1a 22 26
1b I D
1c I E
1d infinite recursion
1e 2

2a
```
size:
  return 1;

harvest:
  bag.add(this);
```

2b
```
size:
int sum = 0;
for (final Tree t: getChildren()) // final optional; OK to use
children directly
    sum += t.size();
return sum;

harvest:
for (final Tree t: getChildren()) // final optional; OK to use
children directly
    t.harvest(bag);
```
2c
```
size:
return getMultiplier() * getChild().size(); // OK to use howmany and
child directly

grow:
return new Cluster(factor * getMultiplier(), getChild()); // ditto
```

or

```
return new Cluster(factor, this);

harvest:

for (int i = 0; i < getMultiplier(); i ++) // OK to use howmany
directly
    getChild().harvest(bag); // OK to use child directly
```

2d

```
return new Cluster(factor, this); // in both cases
```

2e

```
final Branch trunk = new Branch(
```

```
    new Branch(
        new Mango(),
        new Mango()
    ),
    new Branch(
        new Cluster(2, new Mango()),
        new Cluster(1, new Peach())
    )
);
```

2f 5 – only leaf Fruit are counted

2g m m m m p

2h 15 = 3 ∗ 5

2i Composite and Decorator – Branch is a Composite, and Cluster is a Decorator

2j true – see item 2e

2k LSP (Clusters with multipliers and Branches with multiple items both fully implement Tree)

2l true

2m true

2n false – Visitor and all classes that implement it must change

2o false – Shape does not have to be modified, since new capabilities are Visitors!

3a true

3b true false / false false

3c true true / true true

3d false / false