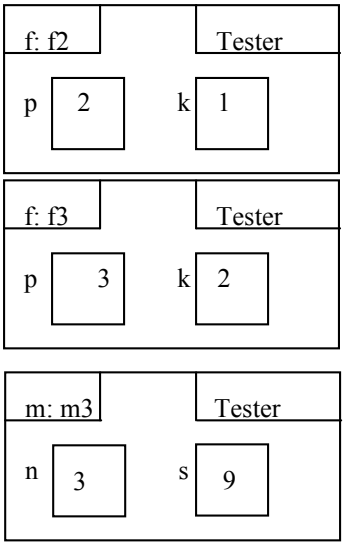
Question 1. (a)



(b) Variables p and k are created when the frame for a call on function `Tester.f` is created. Variable `s` is created when the frame for a call on `Tester.m` is created.

Question 2. Showing only the three functions.

```
/** = the previous president */
public Pres getPrevious() {
      return prev;
}

/** = number of presidents before this one */
public int before() {
      if (prev == null)
         return 0;
      return 1 + prev.before();
   }

/** = "p is a Pres and has the same starting and ending
       year and the same name as this President" */
public boolean equals(Object p) {
      if (!(p instanceof Pres))
         return false;

      Pres pp= (Pres) p;
      return sYear == pp.sYear &&
             eYear == pp.eYear &&
             name.equals(pp.name);
}
```

Question 3 (a). Showing only the two functions.

```
/** = number of Democratic pres, before this one */
public int beforeInSameParty() {
      if (prevDem == null)
         return 0;
      return 1 + prevDem.beforeInSameParty();

/** = number of Republican pres. before this one */
public int beforeInSameParty() {
      if (prevRep == null)
         return 0;
      return 1 + prevRep.beforeInSameParty();
   }
```

(b) 1 is the correct answer. The apparent class of p is Pres, and function beforeInSameParty is not declared in or inherited by Pres.

(c) If you answered 1 for part (b), you should put something like this function in class Pres:

```
/** = number of pres before this one in the same party
       (=  0 if this function is called, instead of a one in
        subclass)*/
public int beforeInSameParty() {
      return 0;
}
```

If you answered 2 for part (b), you can still get full credit on this part (c) by writing something like this:

```
p instanceof Dem ?
    ((Dem)p).beforeInSameParty():
    (p instanceof Rep ?
       ((Rep)p).beforeInSameParty(): 0)
```

Question 4.

(a) 1. 14; 2. 20; 3. 8; 4. true; 5. false; 6. true 7. true; 8. 8.

(b) 1. Apparent is Two and real is Two.
2. Apparent is Two and real is One.

Question 5.

```
/** = the chars of s1 and s2, in alpha. order.
       Precondition: the chars of s1 are in alpha order
       and the chars of s2 are in alpha. order.   */
public static String merge(String s1, String s2) {
   if (s1.length() == 0)
      return s2;

   if (s2.length() == 0)
      return s1;

   // s1 and s2 each have at least 1 character
   if (s1.charAt(0) <= s2.charAt(0))
      return s1.charAt(0) + merge(s1.substring(1), s2);

   // s2[0] <= s1[0]
   return s2.charAt(0) + merge(s1, s2.substring(1));
}
```