/\*

Chris Clemmons

CSC310

Wooky Kim

3/20/19

\*/

package hw5;

//import stack class and scanner

import java.util.Stack;

import java.util.Scanner;

public class Hw5 {

// method to print preOrder traversal

public static void preOrder(int[] data, int p){

if(data[p] != 0){//checks if data is empty

if(p < data.length){//checks if p is less than data length

System.out.print(data[p] + " ");// prints out data at point p

if((p\*2 + 1) < data.length){//checks p\*2+1 against length of data

if(data[p] != 0) {

preOrder(data, (p\*2 + 1));// calls method recusrively

}

}

if((p\*2+2) < data.length){// checks p\*2+2 against length of data

if(data[p] != 0){

preOrder(data, (p\*2 + 2));//calls method recursively

}

}

}

}

}

//method to print in Order traversal

public static void inOrder(int[] data, int p){

if(data[p] != 0){//checks if data is empty

if(p<data.length){//checks if p is less than data length

if((p\*2 + 1) < data.length){ //checks p\*2+1 against data length

if(data[p] != 0)

inOrder(data, (p\*2 + 1));//calls method recursively

}

if(data[p] != 0){//checks if there is data at p

System.out.print(data[p] + " ");

}

if((p\*2+2) < data.length){//checks p\*2+2 against data length

if(data[p] != 0)

inOrder(data, (p\*2 + 2));//calls method recursively

}

}

}

}

//method to solve hanoi tower problem

static void hanoi(int n, Stack A, Stack B, Stack C, char a, char b, char c){

if (n > 0){//checks if n is >0

hanoi(n-1, A, C, B, a, c,b); //calls hanoi recursively

int temp = (int) A.pop(); //sets up temp var

C.push(temp);//push temp var to c

//prints out move

System.out.println("Move disk "+temp+ " from peg "+ a+" to peg "+ c);

hanoi(n-1, B, A, C, b, a, c); //calls hanoi recursively

}

}

public static void main(String[] args) {

//sets up scanner

Scanner in = new Scanner(System.in);

//get input from user

System.out.println("Insert number of disks: ");

int n = in.nextInt(); //grab n from input

//set up stack A

Stack A= new Stack();

for(int i =0; i<=n; i++){//populate stack A

A.push(i);

}

Stack B= new Stack(); //set up stack B &C

Stack C= new Stack();

hanoi(n, A, B, C, 'A', 'B', 'C'); //calls hanoi

//sets up binary tree

int[] t = {1, 0, 2, 0, 0, 3};

System.out.println("Pre-Order: ");

preOrder(t, 0); //calls preOrder

System.out.println();

System.out.println("In-Order: ");

inOrder(t, 0); //calls inOrder

System.out.println();

}

}