Question 1 (24 points)

a) (4 points)

```
FiniteStringBag represents a bag (or multiset) with max size ==
items.length. If size==0, this FiniteStringBag represents the empty
bag { }, otherwise the strings in items[0..size-1] represent the
FiniteStringBag{items[0], ..., items[size-1]} with size <=
items.length.
```

The part about max size == items.length is redundant if they say size <= items.length. If they leave off the part about size == 0, give full points, it's redundant if the rest is there. It's OK to use capacity() instead of items.length. If they use the variable `capacity` instead, deduct 0.5 points. Capacity isn't a field in the object. They must include something about the fixed size, if not deduct a point. They can use the capacity() method.

b) (5 points)

Should check for nulls and check size is <= items.length (capacity). Deduct a point if they use capacity as a variable. They can use the capacity() method.

```
private void checkRep() throws RuntimeException {
     if (size < 0 || size > items.length)
         throw new RuntimeException("size out of bounds");
     for (int k = 0; k < size; k++)
         if (items[k] == null)
          throw new RuntimeException("null item");
    }
}
```

c) (9 points) (3 points each)

There are, of course, many possible tests. Here are some. Give points for reasonable attempts. They need 3 different tests.

(a)

```
Initialize bag b to { "a", "b", "c" }
b.deleteLongStrings(3)
Verify that b contains {"a", "b", "c"}
```

(b)

```
Initialize bag b to {
"xyzzy", "", "ab", "abcd", "abcdefg"}
b.deleteLongStrings(4)

Verify that b contains {"", "ab", "abcd"}
```

(c)

```
Initialize bag b to the empty bag { }
b.deleteLongStrings(1)
Verify that b is still the empty bag { }
```

(d)

```
Initialize bag b to { "abcd", "pqrstuv", "wxyz"
}b.deleteLongStrings(3)
Verify that b is the empty bag { }
```

(e)

```
Initialize bag b to { "abc", "abc", "abc" }
Verify that b contains 3 copies of "abc"
```

d) (2 points) No. All instance variables are private and the only data that is shared by the client code are references to immutable String objects, which the client cannot change.

Give 1 point if they just say No without an explanation.

e) (4 points, 2 points each)

It returns the whole array, but only 0…size-1 are valid entries.

Returning `items` is a rep exposure.

Question 2. (20 pts, 2pts each)

a) False
b) false
c) False
d) True/false (There was some confusion during the exam about what "should" meant – whether it was a good idea or meant "should always". Give 2 points.)
e) False
f) False
g) False
h) False (see Testing2.pdf slides 25, 26, also Duration5.java)
i) True (ADTReasoning.pdf slide 46)
j) false

Question 3 (12 points, 2 points each

```
a) AA
b) BA
c) BB
d) BA
e) CA
f) CA
```

Question 4 (14 points)

a) (2 points) Factory. Also accept Factory Object or prototype.
b) Constructors can't return or create subclass objects, or return a value. Can't return null. (accept any of these)
c) Yes, RatNum is immutable so it could be interned and programs with many RatNums could possibly use less memory.
d) no, graphs are not immutable.
e) singleton
f)) (2 points, 1 point for each condition)
Both of these must hold:
T1 is the same as, or is a subtype of T, can also say return type are covariant
S1 is the same as, or is a supertype of S, can also say arguments are contravariant
They don't have to say same, subtype or supertype is enough.
g) (2 points, 1 point each)

```
static void removeDuplicates(Collection<? extends T> src,
                             Collection <? super T> dst);
```

Question 5 (14 points, 2 points each)
a) ERROR
b) OK
c) ERROR
d) OK
e) ERROR
f) OK
g) OK

Question 6 (8 points, 2 points each)
NOT VALID
NOT VALID
NOT VALID
VALID

Question 7 (8 points, 2 points each) The don't have to list the cases for b and c, but if they do get some right, give partial credit)

a) 16

b) 3 {a true; a false and b true; a false, b false, c true, and d true}

c) 5 {a true; a false, b true; a false, b false, c true, d true; a false, b false, c true, d false; a false ,b false, c false}

d) a