Problem Set 2

Question One: A Teddy Bear Picnic

This question involves a game with teddy bears. The game starts when I give you some bears. You can then give back some bears, but you must follow these rules (where n is the number of bears that you have):

1. If n is even, then you may give back exactly n/2 bears.
2. If n is divisible by 3 or 4, then you may multiply the last two digits of n and give back this many bears. (By the way, the last digit of n is n%10, and the next-to-last digit is ((n%100)/10).
3. If n is divisible by 5, then you may give back exactly 42 bears.

The goal of the game is to end up with EXACTLY 42 bears.

For example, suppose that you start with 250 bears. Then you could make these moves:

- --Start with 250 bears.
- --Since 250 is divisible by 5, you may return 42 of the bears, leaving you with 208 bears.
- --Since 208 is even, you may return half of the bears, leaving you with 104 bears.
- --Since 104 is even, you may return half of the bears, leaving you with 52 bears.
- --Since 52 is divisible by 4, you may multiply the last two digits (resulting in 10) and return these 10 bears. This leaves you with 42 bears.
- --You have reached the goal!

Write a recursive method to meet this specification:

```
public static boolean bears(int n)
      // Postcondition: A true return value means that it is possible
      // to win the bear game by starting with n bears. A false return
      // value means that
      // it is not possible to win the bear game by starting with n
      // bears.

      // Examples:

      //   bear(250) is true (as shown above)
      //   bear(42) is true
      //   bear(84) is true
      //   bear(53) is false
      //   bear(41) is false
```

Hint: To test whether n is even, use the expression ((n % 2) == 0).

Question Two: Triangle Pattern

```
public static void triangle(int m, int n)

// Precondition: m <= n
// Postcondition: The method has printed a pattern of 2*(n-m+1) lines

// to the standard output. The first line contains m asterisks, the next
// line contains m+1 asterisks, and so on up to a line with n asterisks.
// Then the pattern is repeated backwards, going n back down to m.

/* Example output:

triangle(3, 5) will print this:

***
****
*****
*****
****
***

*/
```

Hint: Only one of the arguments changes in the recursive call. Which one?