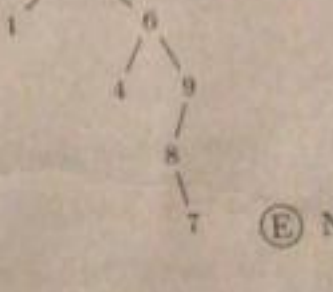


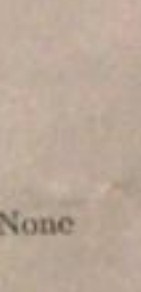
, the tree becomes:





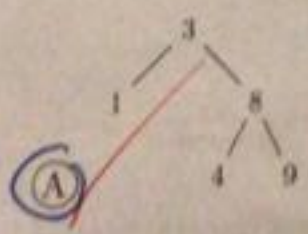


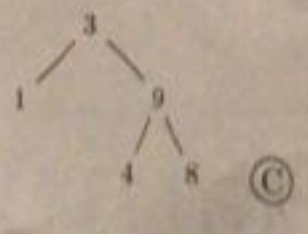


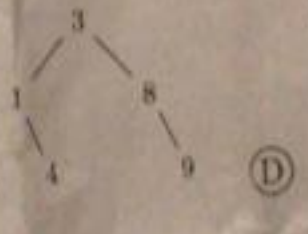


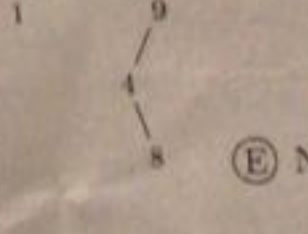
None

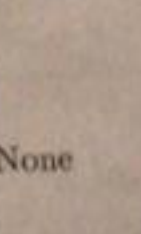
2. After deleting the key 6 from the BST









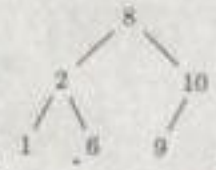


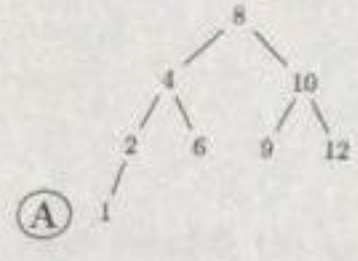
None

, the tree becomes:

Question 1 ..... 20 points

Choose the correct result in each of the following cases **(follow the the convention of replacing with the smallest key in the right sub-tree when necessary):**

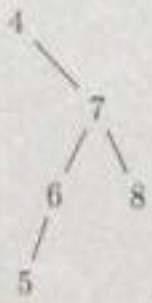
1. After inserting the key 4 in the AVL  , the tree becomes:



2. After inserting the key 11 in the AVL  , the tree becomes:

3. After deleting the key 9 from the AVL  , the tree becomes:

double rotation

4. After deleting the key 2 from the AVL  , the tree becomes:

Question 1 ..... 30 points

Write the **recursive** method `mirror`, member of the class BT, that transforms the tree into its mirror. The signature of the method is: **`public void mirror()`**. This method calls the private recursive method **`private void recMir(BTNode<T> t)`**. Choose the correct option to complete the code of these methods:

```
1  public void mirror() {
2    ...
3  }
4  private void recMir(BTNode<T> t) {
5    BTNode<T> p;
6    ...
7    ...
8    ...
9    ...
10 }
```

1. Line 2:

   (A) `recMir(current);`

   (B) `recMir(root);`

   (C) `recMir(root.left)&& recMir(root.right);`

   (D) `recMir(root.left)|| recMir(root.right);`

   (E) None

2. Line 6:

   (A) **`if (t == null)return;`**

   (B) **`if (current == null)return;`**

   (C) **`if (root == null)return;`**

   (D) **`if (t == null)return null;`**

   (E) None

3. Line 7:

   (A) `p = null;`

   (B) `t.left = t.right;`

   (C) `recMir(t.left);`

   (D) `p = t;`

   (E) `t.right = recMir(t.left);`

   (F) None

4. Line 8:

   (A) `t.right = t.left;`

   (B) `recMir(t.right);`

   (C) `p = t;`

   (D) `t.left = recMir(t.right);`

   (E) None

5. Line 9:

   (A) `p = t.right; t.left = t.right; t.right = p;`

   (B) `p = t.left; t.right = t.left; t.left = p;`

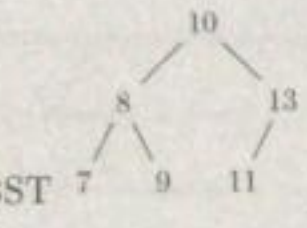
   (C) `p = t.left; t.left = t.right; t.right = p;`

   (D) `p = t.left; t.left = t.right; t.left = p;`

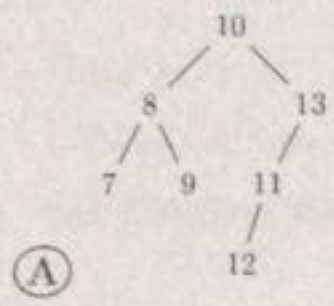
   (E) None

Question 2 ..... 10 points

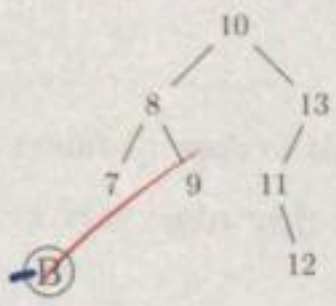
Convention: When necessary, the key must be replaced by the **smallest key in the right subtree**. Answers that do not follow this convention are considered wrong and receives the mark 0.

1. After inserting the key 12 in the BST 

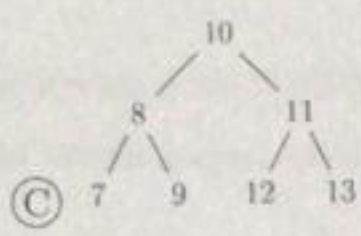
, the tree becomes:



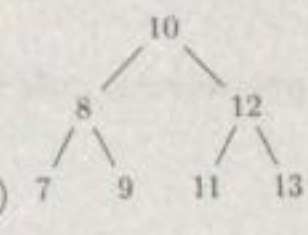
Ⓐ



Ⓑ



Ⓒ

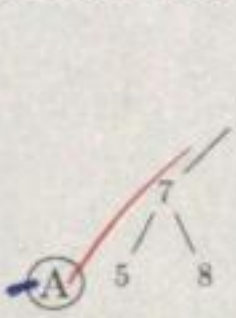


Ⓓ

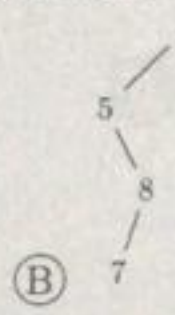
Ⓔ None

2. After deleting the key 6 from the BST 

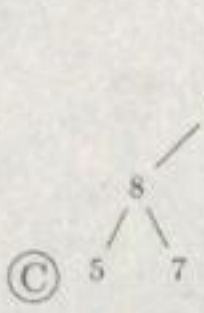
, the tree becomes:



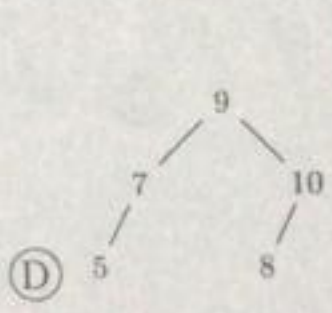
Ⓐ



Ⓑ



Ⓒ



Ⓓ

Ⓔ None

Question 1 ..... 30 points

Write the method **public boolean isFullTree()**, member of the BT class, which returns true if the BT is a full tree, and false otherwise. A BT is a full tree if every node other than the leaves has two children. This method calls the private recursive method **private boolean isFT(BTNode<T> t)**. Assume that the tree is not empty. Choose the correct option to complete the code of these methods:

```
1 public boolean isFullTree() {
2   ...
3 }
4 private boolean isFT(BTNode<T> p) {
5   ...
6   ...
7   ...
8   ...
9 }
```

1. Line 2:
   - (A) `return isFT(root);`
   - (B) `isFT(root);`
   - (C) `return isFT(root.left)|| isFT(root.right);`
   - (D) `return isFT();`
   - (E) None

2. Line 5:
   - (A) `if (p==null) return false;`
   - (B) `if (p==null) return;`
   - (C) `if (p==null) return true;`
   - (D) `if (p!=null) return true;`
   - (E) None

3. Line 6:
   - (A) `if(p.right!=null)return isFT(p.right);`
   - (B) `if(p.left!=null && p.right!=null)return true;`
   - (C) `if(p.left==null && p.right!=null)return false;`
   - (D) `if(p.left!=null && p.right!=null)return true;`
   - (E) None

4. Line 7:
   - (A) `if(p.left!=null && p.right==null)return false;`
   - (B) `if(p.left!=null || p.right!=null)return true;`
   - (C) `if(p.left==null && p.right!=null)return true;`
   - (D) `if(p.left!=null)return isFT(p.left);`
   - (E) None

5. Line 8:

Question 1 ..... 30 points

Write the method **public boolean** `isPathTree()` which is a member of the BT class. It returns true if the BT is a path tree, and false otherwise. A BT is a path tree if no node except root has two children. The method **public boolean** `isPathTree()` calls a **recursive** method **private boolean** `isPT(BTNode<T> p)`. Assume that tree is **not empty**. Choose the correct option to complete the code of these methods:

```
1 public boolean isPathTree() {
2   ...
3 }
4 private boolean isPT(BTNode<T> p) {
5   ...
6   ...
7   ...
8   ...
9 }
```

1. Line 7:
   - (A) `return isPT(root.left)&&isPT(root.right);`
   - (B) `return isPT(root.left)||isPT(root.right);`
   - (C) `return isPT(current.left)&& isPT(current.right);`
   - (D) `return isPT(root);`
   - (E) None
2. Line 5:
   - (A) `if (p != null)return true;`
   - (B) `if (p == null)return true;`
   - (C) `if (root == null)return true;`
   - (D) `if (p != null)return false;`
   - (E) None
3. Line 6:
   - (A) `if (p.left==null||p.right==null)return false;`
   - (B) `if (p.left!=null||p.right!=null)return false;`
   - (C) `if (p.left==null&&p.right==null)return true;`
   - (D) `if (p.left!=null&&p.right!=null)return false;`
   - (E) None
4. Line 7:
   - (A) `if (p.left!=null)return isPT(p.right);`
   - (B) `if (p.right!=null)return isPT(p.left);`
   - (C) `if (p.left!=null)return isPT(p.left);`
   - (D) `if (p.right==null)return isPT(p.right);`
   - (E) None
5. Line 8:
   - (A) `return false;`
   - (B) `return true;`
   - (C) `return isPT(p.right);`
   - (D) `return isPT(p.left);`
   - (E) None

```
1 public boolean isPathTree() {
2   ...
3 }
4 private boolean isPTRec(BTNode<T> p) {
5   ...
6   ...
7   ...
8 }
```

1. Line 2:
   - (A) `return ((isPTRec(root.left))&&(isPTRec(root.right)));`
   - (B) `return ((isPathTree(root.left))&&(isPathTree(root.right)));`
   - (C) `return ((isPTRec(current.left))&&(isPTRec(current.right)));`
   - (D) `return isPTRec(root);`
   - (E) None
2. Line 5:
   - (A) `if (p != null)return true;`
   - (B) `if (p == null)return true;`
   - (C) `if (root == null)return true;`
   - (D) `if (p != null)return false;`
   - (E) None
3. Line 6:
   - (A) `if ((p.left==null)||(p.right==null))return false;`
   - (B) `if ((p.left!=null)||(p.right!=null))return false;`
   - (C) `if ((p.left==null)&&(p.right==null))return true;`
   - (D) `if ((p.left!=null)&&(p.right!=null))return false;`
   - (E) None
4. Line 7:
   - (A) `return true;`
   - (B) `return isPTRec(p.left)&& isPTRec(p.right);`
   - (C) `return isPTRec(p.left)|| !isPTRec(p.right);`
   - (D) `return !isPTRec(p.left)|| !isPTRec(p.right);`
   - (E) None

) Consider the function f below, member of `DoubleLinkedList`:

```
public void f(int n) {
  Node<T> p = head, q;
  for(int i = 0; i < n; i++)
    if(p.next != null)
      p = p.next;

  if(p != null && p.next != null){
    q = p;
```

1	2	Total/40
18	16	28/40

Question 1 ..... 30 points

Write the method **public boolean isPathTree()** which is a member of the BT class. It returns true if the BT is a path tree, and false otherwise. A BT is a path tree if no node except root has two children. The method **public boolean isPathTree()** calls a **recursive** method **private boolean isPT(BTNode<T> p)**. Assume that tree is **not empty**. Choose the correct option to complete the code of these methods.

```
1 public boolean isPathTree() {
2   ...
3 }
4 private boolean isPT(BTNode<T> p) {
5   ...
6   ...
7   ...
8   ...
9 }
```

1. Line 2:
   - (A) return isPT(root.left)&&isPT(root.right);
   - (B) return isPT(root.left)||isPT(root.right);
   - (C) return isPT(current.left)&& isPT(current.right);
   - (D) return isPT(root);
   - (E) None

2. Line 5:
   - (A) if (p != null)return true;
   - (B) if (p == null)return true;
   - (C) if (root == null)return true;
   - (D) if (p != null)return false;
   - (E) None

3. Line 6:
   - (A) if (p.left==null||p.right==null)return false;
   - (B) if (p.left!=null||p.right!=null)return false;
   - (C) if (p.left==null&&p.right==null)return true;
   - (D) if (p.left!=null&&p.right!=null)return false;
   - (E) None

4. Line 7:
   - (A) if (p.left!=null)return isPT(p.right);
   - (B) if (p.right!=null)return isPT(p.left);
   - (C) if (p.left!=null)return isPT(p.left);
   - (D) if (p.right==null)return isPT(p.right);
   - (E) None

5. Line 8:
   - (A) return false;
   - (B) return true;
   - (C) return isPT(p.right);
   - (D) return isPT(p.left);
   - (E) None

(a) The method **private** BTNode<T> mirrorCopy(BTNode<T> t) creates **recursively** a mirror copy of the subtree t. Choose the correct option to complete the code of this method:

```
1 private BTNode<T> mirrorCopy(BTNode<T> t) {
2   ...
3   ...
4   ...
5   ...
6   ...
7   ...
8 }
```

1. Line 2:
   - (A) `if (t.left == null || t.right == null)`
   - (B) `if (t.left == null && t.right == null)`
   - (C) `if (t == null)`
   - (D) `if (root != null)`
   - (E) None
2. Line 3:
   - (A) `return null;`
   - (B) `return root;`
   - (C) `return mirrorCopy(root);`
   - (D) `return mirrorCopy(t);`
   - (E) None
3. Line 4:
   - (A) `BTNode<T> p = new BTNode<T>(t.data);`
   - (B) `BTNode<T> p = new BTNode<T>(root);`
   - (C) `BTNode<T> p = new BTNode<T>(t);`
   - (D) `BTNode<T> p = new BTNode<T>(root.data);`
   - (E) None
4. Line 5:
   - (A) `p.right = mirrorCopy(t.left);`
   - (B) `t.left = mirrorCopy(t.left);`
   - (C) `p.right = mirrorCopy(t.right);`
   - (D) `t.left = mirrorCopy(t.right);`
   - (E) None
5. Line 6:
   - (A) `t.right = mirrorCopy(t.left);`
   - (B) `p.left = mirrorCopy(t.left);`
   - (C) `p.left = mirrorCopy(t.right);`
   - (D) `t.right = mirrorCopy(t.right);`
   - (E) None
6. Line 7:
   - (A) `return p;`
   - (B) `return mirrorCopy(t);`
   - (C) `mirrorCopy(t.left); mirrorCopy(t.right);`
   - (D) `return t;`
   - (E) None

Question 1 ..... 30 points

Write the method **public boolean isPathTree()** which is a member of the BT class. It returns true if the BT is a path tree, and false otherwise. A BT is a path tree if no node except root has two children. The method **public boolean isPathTree()** calls a **recursive** method **private boolean isPT(BTNode<T> p)**. Assume that tree is **not empty**. Choose the correct option to complete the code of these methods:

```
1 public boolean isPathTree() {
2   ...
3 }
4 private boolean isPT(BTNode<T> p) {
5   ...
6   ...
7   ...
8   ...
9 }
```

1. Line 2:
   - (A) return isPT(root.left)&&isPT(root.right);
   - (B) return isPT(root.left)||isPT(root.right);
   - (C) return isPT(current.left)&& isPT(current.right);
   - (D) return isPT(root);
   - (E) None
2. Line 5:
   - (A) if (p != null)return true;
   - (B) if (p == null)return true;
   - (C) if (root == null)return true;
   - (D) if (p != null)return false;
   - (E) None
3. Line 6:
   - (A) if (p.left==null||p.right==null)return false;
   - (B) if (p.left!=null||p.right!=null)return false;
   - (C) if (p.left==null&&p.right==null)return true;
   - (D) if (p.left!=null&&p.right!=null)return false;
   - (E) None
4. Line 7:
   - (A) if (p.left!=null)return isPT(p.right);
   - (B) if (p.right!=null)return isPT(p.left);
   - (C) if (p.left!=null)return isPT(p.left);
   - (D) if (p.right==null)return isPT(p.right);
   - (E) None
5. Line 8:
   - (A) return false;
   - (B) return true;
   - (C) return isPT(p.right);
   - (D) return isPT(p.left);
   - (E) None