Answers to prelim 3, CS100J, Fall 207

1.
```
/** = n reduced to a single digit by repeatedly
      adding its digits together.
      Precondition: n > 0. */
public static int reduce(int n) {
   if (n < 10)
      return n;
   return reduce(n/10 + n%10);
}
```

2.
```
// double arrays a and b have the same length.
double[] c= new double[a.length];

int k= c.length;
// Inv: c[k..c.length-1] contains elementwise
//      averages of a[k..] and b[k..].
while (k != 0) {
     c[k-1]= (a[k-1] + b[k-1]) / 2.0;
     k= k - 1;
}
// Post: c[0..] contains elementwise averages of
//       a[0..] and b[0..].
```

3.
```
/** = a two-dimensional array that contains n+1
      rows, where each row k contains in its
      first k+1 elements row k of Pascal's
      triangle. */
public static int[][] pascal(int n){
   int[][]p= new int[n+1][n+1];

   // inv: p[0..r-1] contains the first r rows
          of Pascal's trangle
   for (int r= 0; r <= n; r= r+1) {
      //Store row r of Pascal's triangle in row p[r].
      p[r][0]= 1;
      for (int k= 1; k < r; k= k+1) {
         p[r][k]= p[r-1][k-1] + p[r-1][k];
      }
      p[r][r]= 1;
   }
   return p;
}
```

4a.
```
/** An instance is an integer in mod "modulus"
    arithmetic. */
public class Mod {
   private int m;        // The modulus. m > 1
   private int k;        // The integer. 0 ≤ k < m

   /** Constructor: integer k in mod m arithmetic.
       Precondition: m > 1 and k ≥ 0.*/
   public Mod(int k, int m) {
      this.m= m;
      this.k= k % m;
   }

/** If  this object and r do not have the same
    modulus, return null; otherwise, return an object
   that contain the sum of the two mod-m integers
   represented by this object and r. */
public Mod add(Mod r) {
     if (m != r.m)
        return null;
     return new Mod(k + r.k, m);
}

/** = "ob is a non-null Mod object with the same
        modulus and value as this one". */
public boolean equals(Object ob) {
     if (!(ob instanceof Mod))
        return false;
     Mod mob= (Mod)ob;
     return k == mob.k  && m == mob.m;
}
```

4b. Override a method m(…) inherited from a superclass by redefining it in the subclass. Call the overridden method using **super**.m(…).

4c. Instance variable or field: in a class. Class variable or static variable: in a class. Parameter: in the header of a method definition. Local variable: in a method body.

4d. Yes. Because of the principal that inherited fields should be initialized first. If such a call is missing, the call **super**(); is used.

5.
```
/** = the integer k that satisfies
           d[p..k] <= w < d[k+1..q-1].
        Precondition: b is in dictionary order. */
  public static int bSearch(String[] d,
                            int p, int q, String w) {
     int k= p-1;
     int t= q;

     // inv: d[p..k] <= w < d[t..q-1]
     while (k+1 < t) {
        int e= (k + t) / 2;
        if (comesBefore(d[e], w) <= 0) k= e;
        else t= e;
     }

     return k;
  }
```