Assignment #1:

- There are many ways to represent binary trees (equivalently ordered forests) as sequences of integers of various types. The purpose of this first problem is to explore two of those representations, converting from the sequence to the binary tree and vice-versa. Binary trees will be represented using nodes as defined in the class shown below

```
class BT {
  BT L; BT R;
  BT( BT l, BT r ) { L = l; R = r; }
}
```

Binary trees will be represented together with some sequence representation by extending the abstract class shown below.

```
abstract class TreeAndRepresentation {
   int N;   // number of nodes in the binary tree t.
   int M;   // number of entries in the sequence representation;
   int[] a; // a sequence representation of a tree.
   BT t;    // a binary tree;
   TreeAndRepresentation( int N, BT t )    { this.t = t; this.N = N; }  // subclass should set a,M.
   TreeAndRepresentation( int M, int[] a ) { this.a = a; this.M = M; } // subclass should set t,N.
}
```

You should extend `TreeAndRepresentation` to two classes called `ZOrep` and `FLrep`.

First Representation (ZO): Let T be a binary tree with n nodes. Perform a preorder traversal of T, recording a 1 as each node is encountered and a 0 for an empty subtree, to obtain a binary "tree sequence". For example, the binary tree shown at the top of page 396 in the book produces the tree sequence `111010011000100`. This is like a well-formed parentheses string where a 1 is a left paren and a 0 is a right paren, if we ignore the final 0; in our example, we get `((()())(()))()`. Furthermore, we can reverse the process, starting from a well-formed parentheses string to produce a unique binary tree.

Here is a partially filled in template to get you started. Make sure that you understand how `build` works before moving on to the second representation.

```
class ZOrep extends TreeAndRepresentation {
  private int k;
  ZOrep( int m, int[] b ) { // given sequence build tree
     super( m, b );
     N = (M-1)/2;
     k  = -1;
     t = build();
  }
  ZOrep( int n, BT t ) { // given tree build sequence
      // YOU SHOULD ADD SOME CODE HERE (including a call to super).
      traverse( t );
  }

  BT build() { return( a[++k] == 0 ? null : new BT( build(), build() )); }

  void traverse( BT t ) {
      // FOR YOU TO FILL IN
  }
}
```

Second Representation (FL): Consider the following process on a given ordered forest F:

  - Labels the nodes of the forest according to their depth, with roots having depth 0.
  - Convert F to the corresponding binary tree B (under the left-child right-sibling correspondence).
  - Traverse B in inorder to get a sequence (array) `a[0],a[1],...,a[N-1]` where N is the number of nodes in F.

For example `a[1..9] = 2 1 2 1 0 0 1 0 0` is such a sequence, as is `4 6 6 5 4 3 3 5 4 3 6 5 4 4 4 6 5 4 3 5 4 3 2 3 2 1 0 1 0 0`. As above, you need to write the constructors that convert back and forth between the tree and the sequence.

```
class FLrep extends TreeAndRepresentation {
  // VARIABLES, IF ANY NEEDED
  FLrep( int m, int[] b ) { // given sequence build tree
    // YOU FILL IN THE CODE
  }
  FLrep( int n, BT t ) { // given tree build sequence
    // YOU FILL IN THE CODE
  }
  // ANY ADDITIONAL METHODS GO HERE
}
```

We will test your program using programs somewhat similar to that shown below. You are advised to test your program with this code. Note that `StdIn` coomes from the book's `algs4.jar` file. It is similar to what you would find in the `Scanner` class (which could be used instead). NOTE: you could also declare `a` as an `int[M]` in line 4 below and have a perfectly correct solution; we used the extra entry to store a *sentinel* which made one of our loops simpler in Frank's solution to the problem.

```
class TestAss1 {
   public static void main ( String[] args ) {
      int M = StdIn.readInt();
      int a[] = new int[M+1];  /* NOTE */
      int b[] = null;
      for (int i=0; i<M; ++i) a[i] = StdIn.readInt();
      boolean testZO = false;
      if (testZO) {
        ZOrep zo1 = new ZOrep( M, a );
        ZOrep zo2 = new ZOrep( zo1.N, zo1.t );
        b = zo2.a;
      } else {
        FLrep fl1 = new FLrep( M, a );
        FLrep fl2 = new FLrep( fl1.N, fl1.t );
        b = fl2.a;
      }
      // the arrays should be the same.
      for (int i=0; i<M; ++i) System.out.print( a[i] );  System.out.println();
      for (int i=0; i<M; ++i) System.out.print( b[i] );  System.out.println();
   }
}
```

What to turn in: Submit two files `ZOrep.java` and `FLrep.java`. Do not include the classes `BT` or `TreeAndRepresentation`. Testing will be automated. You will receive zero marks if you code does not compile. Your code will also be judged on its efficiency. If your algorithms run correctly in time linear in the number of nodes in the tree then you should receive full marks. You may assume that the maximum tree size is at most 100,000 nodes.

- Additional written questions:
  - What is the ZO sequence produced from the tree on page 396 if we look at it in a vertical mirror (i.e., flip it over on the page)?
  - In the code for `build` given in `ZOrep`, as a function of N, (a) how many times is `k` incremented? (b) how many times is `build()` called in total?
  - What is the FL sequence for the "ordered" forest for the two tree forest shown at the bottom of page 225?
  - Exercise 1.5.8 in the book, except, instead of giving a counter-example, give a one sentence explanation of what can go wrong.
  - Exercise 1.5.9 from the book.