Description: The Tower of Hanoi (also called the Tower of Brahma or Lucas Tower) is a mathematical game or puzzle. It consists of three rods, and a number of disks of different sizes which can slide onto any rod. The puzzle starts with the disks in a neat stack in ascending order of size on one rod, the smallest at the top, thus making a conical shape, see Figure 1.

![Figure 1: A visual example of the Towers of Hanoi problem](image)

The objective of the puzzle is to move the entire stack to another rod, obeying the following simple rules:

1. Only one disk can be moved at a time.

2. Each move consists of taking the upper disk from one of the stacks and placing it on top of another stack i.e. a disk can only be moved if it is the uppermost disk on a stack.

3. No disk may be placed on top of a smaller disk.

With three disks, the puzzle can be solved in seven moves. The minimum number of moves required to solve a Tower of Hanoi puzzle is $2^n - 1$, where $n$ is the number of disks. Your project is to code an implementation of the solution of the Tower of Hanoi that uses the optimal number of moves. For this project you will be implementing and using the stack abstract data structure.

Part 1 - Stack ADT (60 points) In this part of the project you will be creating a custom stack ADT in two different ways. The first way is using an array as the internal data structure. This stack ADT class is called “ArrayStack”. The second way is using a singly linked list implementation. This stack ADT class is called “ListStack”.

You will be implementing the exact same methods for both classes. The methods for both ArrayStack and ListStack include,

public ArrayStack() and public ListStack() - Fill in the constructor
**public void push(T num)** - Push an item to the top of the stack. In the array-based implementation, if the stack becomes larger than the array, double the array size and copy over the values to a larger array. Then push the item to the top of the stack. (linked lists dynamically grow)

**public T pop() throws StackUnderflowException**- Pop off the top element of the stack and return the item. If the stack is empty, then throw a StackUnderflowException.

**public T top() throws StackUnderflowException**- Peek at the top of the stack and return the item. If the stack is empty, then throw a StackUnderflowException.

**public T get(int ind)**- Get the item at index ind. Return null if the ind is out of bounds.

**Part 2 - Tower of Hanoi (recursive and iterative solution) (60 points)** In this part of the assignment, you will code two solutions to the Tower of Hanoi problem. For this implementation you may use either the ArrayStack or the ListStack. It does not matter which you choose, but be aware that both stacks will be tested for full functionality. The recursive solution is described and can be found in the book, chapter 4.3. However, significant modifications will be necessary to fit the book code into the provided template.

The second part of this assignment is to code an iterative solution to this problem. The solution can be described as the following:

1. Move odd-numbered disks left and even-numbered disks right.
2. Do not move the same disk twice in succession (iterate between the smallest disk and next smallest disk)
3. Do not place a larger disk on top of a smaller one.

Provide your implementations in the methods named,

**public static void recursiveSolution(int n, CStack startPeg, CStack auxPeg, CStack endPeg)**

and

**public static void interativeSolution(int n).**

**Part 3 - PDF write up (30 points)** Create a 2-3 page write up that explains the interesting parts of your program. Include how you handled the two types of Stack ADTs and especially how you went about coding the recursive and iterative solutions.

**Deliverables:** Submit on blackboard.