**Question one**

Write a recursive method with two int parameters, m and n. The precondition requires 0 <= m and m <= n. The method prints a line of m asterisks, then a line of m+1 asterisk, and so on up to a line of n asterisks. Then the same pattern is repeated backward: a line of n asterisks, then n-1, and so on down to n. The only loop allowed in your implementation is a loop to print a line of m asterisks. You may have two copies of this loop in different places of the implementation.

**Question two**

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***

**Question 3**

Suppose that you want to implement the PriorityQueue so that insertions occur in constant time, but getFront requires linear time. You will use these class definitions, where the data entering the PriorityQueue is a String and the priorities are ints.

public class PriorityQueue

{

// A PriorityNode is a node from a linked list of strings, with

// methods for getString, setString, getPriority, setPriority,

// getLink, and setLink.

private PriorityNode head;

public void insert(String entry, int priority)...

public String getFront( )...

...

}

1. ***Write ONE sentence to describe how the insert method will work (with constant time).***
2. ***Then implement the getFront method (which will have linear worst-case time). In your implementation, you DO NOT have to worry about items with equal priority*** ***(they may come out of the prioirty queue however you like, without necessarily having FIFO behavior). To remove the head node of a linked list, use the assignment:***

head = head.getLink( );

**question 4**

Write a method with one positive int parameter called n. The method will write 2^n-1 integers (where ^ is the exponentiation operation). Here are the patterns of output for various values of n:   
n=1: Output is: 1   
n=2: Output is: 1 2 1   
n=3: Output is: 1 2 1 3 1 2 1   
n=4: Output is: 1 2 1 3 1 2 1 4 1 2 1 3 1 2 1   
And so on. Note that the output for n always consists of the output for n-1, followed by n itself, followed by a second copy of the output for n-1.