WEEK#1 - ASSIGNMENT 3

MS Connect

Vijay Sekhar G

Objectives

- Get comfortable with .NET.
- Start thinking more carefully.
- Solve problems in C# using built in libraries and advanced concepts.

Reasonable

1. Communicating with colleagues about problem problems in English (or some other spoken language).
2. Discussing the assignment material with others in order to understand it better.
3. Helping a colleagues identify a bug in his or her code, as by viewing, compiling, or running his or her code, even on your own computer.
4. Incorporating snippets of code that you find online or elsewhere into your own code, provided that those snippets are not themselves solutions to assigned problems and that you cite the snippets' origins.
5. Sending or showing code that you've written to someone, possibly a colleagues, so that he or she might help you identify and fix a bug.

Deliverables

- Document your solutions in word file.
- Write proper comments for each line in your source code.
- Document the output of your program.
- Your program should address the problem, there should NOT be any deviations in output.

SECTION – A

① The A man arrives at a bus stop at 12:00. He remains there during 12:00-12:59. The bus stop is used by a number of bus routes. The man notes the times of arriving buses. The times when buses arrive are given.

- Buses on the same route arrive at regular intervals from 12:00 to 12:59 throughout the entire hour.
- Times are given in whole minutes from 0 to 59.
- Each bus route stops at least 2 times.
- The number of bus routes in the test examples will be <=17.
- Buses from different routes may arrive at the same time.
- Several bus routes can have the same time of first arrival and/or time interval. If two bus routes have the same starting time and interval, they are distinct and are both to be presented.

Find the schedule with the fewest number of bus routes that must stop at the bus stop to satisfy the input data. For each bus route, output the starting time and the interval.

Input Data

The input contains a number n (n<=300) telling how many arriving buses have been noted, followed by the arrival times in ascending order.

Example:

```
17
0 3 5 13 13 15 21 26 27 29 37 39 39 45 51 52 53
```

Output Data

Write a table to the OUTPUT.TXT file with one line for each bus route. Each line in the file gives the time of arrival for the first bus and the time interval in minutes. The order of the bus routes does not matter. If there are several solutions, only one is required.

Example:

```
0 13
3 12
5 8
```

②

```
|---|---|---|---|---|
| 1 | 1 | 3 | 5 | 1 |
|---|---|---|---|---|
| 3 | 3 | 2 | 0 | 3 |
|---|---|---|---|---|
| 3 | 0 | 3 | 2 | 3 |
|---|---|---|---|---|
| 1 | 4 | 0 | 3 | 3 |
|---|---|---|---|---|
| 3 | 3 | 3 | 1 | 1 |
|---|---|---|---|---|
```

Above figure shows a square. Each row, each column and the two diagonals can be read as a five digit prime number. The rows are read from left to right. The columns are read from top to bottom. Both diagonals are read from left to right. Using the data in the INPUT.TXT file, write a program that constructs such squares.

- The prime numbers must have the same digit sum (11 in the example).
- The digit in the top left-hand corner of the square is pre-determined (1 in the example).
- A prime number may be used more than once in the same square.
- If there are several solutions, all must be presented.
- A five digit prime number cannot begin with zeros, ie 00003 is NOT a five digit prime number.

Input Data

The program reads data from the INPUT.TXT file. First the digit sum of prime numbers and then the digit in the top left-hand corner of the square. The file contains two lines. There will always be a solution to the given test data. In our

Example

11
1

Output Data

In the OUTPUT.TXT file, write five lines for each solution found, where each line in turn consists of a five digit prime number. The above example has 3 solutions which means that the OUTPUT.TXT file contains the following (the empty lines are optional):

Example

11351
14033
30323
53201
13313

```
11351
33203
30323
14033
33311

13313
13043
32303
50231
13331
```

③ A number of schools are connected to a computer network. Agreements have been developed among those schools: each school maintains a list of schools to which it distributes software (the "receiving schools"). Note that if B is in the distribution list of school A, then A does not necessarily appear in the list of school B

You are to write a program that computes the minimal number of schools that must receive a copy of the new software in order for the software to reach all schools in the network according to the agreement (Subtask A). As a further task, we want to ensure that by sending the copy of new software to an arbitrary school, this software will reach all schools in the network. To achieve this goal we may have to extend the lists of receivers by new members. Compute the minimal number of extensions that have to be made so that whatever school we send the new software to, it will reach all other schools (Subtask B). One extension means introducing one new member into the list of receivers of one school.

Input Data

The first line of file INPUT.TXT contains an integer N: the number of schools in the network (2<=N<=100). The schools are identified by the first N positive integers. Each of the next N lines describes a list of receivers. The line i+1 contains the identifiers of the receivers of school i. Each list ends with a 0. An empty list contains a 0 alone in the line.

Output Data

Your program should write two lines to the file OUTPUT.TXT. The first line should contain one positive integer: the solution of subtask A. The second line should contain the solution of subtask B.

Example Input and Output

INPUT.TXT

5
2 4 3 0
4 5 0
0
0
1 0

OUTPUT.TXT
1
2

④

```
    1   2   3   4   5   6   7
  #############################
1 #   |   #   |   #   |   |   #
  #####---#####---#---#####---#
2 #   #   |   #   #   #   #   #
  #---#####---#####---#####---#
3 #   |   |   #   #   #   #   #
  #---#########---#####---#---#
4 # ->#   |   |   |   |   #   #
  #############################

      #  = Wall
      |  = No wall
      -  = No wall
      -> = It points to the wall to remove according to the example output.
```

Above figure shows the map of a castle.

Write a program that calculates

1. how many rooms the castle has
2. how big the largest room is
3. Which wall to remove from the castle to make as large a room as possible.
   The castle is divided into m * n (m<=50, n<=50) square modules. Each such module can have between zero and four walls.

Input Data

The map is stored in the INPUT.TXT file in the form of numbers, one for each module.

- The file starts with the number of modules in the north-south direction and the number of modules in the east-west direction.

- In the following lines each module is described by a number (0<=p<=15). This number is the sum of: 1 (= wall to the west), 2 (= wall to the north), 4 (= wall to the east), 8 (= wall to the south). Inner walls are defined twice; a wall to the south in module 1,1 is also indicated as a wall to the north in module 2,1.
- The castle always has at least two rooms.

INPUT.TXT for example:

```
4
7
11 6 11 6 3 10 6
7 9 6 13 5 15 5
1 10 12 7 13 7 5
13 11 10 8 10 12 13
```

Output Data

In the OUTPUT.TXT file, the following are written on three lines: First the number of rooms, then the area of the largest room (counted in modules) and a suggestion of which wall to remove (first the row and then the column of the module next to the wall and finally the compass direction that points to the wall).

Output example

("4 1 E" is one of several possibilities, you need only produce one):

```
5
9
4 1 E
```

SECTION – B

1 Write following C# programs using LINQ:

1. To Count File Extensions and Group it using LINQ
2. To Calculate Size of File using LINQ
3. To Generate Odd Numbers in Parallel using LINQ
4. To Implement IEnumerable Interface using LINQ
5. To Divide Sequence into Groups using LINQ
6. To Display the Student Details using Select Clause LINQ
   Create student class & populate the data in to StudentList.
7. To Display the Greatest numbers in an Array using WHERE Clause LINQ
8. To Display the Smallest numbers in a List Collection using FROM Clause LINQ
9. To Implement Let Condition using LINQ

To do this you need to create sample basic database (or Classes of at least two) with at least two simple stored procedures (to insert & Select)

2 Write C# programs to illustrate ADO.NET

1. To read data from data Reader (from SQL Query)
2. To execute a stored procedure to insert record in database
3. To read the multiple result sets from a stored procedure
4. To read scalar values from SQL Command
5. Using Dataset & Data Adapter

3 Write C# programs to illustrate Entity Framework

1. Code-first Approach
2. Database-first Approach

4 Write C# programs to demonstrate the use of Microsoft Enterprise Library

1. Data Access Application Block
2. Logging Application Block
3. Exception Handling Application Block

5 Write C# program to do logging in event logger.