

CS100J Spring 2007. Answers to final

```
1. s= - cumprod(-1 + zeros(1, n)); % the signs
   num= s .* cumprod(5 + zeros(1,n));
   den= 1:n;
   value= cumsum(num ./ den);

// postcondition: (digit sum of n) = c;
2.   int m= n;
int c= 0;
//inv: (digit sum of n) = c + (digit sum of m) and
    0 <= c < 10
while (m > 0) {
    c= c + m % 10;
    if (c >= 10)
        c= c/10 + c%10;

    m= m / 10;
```

```
3 }/** See final for spec */
public static String[] histo(int[] scores) {
String[] s= new String[scores.length];
//invariant: s[0..k-1] has their final values
for (int j= 0; j < s.length; j= j+1) {
    // Create s[k]
    s[j]= (j < 10 ? "0" : "") + j + " ";
    for (int n= 0; n < scores[j]; n= n+1) {
        s[j]= s[j] + "**";
    }
}

return s;
}
```

```
4. CS100 Blues - Gries : 9.0
al (name of an object)
Help! - Student : 10.0
Length of sound: 59.0
The author is: Gries
ERROR
The songs have the same author?false
The songs have the same author?true
```

5. Note that in this case we put the base case last simply because testing for the base case doesn't have to be done then.

```
/** = value of expression e. */
public static int eval(Exp e) {
    if (e.n.equals("+"))
        return eval(e.left) + eval(e.right);

    if (e.n.equals("-"))
        return eval(e.left) - eval(e.right);

    if (e.n.equals("*"))
        return eval(e.left) * eval(e.right);

    return Integer.parseInt(e.n);
}
```

6. (a) body of constructor:

```
super(n,p,s);
canLaugh= b;
```

body of equals:

```
if (!(ob instanceof BabyDoll))
```

```
    return false;
```

```
BabyDoll bd= (BabyDoll) ob;
```

return

```
ob.getName().equals(getName()) &&
ob.getPrice() == getPrice() &&
ob.getSeller().equals(getSeller()) &&
ob.canLaugh == canLaugh;
```

(b) Apparent class: Doll. Real class: BabyDoll.

```
(c) public class BarbieDoll extends Doll {
    private String hairColor; // hair color
    private String style; // Style of clothes
```

```
/** Constructor: a BarbieDoll with name n, price p,
    seller s, hair color hc, and clothing style cs */
```

```
public BarbieDoll(String n, double p, String s,
    String hc, String cs) {
```

```
    super(n, p, s);
    hairColor= hc;
    style= cs;
```

```
}
```

```
/** = hair color */
```

```
public String getHairColor()
    { return hairColor; }
```

```
/** = clothing style */
```

```
public String getStyle()
    { return style; }
```

```
/** String representation of this BarbieDoll*/
```

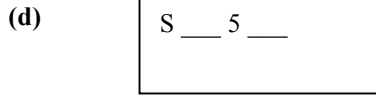
```
public String toString(){
    return super.toString() +
        ",Hair color: " + hairColor +
        ",Style: " + getStyle();
}
```

```
}
```

7. (a) If `b` contains the name of an object of class `Object`, with no subclasses, then `b.equals(ob)` is the same as `b == ob`.

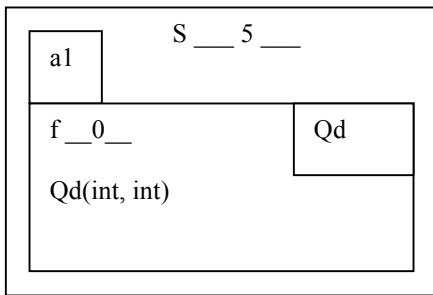
(b) `super()`;

(c) `this` evaluates to the name of the object in which it occurs.

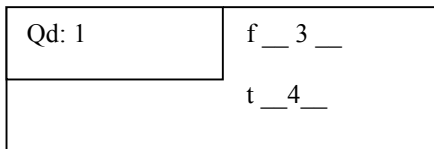


Steps in evaluating `new Qd(3, 4)`:

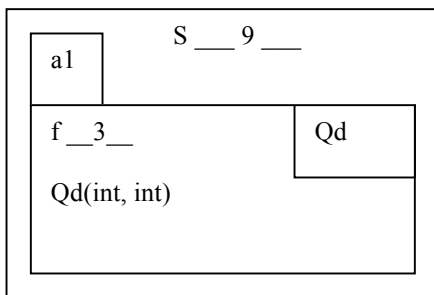
1. Draw an object of class `Qs` (and put it in the file drawer)



2. Execute the constructor call `Qd(3, 4)`. Here is the frame for the call:



and after execution, the object and the file drawer look like this:



3. The value of the new expression is: the name `a1`.

8. **Algorithm Partition.** We give assertions as formulas; you can translate them easily into diagrams.

```
/** Given a value x in b[p], partition b[p..q-1] and store a
    value in local variable j so that
    b[p..j-1] <= x = b[j] <= b[j+1..q-1].
    Then return j. */
```

```
public static int partition(int[] b, int p, int q) {
    int j= p;
    int k= q;
    // inv: b[p..j-1] <= x = b[j] <= b[k..q-1]
    while (j+1 != k) {
        if (b[j+1] <= b[j]) {
            Swap b[j] and b[j+1]; j= j+1;
        } else {
            k= k-1; Swap b[k] and b[j+1]
        }
    }
    return j;
}
```