Answer to 1(b):

Fragment 1:
output is

```
 Apples: ****
Bananas: **
Oranges: *****
```

Fragment 2:
output is

```
Chocolate: !!!!!!!!!!
   Onions: !!!
unlabeled:
unlabeled:
```

Fragment 3:
fails due to incorrect call to constructor: the required parameter numCategories is missing

Fragment 4:
fails due to an array out of bounds error: number of categories are 2, so therefore the indexes are 0 and 1, but an attempt is made to access element 2 of the arrays

Answer to Q2:

```
class CumulativeHistogram extends Histogram {

  public CumulativeHistogram(int numCategories){
     super(numCategories);
  }

  public String toString() {
    String result = "";
    int sum=0;
    for (int index=0; index<categories.length; index++) {
      result+=categories[index]+": ";
      sum+=frequencies[ index ];
      result+=repeatSymbol(sum);
      result+="\n";
    }
    return result;
  }
}
```

Answer to Q4b:

Fragment 1:
```
a sportscar drives
```

Fragment 2: fails because it tries to create an object from an abstract class

Fragment 3:
```
a bike is ridden
```

Fragment 4:
```
a bike is ridden
```

Fragement 5:
```
sporty's top speed is 210.0
```

Fragment 6: fails because reference type Car does not have field topSpeed

Fragment 7:
```
sporty3's top speed is 250.0
```

Fragment 8:
```
a minivan drives
```

Fragment 9: fails due to van3 being null when attempt to call drive() method is made

Fragment 10:
```
a minivan drives
van4's make = Toyota
a minivan drives
```

Answer to Q5:

A good set of test cases is

“Normal” cases, where there are no incorrect parameters, e.g.
```
randomPattern(4,5);
randomPattern(6,2);
randomPattern(100,100);
```
etc

“Error” cases where your would expect to get an error.
Note that in this example, either one or both of the parameters may be wrong therefore you should test each combination
```
// First parameter correct, second one wrong
randomPattern(23,-5);
randomPattern(3,-100);
// First parameter incorrect, second one correct
randomPattern(-22,12);
randomPattern(-4,47);
// Both parameters wrong
```

```
randomPattern(-55,-292);
randomPattern(-10,-8);
```

“Boundary cases” which sit at the border between correct/error cases, e.g.

```
randomPattern(0,0);
randomPattern(-1,0);
randomPattern(0,-1);
randomPattern(-1,-1);
```

Answer to Q6:

In the first code fragment, only one object is created but there are two references to the object. In the second code fragment, two objects are created (the second being a copy of the first) and each has its own reference.

Answer to Q7:

(a) super(x,y) calls MyVector’s superclass’ constructor, passing it parameters x and y. No fields x and y are needed in MyVector because they are inherited from Vector.

(b) “this.q” refers specifically to the field q of the current object; “q” may refer either to a field or a local variable

(c) Yes it will run

(d) The dist() method is inherited from the PVector class and it is this method that is called

(e) It will work because MyVector inherits from PVector, and therefore by polymorphism the object “yours” can pretend to be a PVector even though its class is MyVector