Sample answers, CS1110 Prelim 3, Fall 2008 David Gries djg17

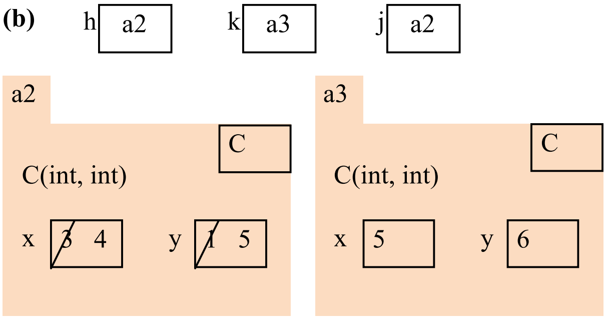
Q1.

```
/** =  Change m into its transpose.
        Precondition: m is not null and is square*/
public static void transpose(int[][] m) {
   // inv: Rows 0..r-1 have been transposed
   for (int r= 0; r != m.length; r= r+1) {
      // Swap m[r][r+1..] with m[r+1..][r]
      for (int c= r+1; c != m.length; c= c+1) {
         // Swap m[r][c] with m[c][r]
         int temp= m[r][c];
         m[r][c]= m[c][r];
         m[c][r]= temp;
      }
   }
}
```

Q2. (a)

```
int[][] b= new int[][] {
          {1,3,6,10},
          {2,5,9,13},
          {4,8,12,15},
          {7,1,14,16}};
```

(b)



(c)

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

Q3. We give the complete class, not only what you had to write.

```
/** An instance is a rational number */
public class Rational  {
   /** Class inv: the rational number is num / den
        Restrictions on fields:
           1. den > 0
           2. if num = 0, then den = 1
           3. num / den is in lowest possible terms.
        E.g. instead of 20/5 or –5/10, these numbers
        are stored as 4/1 and –1/2. */

   private int num;
   private int den;

   /** Constructor: instance with rational num. n / d.
        Precondition: d != 0 */
   public Rational(int n, int d) {
     num= n; den= d;
     if (den < 0) {
        num= -num;
        den= -den;
     }
     reduce();
   }

   /** = the numerator */
   public int getNumerator()
      { return num; }

   /** = the denominator */
   public int getDenominator()
      { return den; }

   /** = repr. of this rational num. Form: num/den. */
   public String toString() {
      return num + "/" + den;
   }

   /** = "r is of class Rational and has the same
          value as this Rational number”. */
   public boolean equals(Object r) {
     if (!(r instanceof Rational))
        return false;
     Rational rat= (Rational) r;
     return num == rat.num  &&  den == rat.den;
   }

   /** Reduce this rational number to the lowest
possible terms, e.g. 8/24 becomes 1/3. */
   public void reduce(){
     if (num == 0) {
        den= 1;
        return;
     }
     // num != 0, den > 0
     int gcd= gcd(Math.abs(num), den);
     num= num/gcd;
     den= den/gcd;
   }

   /** = greatest common divisor of x and y.
          Precondition: x > 0, y > 0 */
   public static int gcd(int x, int y) {
     int a= x;
     int b= y;
     // invariant: gcd(x,y) = gcd(a,b)
     while (a != b) {
        if (a > b) a= a-b;
        else b= b-a;
     }
     return a;
   }

}
```

```
Q4. /** Remove all Rational elements from v[h..]
        that have the same value as r.

          Precondition: 0 <= h < v.size(). */
public static void removeRationalEquals
                       (Vector v, int h, Rational r) {
   int k= h;
   // inv: No Rational element in v[h..k-1] equals r */
   while (k < v.size()) {
       if (r.equals(v.get(k))) {
           v.remove(k);
       } else {
          k= k+1;
       }
   }
  }

/** Remove all duplicate Rational numbers from v
     (so that each appears in it only once). */
public static void removeRationalDups(Vector v) {
   // inv: No Rational instance in v[0..h-1] has a
   //        duplicate in v.
   for (int h= 0; h != v.size(); h= h+1) {
       if (v.get(h) instanceof Rational) {
          Rational r= (Rational)(v.get(h));
          removeRationalEquals(v, h+1, r);
       }
   }
}
```

```
Q5. (a) /** = an integer j that satisfies
                  b[p..j] <= x < b[j+1..q-1]
        Precondition: b[p..q-1] is sorted */
public static int bsearch(int[] b, int x, int p, int q)

(b) int j= p-1;
    int k= q;
    // invariant: b[p..j] <= x < b[k..q-1]
    while (j+1 != k) {
        int e= (j+k)/2;
        if (b[e] <= x) j= e;
        else k= e;
    }
    return j;
```