CS101 Lab Quiz 2 - Batch B
16 April 2024 - 20:30 hrs to 23:00 hrs
3 Compulsory Questions - 40 Marks (10, 10, 20)

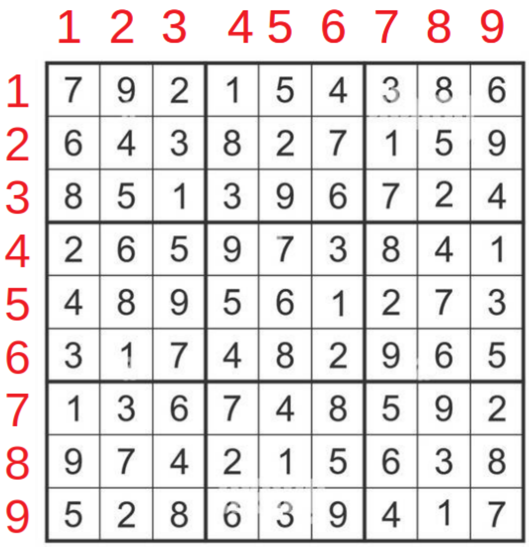
Instructions:

- Keep your ID card on the table for ready reference. If your ID card isn't with you, you won't be allowed to appear for the quiz/exam.
- Keep your phones, tablets, notes, bags, books, etc., near the instructor's platform.
- Rough sheets will be provided to you.
- **Create a folder on your `Desktop` and name it `submission_YourRollNumber`**
  E.g.: If my roll number is `23k1234` then my folder name is `submission_23k1234`
- **Create all three programs in the newly created folder.**
- Name the program files as mentioned in this pdf only.
- No clarifications will be provided for any question by anyone (TAs/Instructor). When in doubt, make suitable assumptions, state them clearly as comments in your program file itself, and proceed to solve the problem.
- Please note that your answers should NOT include any programming concept that hasn't been covered in the class. You are not allowed to use any advanced concepts of C/C++ like strings, vectors, etc.. Such solutions if found will NOT be graded.
- Marks will be given for each hidden test case that passes.
- At this stage of the course, we expect you to write correct code and fix compilation and logical errors by yourselves. Hence, there will not be any partial marks for any errors. Marks will be given for each hidden test case that passes. Cribs like only one semicolon is missing or instead of <, <= should have been written, etc. will NOT be considered. TAs will not make any changes to your program. It is your responsibility to make it work.
- TAs would be around to help you with respect to the logistics like saving programs, compiling, submitting, etc. They will not help you debug the error or resolve your issues related to syntax/logical errors in your program.
- All the Best!

Filename: LQ02_B_Q1.cpp   Validate Sudoku Puzzle   (10 Marks)

Sudoku is a logic-based number-placement puzzle. Write a C++ program to take input a 9x9 array containing digits from 1 to 9. Your task is to check whether the given 9x9 array is a valid sudoku or not. If it is invalid, print the count of invalid rows, invalid columns, and invalid grids, respectively, in a single line, separated by spaces and if it is valid print 0 0 0.

A Sudoku puzzle is valid if each row, each column, and each of the nine 3x3 subgrids that compose the larger grid contains all of the digits from 1 to 9 exactly once. A solved valid sudoku puzzle is given below. The 3x3 subgrids are marked with darker lines. (Sub-grid 1 is from cell: 1,1 to 3,3; Sub-grid 2 is from cell: 1,4 to 3,6; Sub-grid 3 is from 1,7 to 3,9, and so on).



	1	2	3	4	5	6	7	8	9
1	7	9	2	1	5	4	3	8	6
2	6	4	3	8	2	7	1	5	9
3	8	5	1	3	9	6	7	2	4
4	2	6	5	9	7	3	8	4	1
5	4	8	9	5	6	1	2	7	3
6	3	1	7	4	8	2	9	6	5
7	1	3	6	7	4	8	5	9	2
8	9	7	4	2	1	5	6	3	8
9	5	2	8	6	3	9	4	1	7

Input Format:

The input consists of nine lines, each containing nine space-separated digits representing a row in the Sudoku puzzle.

Output Format:

Print three integers in a single line: the count of invalid rows, the count of invalid columns, and the count of invalid grids, separated by a space in case of invalid sudoku and print 0 0 0 in case of valid sudoku that denotes no invalid rows, columns, grids.

Assume

- Assume that the input entered by the user will always be between 1 and 9 (both inclusive).

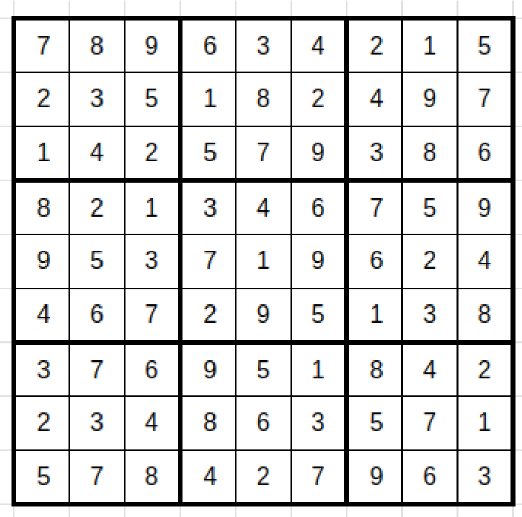
Example Input

```
7 8 9 6 3 4 2 1 5
2 3 5 1 8 2 4 9 7
1 4 2 5 7 9 3 8 6
8 2 1 3 4 6 7 5 9
9 5 3 7 1 9 6 2 4
4 6 7 2 9 5 1 3 8
3 7 6 9 5 1 8 4 2
2 3 4 8 6 3 5 7 1
5 7 8 4 2 7 9 6 3
```

Example Output

```
4 3 3
```

Explanation



7	8	9	6	3	4	2	1	5
2	3	5	1	8	2	4	9	7
1	4	2	5	7	9	3	8	6
8	2	1	3	4	6	7	5	9
9	5	3	7	1	9	6	2	4
4	6	7	2	9	5	1	3	8
3	7	6	9	5	1	8	4	2
2	3	4	8	6	3	5	7	1
5	7	8	4	2	7	9	6	3

- For this explanation, we are assuming that row and column starts with index 1 and not 0.
- Four rows are invalid (2, 5, 8, and 9) as the numbers {2, 9, 3, and 7} are repeated twice, respectively
- Three columns are invalid (1, 2, and 9) as the numbers {2, 3, and 9} are repeated twice, respectively.
- Three grids are invalid (1, 5, 7) as the numbers {2, 9, and 3} are repeated twice, respectively.

Practice Test cases:

Input	Output

7 8 9 6 3 4 2 1 5
2 3 5 1 8 2 4 9 7
1 4 2 5 7 9 3 8 6
8 2 1 3 4 6 7 5 9
9 5 3 7 1 9 6 2 4
4 6 7 2 9 5 1 3 8
3 7 6 9 5 1 8 4 2
2 3 4 8 6 3 5 7 1
5 7 8 4 2 7 9 6 3
	4 3 3

5 3 4 6 7 8 9 1 2
6 7 2 1 9 5 3 4 8
1 9 8 3 4 2 5 6 7
8 5 9 7 6 1 4 2 3
4 2 6 8 5 3 7 9 1
7 1 3 9 2 4 8 5 6
9 6 1 5 3 7 2 8 4
2 8 7 4 1 9 6 3 5
3 4 5 2 8 6 1 7 9
	0 0 0

5 3 4 6 7 8 9 1 1
6 7 2 1 9 5 3 4 8
1 9 8 3 4 2 5 6 7
8 5 9 7 6 1 4 2 3
4 2 6 8 5 3 7 9 1
7 1 3 9 2 4 8 5 6
9 6 1 5 3 7 2 8 4
2 8 7 4 1 9 6 3 5
3 4 5 2 8 6 1 7 9
	1 1 1

4 6 5 9 2 8 3 7 1
7 3 9 6 1 4 2 8 5
8 2 1 3 0 5 9 6 4
9 1 3 5 8 7 6 4 2
6 8 4 2 9 1 5 3 6
5 7 2 4 3 6 8 1 9
3 5 6 1 4 2 7 9 8
2 4 8 7 6 9 1 5 3
1 9 7 1 5 3 4 2 6
	3 3 3

7 8 3 1 4 7 9 3 6
9 6 1 3 2 8 5 4 7
2 4 5 9 6 5 1 2 8
1 3 6 5 8 2 7 9 4
7 9 8 4 1 6 2 5 3
2 5 4 7 9 3 8 6 1
6 2 5 8 3 1 4 7 9
4 1 3 2 7 9 6 8 5
8 7 9 6 5 4 3 1 2
	2 2 0

4 4 5 9 2 8 3 7 1
7 3 9 6 1 4 2 8 5
8 2 1 3 0 5 9 6 4
9 1 3 5 8 7 6 4 2
6 8 4 2 9 1 5 3 6
5 7 2 4 3 6 8 1 9
3 5 6 1 4 2 7 9 8
2 4 8 7 7 9 1 5 3
1 9 7 1 5 3 4 2 6
	5 4 4

8 8 3 1 4 7 9 3 6
9 6 1 3 2 8 5 4 7
2 4 5 9 6 5 1 2 8
1 3 6 5 8 2 7 9 4
7 9 8 4 1 6 2 5 3
6 2 5 8 3 1 4 7 9
2 5 4 7 9 3 8 6 1
4 1 3 2 7 9 6 8 5
8 7 9 6 5 4 3 3 2
	3 3 7

3 3 4 6 7 8 9 1 1
6 7 2 1 9 5 3 4 8
1 9 8 3 4 2 5 6 7
8 5 9 7 6 1 4 2 3
4 2 6 8 5 3 7 9 1
7 1 3 9 2 4 8 5 6
9 6 1 5 3 7 2 8 4
2 8 7 4 9 9 6 3 5
3 4 5 2 8 6 1 7 9
	2 3 3

Filename: LQ02_B_Q2.cpp Timestamp Operations Using Structures (10 Marks)

As you know, time is represented using hours and minutes (not to be confused with the time shown on the clock). In this programming assignment, using structures, we would like to perform operations on time.

- **Compare two time objects**
  E.g. T1: 6 hrs 12 mins and T2: 4 hrs 10 mins
  T1 is greater than T2

- **Add two time objects**
  E.g. 2 hrs 15 mins + 4 hrs 55 mins = 7 hrs 10 mins
  While adding, make sure that you handle overflow of minutes
  i.e. minutes should always be less than or equal to 60

- **Subtract two time objects**
  E.g. 6 hrs 10 mins - 4 hrs 30 mins = 1 hrs 40 mins
  While subtracting, make sure that you handle overflow of minutes
  i.e. minutes should always be less than or equal to 60
  Note that the first time object will always be greater than or equal to the second one

- **Print a time object**
  E.g. 5 hrs 25 min

- **Add minutes to a time object**
  E.g. 0 hrs 10 mins + 49 minutes = 0 hrs 59 mins

A skeleton code is given to you (just below the test cases, in the Helper code section on Bodhitree and as a .cpp file on the Desktop). It has:

- A structure with member variables defined
- Functions that you need to code up
- A main program that creates objects for structures and calls the required functions

Please go through the comments given in the skeleton code for more details.

Note:

- Do NOT modify the main_program
- You need to complete the code in the functions
- If you want, you can create more functions if needed.
- No need to take care of the output. Only complete the code and don’t change the output format.

Assume the following:

- Assume that the input for hours and minutes will always be between 0 and 100000
- Sum of Hours or Minutes will not exceed 2000000

Practice Test Cases:

Input	Output

6 12
4 10
0 10
49
	6 hrs 12 mins + 4 hrs 10 mins = 10 hrs 22 mins
	6 hrs 12 mins - 4 hrs 10 mins = 2 hrs 2 mins
	0 hrs 10 mins + 49 minutes = 0 hrs 59 mins

12 61
10 60
13 0
121
	13 hrs 1 mins + 11 hrs 0 mins = 24 hrs 1 mins
	13 hrs 1 mins - 11 hrs 0 mins = 2 hrs 1 mins
	13 hrs 0 mins + 121 minutes = 15 hrs 1 mins

12 59
10 60
13 0
121
	12 hrs 59 mins + 11 hrs 0 mins = 23 hrs 59 mins
	12 hrs 59 mins - 11 hrs 0 mins = 1 hrs 59 mins
	13 hrs 0 mins + 121 minutes = 15 hrs 1 mins

10 21
10 21
10 21
40
	10 hrs 21 mins + 10 hrs 21 mins = 20 hrs 42 mins
	10 hrs 21 mins - 10 hrs 21 mins = 0 hrs 0 mins
	10 hrs 21 mins + 40 minutes = 11 hrs 1 mins

10 60
12 61
13 0
121
	11 hrs 0 mins + 13 hrs 1 mins = 24 hrs 1 mins
	13 hrs 1 mins - 11 hrs 0 mins = 2 hrs 1 mins
	13 hrs 0 mins + 121 minutes = 15 hrs 1 mins

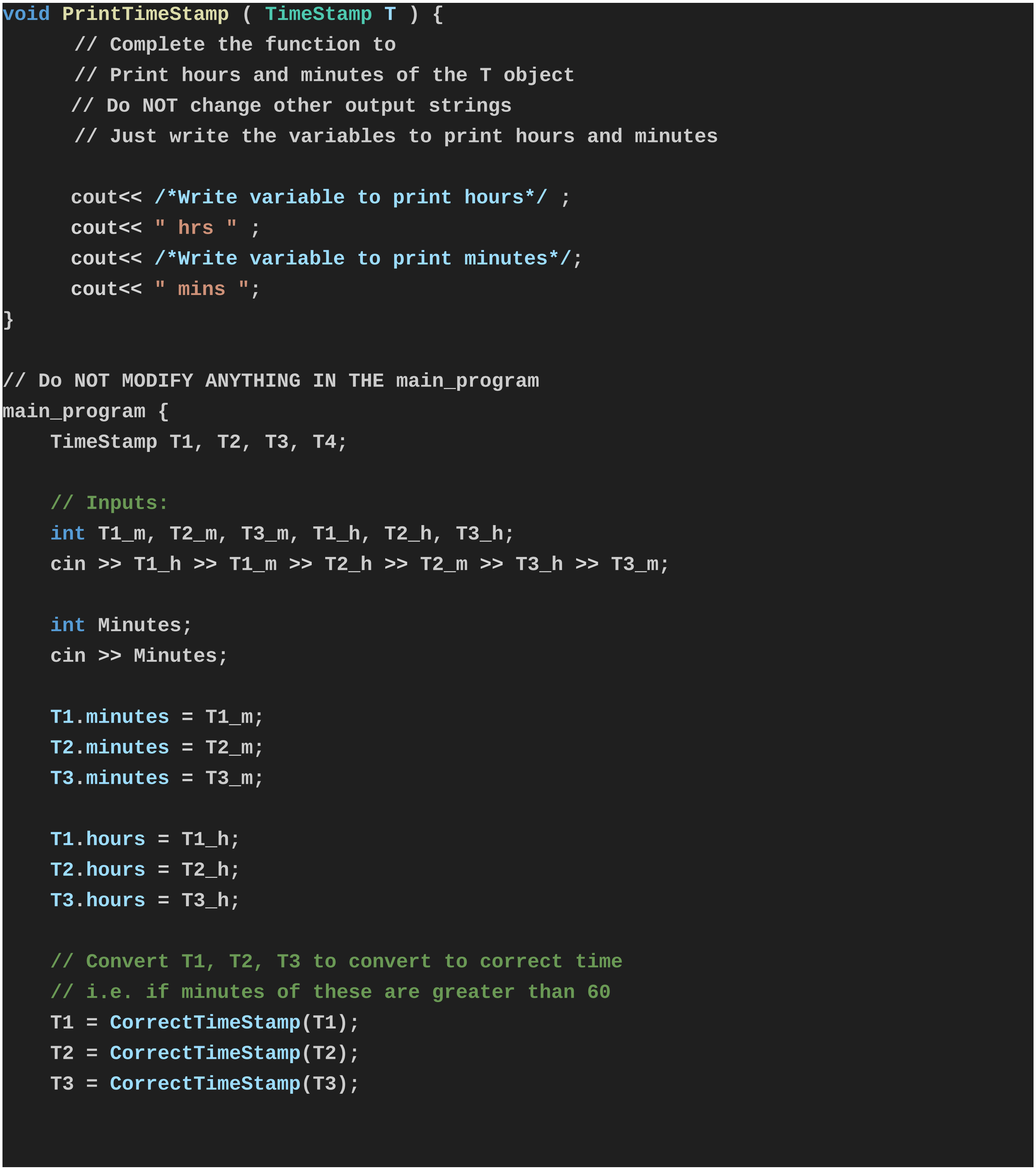
100 160
101 61
13 340
121
	102 hrs 40 mins + 102 hrs 1 mins = 204 hrs 41 mins
	102 hrs 40 mins - 102 hrs 1 mins = 0 hrs 39 mins
	18 hrs 40 mins + 121 minutes = 20 hrs 41 mins

1 0
0 58
0 31
29
	1 hrs 0 mins + 0 hrs 58 mins = 1 hrs 58 mins
	1 hrs 0 mins - 0 hrs 58 mins = 0 hrs 2 mins
	0 hrs 31 mins + 29 minutes = 1 hrs 0 mins

2 30
1 40
1 52
9
	2 hrs 30 mins + 1 hrs 40 mins = 4 hrs 10 mins
	2 hrs 30 mins - 1 hrs 40 mins = 0 hrs 50 mins
	1 hrs 52 mins + 9 minutes = 2 hrs 1 mins

Skeleton Code for your reference

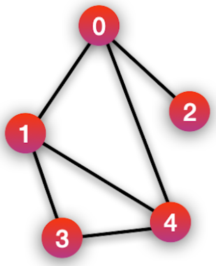
```
#include <simplecpp>

struct TimeStamp {
    // Member Variables
    int hours;
    int minutes ;
};

TimeStamp CorrectTimeStamp(TimeStamp T){
      // Complete the function to handle overflow of minutes
      // Return the correct timestamp
      // E.g. if T.hours = 2 and T.minutes = 75
      //After execution of this function, it should be
      // T.hours = 3 and T.minutes = 15
      // You can call this function from other functions if needed

}

TimeStamp Add ( TimeStamp T1, TimeStamp T2 ) {
      // complete the function to add T1 and T2
      // return the result as a TimeStamp object back to the main
}

TimeStamp Subtract( TimeStamp T1, TimeStamp T2){
      // complete the function to subtract T2 from T1 (T1-T2)
      // return the result as a TimeStamp object back to the main

}

bool Compare(TimeStamp T1, TimeStamp T2){
      // Complete the function to compare T1 and T2
      // return True if T1 >= T2, else return false

}

TimeStamp AddMinutesToTimeStamp(TimeStamp T3 ,int Minutes){
      // complete the function to add Minutes to and T3 object
      // return the result as a TimeStamp object back to the main

}
```



```
void PrintTimeStamp ( TimeStamp T ) {
      // Complete the function to
      // Print hours and minutes of the T object
      // Do NOT change other output strings
      // Just write the variables to print hours and minutes

      cout<< /*Write variable to print hours*/ ;
      cout<< " hrs " ;
      cout<< /*Write variable to print minutes*/;
      cout<< " mins ";
}

// Do NOT MODIFY ANYTHING IN THE main_program
main_program {
    TimeStamp T1, T2, T3, T4;

    // Inputs:
    int T1_m, T2_m, T3_m, T1_h, T2_h, T3_h;
    cin >> T1_h >> T1_m >> T2_h >> T2_m >> T3_h >> T3_m;

    int Minutes;
    cin >> Minutes;

    T1.minutes = T1_m;
    T2.minutes = T2_m;
    T3.minutes = T3_m;

    T1.hours = T1_h;
    T2.hours = T2_h;
    T3.hours = T3_h;

    // Convert T1, T2, T3 to convert to correct time
    // i.e. if minutes of these are greater than 60
    T1 = CorrectTimeStamp(T1);
    T2 = CorrectTimeStamp(T2);
    T3 = CorrectTimeStamp(T3);
    // T1 + T2
    T4 = Add(T1,T2);
    PrintTimeStamp(T1);
    cout<<" + ";
    PrintTimeStamp(T2);
    cout<<" = ";
    PrintTimeStamp(T4);
    cout<<endl;

    // Compare T1 and T2
    if(Compare(T1,T2)) { // If T1 >= T2 then perform T1 - T2
        PrintTimeStamp(T1);
        cout<<" - ";
        PrintTimeStamp(T2);
        cout<<" = ";
        PrintTimeStamp(Subtract(T1,T2));
    }
    else { // Else perform T2 - T1
        PrintTimeStamp(T2);
        cout<<" - ";
        PrintTimeStamp(T1);
        cout<<" = ";
        PrintTimeStamp(Subtract(T2,T1));
    }
    cout<<endl;

    // T3 + Minutes
    PrintTimeStamp(T3);
    cout<<" + ";
    cout<<Minutes;
    cout<<" minutes = ";
    T4 = AddMinutesToTimeStamp(T3,Minutes);
    PrintTimeStamp(T4);
    cout<<endl;
}
```

Filename: LQ02_B_Q3.cpp Shortest Path (20 Marks)

In discrete mathematics, *graphs* are composed of *nodes* and *edges*. Nodes represent the entities (think of them as cities), while the edges represent connections between nodes (think of them as roads). An undirected graph is a type of graph where edges have no direction i.e. if node 1 is connected to node 4, then node 4 is also connected to node 1. As an example, consider the figure given below. We have an Undirected Graph with 5 Nodes and 6 Edges. The Nodes are numbered as 0 to 4.



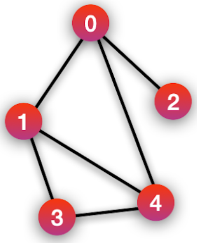
Un-Directed Graph
with *5 Nodes* and *6 Edges*

0	1	1	0	1
1	0	0	1	1
1	0	0	0	0
0	1	0	0	1
1	1	0	1	0

`graph[5][5]`
`Adjacency Matrix`

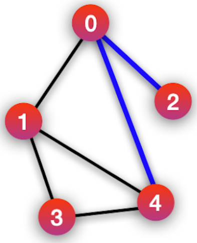
In programming, one can represent a graph as an *adjacency matrix*, where `graph[i][j] = 1` indicates that there is an edge between `node i` and `node j`, and `graph[i][j] = 0` indicates no edge between them.

Now assume that you need to find the shortest path from Node 4 to Node 2. So, you will traverse from Node 4 to Node 0 and then to Node 2. The path (edges) are highlighted in blue within the figure presented on the right.



Un-Directed Graph
with *5 Nodes* and *6 Edges*

Task to *find shortest path between*
Source S and Destination D



Shortest Path from
Node 4 to 2

Write a C++ program to accept the total number of nodes N, followed by the source node S, and destination node D, followed by the graph in the form of an adjacency matrix. Your task is to find the shortest path from S to D and print the number of edges. So, in the above example, 2 should have been pritnted. Note that there will always be a path and only one shortest path between S and D.

Assume that each edge in the graph has a weight of 1. Ignore this if you do not understand it.

Input Format:

- The first line denotes the total Number of nodes N (Numbered 0 to N-1).

- The next line has two integers S (source node) and D (destination node), separated with a space.
- The following lines contain the Binary Adjacency Matrix of size NxN.

Output Format:
Print the number of edges in the shortest path from S to D

Assume:

- The value of variable N entered by the user will always be between 3 and 100 (both inclusive)
- The value of S and D entered by the user will always be less than N
- The user will always enter the matrix elements as 0 or 1.

Refer to the Practice tests on the next page

Practice Test Cases:

Input	Output

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

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

25
21 12
0 1 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0
1 0 1 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0
0 1 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 1 1 0 0
0 0 0 0 1 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 1 0 0
0 0 0 1 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0
0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 1 0 0 1 0
0 0 0 0 0 0 0 1 0 0 0 0 0 0 1 0 0 0 0 0 0 0 0 0 0
0 0 0 0 0 0 1 0 1 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0
0 0 0 0 0 0 0 1 0 1 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0
0 0 0 0 0 0 0 0 1 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 1
0 0 0 0 0 0 0 0 0 0 0 1 0 0 0 0 0 0 0 0 0 0 0 0 0
0 0 0 0 0 0 0 0 0 0 1 0 1 0 0 0 0 0 0 0 0 0 0 0 1
0 0 0 0 0 0 0 0 0 0 0 1 0 0 0 0 0 0 0 0 0 1 0 0 0
0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 1 1 0 0 0 0 0 0 0
0 0 0 0 0 0 1 0 0 0 0 0 0 0 0 0 1 0 0 0 0 0 0 0 0
0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 1 0 0 0 0 0 0 0
0 0 0 0 0 0 0 0 0 0 0 0 0 1 1 0 0 0 0 0 0 0 0 0 0
0 0 0 0 0 0 0 0 0 0 0 0 0 1 0 1 0 0 0 0 1 0 0 0 0
0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 1 0 0 0 0
0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 1 0 0 0 0
0 0 0 0 0 1 0 0 0 0 0 0 0 0 0 0 0 1 1 1 0 0 0 0 0
0 0 1 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 1 0
0 0 1 1 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0
0 0 0 0 0 1 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 1 0 0 0
0 0 0 0 0 0 0 0 0 1 0 1 0 0 0 0 0 0 0 0 0 0 0 0 0
	14

16
2 1
0 1 0 0 0 0 0 0 0 0 0 0 0 0 0 0
1 0 0 0 0 0 0 0 0 0 1 0 0 1 0 0
0 0 0 0 0 0 0 0 0 1 0 0 0 0 0 0
0 0 0 0 0 0 0 1 1 0 0 0 0 0 0 0
0 0 0 0 0 0 1 0 0 1 0 0 0 0 0 0
0 0 0 0 0 0 1 0 1 0 0 0 0 0 0 0
0 0 0 0 1 1 0 0 0 0 0 0 0 0 0 0
0 0 0 1 0 0 0 0 0 0 0 1 1 0 0 1
0 0 0 1 0 1 0 0 0 0 0 0 0 0 0 0
0 0 1 0 1 0 0 0 0 0 0 0 0 0 0 0
0 1 0 0 0 0 0 0 0 0 0 0 0 0 1 0
0 0 0 0 0 0 0 1 0 0 0 0 1 0 0 0
0 0 0 0 0 0 0 1 0 0 0 1 0 1 0 0
0 1 0 0 0 0 0 0 0 0 0 0 1 0 0 0
0 0 0 0 0 0 0 0 0 0 1 0 0 0 0 1
0 0 0 0 0 0 0 1 0 0 0 0 0 0 1 0
	10

7
2 6
0 1 0 0 0 1 0
1 0 1 0 0 0 0
0 1 0 1 0 1 0
0 0 1 0 1 0 1
0 0 0 1 0 0 0
1 0 1 0 0 0 0
0 0 0 1 0 0 0
	2

6
5 3
0 1 0 0 0 0
1 0 1 0 0 0
0 1 0 1 1 0
0 0 1 0 0 0
0 0 1 0 0 1
0 0 0 0 1 0
	3

10
9 3
0 1 1 0 0 0 0 0 0 0
1 0 1 0 0 0 0 0 0 0
1 1 0 1 1 1 1 1 1 1
0 0 1 0 0 0 1 0 0 0
0 0 1 0 0 1 0 0 0 0
0 0 1 0 1 0 0 0 0 0
0 0 1 1 0 0 0 0 0 0
0 0 1 0 0 0 0 0 0 0
0 0 1 0 0 0 0 0 0 0
0 0 1 0 0 0 0 0 0 0
	2

12
8 2
0 1 1 0 0 0 0 0 1 0 1 0
1 0 1 0 0 0 1 0 0 0 0 0
1 1 0 0 0 0 0 0 0 0 0 0
0 0 0 0 1 1 0 0 0 0 0 1
0 0 0 1 0 1 0 0 0 0 0 0
0 0 0 1 1 0 0 0 0 1 0 0
0 1 0 0 0 0 0 1 1 0 0 0
0 0 0 0 0 0 1 0 1 1 0 0
1 0 0 0 0 0 1 1 0 0 0 0
0 0 0 0 0 1 0 1 0 0 1 0
1 0 0 0 0 0 0 0 0 1 0 0
0 0 0 1 0 0 0 0 0 0 0 0
	2