```
Enter File Name:
C:\Users\Jamie\Desktop\AI Files\file9
1 : 9 11 13
2 : 4 7 15
3 : 2 5 14
4 : 1 8 12
5 : 3 6 10

----------------------------------------------------------------------
-
Enter File Name:
C:\Users\Jamie\Desktop\AI Files\file10
1 : 13 17
2 : 9 10
3 : 8 14
4 : 7 12
5 : 6 20
6 : 2 16
7 : 1 18
8 : 5 19
9 : 3 15
10 : 4 11
```

```java
import java.util.LinkedList;

//Solves Professors Problem
public class Professors {

    private class ClassList{
        private int[] list;                  //list of characters
        private int classesOffered;          //m*k

        //Constructor
        public ClassList(int m, int k){
            classesOffered = m*k;

            list = new int[classesOffered];

            for(int i = 0; i < list.length; i++)
                list[i] = -1;                //Start list with -1
        }

        //Copy Constructor
        public ClassList(ClassList other){
            this.classesOffered = other.classesOffered;

            this.list = new int[this.classesOffered];

            for(int i = 0; i < this.list.length; i++)
                this.list[i] = other.list[i];
        }

        public int checkNext(){
            for(int i = 0; i < list.length; i++)
                if (list[i] == -1)
                    return i;                //returns next index
            return list.length-1;                       //returns end
 of array
        }

        public void addNext(int professor){
            list[checkNext()] = professor;
        }

        public int getLastProfessor(){
            int next = checkNext();
            return list[next - 1];
        }

        public int[] getList() {
            return list;
        }

        public int getClassesOffered() {
            return classesOffered;
        }
    }
    private int classesPreferred;        //n
    private int classesTaught;          //k
    private int numProfessors;          //m
    private int[][] preferenceList;     //List of preferred classes

    //Constructor
    public Professors(int m, int k, int n, int[][] preferenceList ){
        classesPreferred = n;
        classesTaught = k;
        numProfessors = m;
        this.preferenceList = preferenceList;       //only copies reference
    }

    //Solve professor problem
    public void solve(){
        LinkedList<ClassList> list = new LinkedList<>(); //List of lists

        ClassList classList = new ClassList(numProfessors, classesTaught);
        list.addFirst(classList);
        while(!list.isEmpty()){             //While list is not empty
            classList = list.removeFirst(); //Remove first
            if(complete(classList)){        //check if complete
                display(classList);         //display list
                return;                     //stop
            }
            else{
                LinkedList<ClassList> children = generate(classList);  //child list
                if(children != null)                                   //Check if no children
                    for(int i = 0; i < children.size(); i++)          //Add children
                        list.addFirst(children.get(i));
            }
        }
        System.out.println("No Solution");  //If none in list, no solution
    }

    //Generates Children
    private LinkedList<ClassList> generate(ClassList parent){
        LinkedList<ClassList> children = new LinkedList<>();            //Children List
        LinkedList<Integer> possible = findPossible(parent.checkNext());  //Find possible professors
        for(int i = 0; i < possible.size(); i++){           //For all professors
            ClassList child = new ClassList(parent);        //Create Copy
            child.addNext(possible.get(i));     //Add Professor

            if(checkProfessors(child, possible.get(i)))      //Check if valid
                children.addFirst(child);   //Add if it is
        }
        return children;
    }

    //returns professors who want class
    private LinkedList<Integer> findPossible(int nextClass){
        nextClass ++;
        LinkedList<Integer> possible = new LinkedList<>();      // List of possible professors
        for(int i = 0; i < numProfessors; i++ )                 // iterate through profs
            for (int j = 0; j < classesPreferred; j++)           // iterate through their classes
                if (preferenceList[i][j] == nextClass)          //If they prefer that class
                    possible.addFirst(i);
        return possible;                                        // return list
    }

    //checks if professors teach more than possible classes
    private boolean checkProfessors(ClassList classList, int lastProfessor){
        //int lastProfessor = classList.getLastProfessor();       // Get last prof added
        int countOfClasses = 0;                                 // start count of classes they taught

        for(int i = 0; i < classList.getClassesOffered(); i++) {
            if (classList.getList()[i] == lastProfessor) {
                countOfClasses++;                              // count classes they are taught
            }
        }

        if(countOfClasses > classesTaught) {                      // check if valid amount
            return false;
        }
        return true;
    }

    //checks if board is complete
    private boolean complete(ClassList classList){
        for(int i = 0; i < classList.getClassesOffered(); i++)
            if(classList.getList()[i] < 0)
                return false;
        return true;
    }

    //display schedule
    private void display(ClassList classList){
        for(int i = 0; i < numProfessors; i++){
            System.out.print(i + 1 +" : ");
            for(int j = 0; j < classList.getClassesOffered(); j++)
            {
                //System.out.print(classList.getList()[j]+ " ");
                if (i == classList.getList()[j])
                    System.out.print((j+1) + " ");
            }
            System.out.println();
        }
    }
}

```

```java
import java.util.Scanner;
import java.io.*;

public class ProfessorsTester {
    public static void main(String[] args)throws IOException{
        Scanner keyIn = new Scanner(System.in);

        System.out.println("Enter File Name:");
        String fileName = keyIn.nextLine();

        File file = new File(fileName);
        Scanner sc = new Scanner(file);

        String firstLine[] = sc.nextLine().split(" ");

        int m = Integer.parseInt(firstLine[0]);

        int n = Integer.parseInt(firstLine[1]);

        int k = Integer.parseInt(firstLine[2]);


        sc.nextLine();
        int[][] preferenceList = new int[m][n];
        for(int i = 0; i < m; i++){
            String line[] = sc.nextLine().split(" : |\\s+");
            for(int j = 1; j <= n; j++)
                preferenceList[i][j-1] = Integer.parseInt(line[j]);
        }

        Professors s = new Professors(m,k,n,preferenceList);
        s.solve();
    }
}

```