1. Consider the following FLFL code:

   ```flfl
   let a = 3
   let f = (proc x (:= x (+ 1 x)))
   (begin
     (call f (begin (:= a (* 2 a)) a) )
     a)
   ```

   (a) With the translation $T$ from FLFL to FLK! defined as in class, what would this expression evaluate to?

   (b) How would you modify your definition of $T$ so that the expression above evaluates to (the more intuitive value) 7? For simplicity you can assume that a `begin` construct has only two subexpressions, as in (begin $E_1$ $E_2$).

   Note that this issue would not show up in the code fragment discussed in class, because there, the (second) parameter which was bound to the argument involving the `begin` was only read by the procedure.

2. We discussed in class the syntactic translator $T$ which inputs an expression in a Fortran/Java-like language that uses implicit references (locations) and outputs an FLK! program that uses explicit cells and calls to `new`, `read` and `write`.

   We wish to extend this translation to handle arrays. As discussed on 2002-08-30, we can use an $AEnv$ to hold array-related bindings. But this means our “translation” is not quite syntactic, because it involves changes to runtime data structures, and so we might as well write a direct denotational semantics ($E$) for the source (array-based) language.

   (a) Design $E$ for FLFL so that the following program finishes with $u = (5, 7, 4)$ and $v = (3, 9)$.

   ```flfl
   letarray u[3], v[2]
   (begin
     (:= v[0] 3) (:= v[1] 8)
     let p = (proc x (begin (:= x[1] 7) (:= x v) (:= x[1] 9)) )
     (call p u) )
   ```

   (b) How would your design of $E$ need to change if the desired states of the arrays after execution of the above program are $u = (3, 9)$ and $v = (3, 8)$?

   (c) In yet another (not very useful) interpretation, the arrays may remain unchanged after they are initialized. Write down $E$ corresponding to this policy.
Based on the questions above, complete the following Java code in various ways and figure out that $E$ closest in spirit to the Java interpreter.

```java
public class Array {
    static int vb[];
    static {
        vb = new int[3];
    }
    static void mangle(int x[]) {
        // complete this procedure in various ways
    }
    static public final void main(String argv[]) {
        int va[] = new int[3];
        va[0] = 100; va[1] = 101; va[2] = 102;
        mangle(va);
        print(va);
        print(vb);
    }
    static void print(int y[]) {
        for ( int yi = 0; yi < y.length; yi++ ) {
            System.out.print(y[yi] + " ");
        }
        System.out.println();
    }
}
```