```
/*
 * Andrew Kalathra
 * 9/17/23
 * The purpose of this program is to (theoretically)
 * restore files using inputed FAT and
 * directory tables
 */
import java.util.Scanner;
import java.util.Stack;
import java.util.ArrayList;

public class FileRecovery {

	public static void main(String[] args) {
		Scanner input = new Scanner(System.in);

		// root table
		Stack<Integer> rootTable = new Stack<Integer>();
		System.out.println("First cluster of each existing file in the root
table (separated by spaces): ");
		String hold = input.nextLine();
		String[] holdArr = hold.split(" ");
		for (int i = 0; i < holdArr.length; i++) {
			rootTable.add(Integer.valueOf(holdArr[i]));
		}
		// checker
		/*
		 * while(!rootTable.empty()) { System.out.print(rootTable.pop() + "
"); }
		 */

		// FAT
		System.out.println("First row of FAT: ");
		hold = input.nextLine();
		holdArr = hold.split(" ");
		// System.out.println(holdArr.length);
		int[][] FAT = new int[2][holdArr.length];
		for (int i = 0; i < holdArr.length; i++) {
			FAT[0][i] = Integer.valueOf(holdArr[i]);
		}
		// System.out.println();
		System.out.println("Second row of FAT: ");
		hold = input.nextLine();
		holdArr = hold.split(" ");
		for (int i = 0; i < holdArr.length; i++) {
			FAT[1][i] = Integer.valueOf(holdArr[i]);
		}
		// checker
		/*
```

```
                 * for (int i = 0; i < 2; i++) { for (int j = 0; j < FAT[1].length;
j++) {
                 * System.out.print(FAT[i][j] + " "); } System.out.println(); }
                 */
                System.out.println();
                input.close();

                // find missing positions
                int[][] misDocFAT = missingPos(FAT, rootTable);
                ArrayList<Integer> newRow1 = new ArrayList<Integer>();
                ArrayList<Integer> newRow2 = new ArrayList<Integer>();
                for (int j = 0; j < misDocFAT[1].length; j++) {
                        newRow1.add(misDocFAT[0][j]);
                        newRow2.add(misDocFAT[1][j]);
                }
                // checker
                /*      System.out.println(newRow1);
                 *  System.out.println(newRow2);
                 */
                Stack<Integer> answer = new Stack<Integer>();
                int eliminate = 0;
                int index = 0;
                while (newRow2.contains(eliminate)) {
                        index = newRow2.indexOf(eliminate);
                        eliminate = newRow1.get(index);
                        answer.add(eliminate);
                        //checker
                        //System.out.println(eliminate);
                }
                System.out.println(answer);

        }

        public static int[][] missingPos(int[][] FAT, Stack<Integer> root) {

                ArrayList<Integer> row1 = new ArrayList<Integer>();
                ArrayList<Integer> row2 = new ArrayList<Integer>();

                for (int j = 0; j < FAT[0].length; j++) {
                        row1.add(FAT[0][j]);
                        row2.add(FAT[1][j]);
                }
                // checkers
                /*
                 * System.out.println(row1); System.out.println(row2);
                 */
                int eliminate = root.pop();
                int index = 0;
                while (!root.empty()) {
                        if (eliminate == -1) {
```

```
                                eliminate = root.pop();
                        }
                        index = row1.indexOf(eliminate);
                        row1.remove(index);
                        // checker
                        // System.out.println("Eliminate:" + eliminate + " Index:" +
index);
                        if (row2.get(index) != 0) {
                                eliminate = row2.get(index);
                        } else {
                                eliminate = -1;
                        }
                        row2.remove(index);
                }
                // checker
                /*
                 * System.out.println(row1); System.out.println(row2); *
                 */

                int rowSize = row1.size();
                int[][] newFAT = new int[2][row1.size()];
                for (int j = 0; j < rowSize; j++) {
                        newFAT[0][j] = row1.get(j);
                        newFAT[1][j] = row2.get(j);
                }
                // checker
                /*
                 * System.out.println(newFAT.length + " " + newFAT[0].length);
for(int i=0; i<2;
                 * i++) { for(int j=0; j<rowSize; j++) {
System.out.print(newFAT[i][j] + " "); }
                 * System.out.println(); }
                 */
                return newFAT;
        }

}
```