Prob. 1 / 1

```
    private boolean areMirror(BTNode<T> t1, BTNode<T> t2)
    {
        if (t1 == null && t2 == null)
            return true;

        if (t1 == null || t2 == null)
            return false;

        return  t1.data == t2.data && areMirror(t1.left,
t2.right) && areMirror(t1.right, t2.left);
   }
```

Prob. 1 / 2

```
    private void swap(BTNode<T> t)
    {
     if (t != null)
     {
          if (t.left != null)
               t.left.data = t.data;
          else if (t.right != null)
               t.right.data = t.data;

          swap(t.left);
          swap(t.right);
     }
    }
```

Prob. 2 / 1

```
     public static <T> LinkList<BTNode<T>>
collectLeaves(BT<BTNode<T>> bt) s{

          LinkList<BTNode<T>> l = new LinkList<BTNode<T>>();
          LinkStack<BTNode<T>> nodes = new
LinkStack<BTNode<T>>();
          bt.find(Relative.Root);
          nodes.push(bt.retrieve());

          while (! nodes.empty()){
               BTNode<T> current = nodes.pop();

               if (current.right != null)
                    nodes.push(current.right);

               if (current.left != null)
                    nodes.push(current.left);

               if (current.left == null && current.right== null)
                    l.insert(current);
          }
          return l;
     }
```

OR

```
     public static <T> LinkList<BTNode<T>>
collectLeaves(BT<BTNode<T>> bt,LinkList<BTNode<T>> l)
     {
          if (! bt.empty())
          {
               if (bt.find(Relative.LeftChild) == true)
                    l = collectLeaves(bt, l);

               if (bt.retrieve().left == null &&
bt.retrieve().right == null)
                    l.insert(bt.retrieve());

               if (bt.find(Relative.RightChild) == true)
                    l = collectLeaves(bt, l);
          }
          return l;

     }
```

Prob. 2 / 2

```
   public LinkList<T> collectLeaves()
   {
     LinkList<T> l = new LinkList<T>();
     return collectLeaves(root,l);
   }

   private LinkList<T> collectLeaves(BTNode<T> t,LinkList<T> l)
   {
     if (t != null)
     {
          l = collectLeaves(t.left,l);

          if (t.left == null && t.right == null)
               l.insert(t.data);

          l = collectLeaves(t.right,l);
     }

     return l;
  }
```

Prob 4 – 1

```
private void swapData(int k)
 {
  if (! empty())
  {
       if(findkey(k))
       {
            T val = null;
            BSTNode<T> q = new BSTNode<T>(k,val);
            BSTNode<T> p = findparent(q);
            update(k, p.data);
       }
  }
 }
```

Prob 4 -2

```
private void inorder(BSTNode<T> p)
{
    if (p != null)
    {
      inorder(p.right);
      System.out.println(p.key);
      inorder(p.left);
    }
}
```