**PROJECT REPORT**

**ON**

**“SUDOKU GAME IN C”**

**SUBMITTED BY:**

Ajitesh Bargotra

Roll Number:2022a1r128

**A picture containing text, clipart

Description automatically generated**

**SUBMITTED TO: BHAGYALAKSHMI MAM**

**Department Name: Computer Science & Engineering (CSE)**

**Model Institute of Engineering and Technology (Autonomous)**

**Jammu, India**

**2023**

**CANDIDATES’ DECLARATION**

I, **Ajitesh Bargotra (2022a1r128),** hereby declare that the work which is being presented in the Project Report entitled, “**Sudoku Game in C**” in partial fulfilment of requirement for the award of degree of B.E. (CSE) and submitted in the Computer Science & Engineering department, Model Institute of Engineering and Technology (Autonomous), Jammu is an authentic record of my own work carried by me at “MIET, Jammu” under the supervision and mentorship of **Shafalika mam,Assistant Professor at Model Institute of Engineering and Technology**. The matter presented in this report has not been submitted in this or any other University / Institute for the award of B.E. Degree.

*Dated*:15-02-2023

**(Ajitesh Bargotra)**

**Roll Number:2022a1r128**

**Group Members**

* Ajitesh Bargotra (2022a1r128)
* Kalash Bakshi (2022a1r127)
* Anushka Raina (2022a1r123)

**Contents**

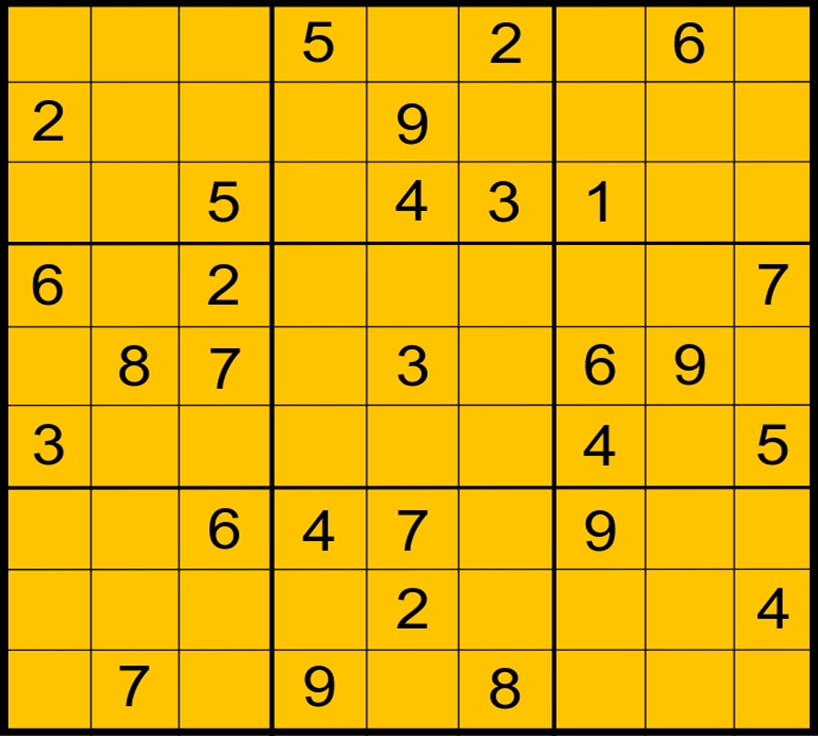
1. Candidates’ declaration
2. Aim of project
3. Introduction
4. Rules for Sudoku
5. Algorithm
6. Flowchart
7. Code for Sudoku game
8. Explanation of the code
9. Output

**OUR PROJECT**

Our group had been assigned the project titled, “Sudoku”. We used the C language for creating this project.

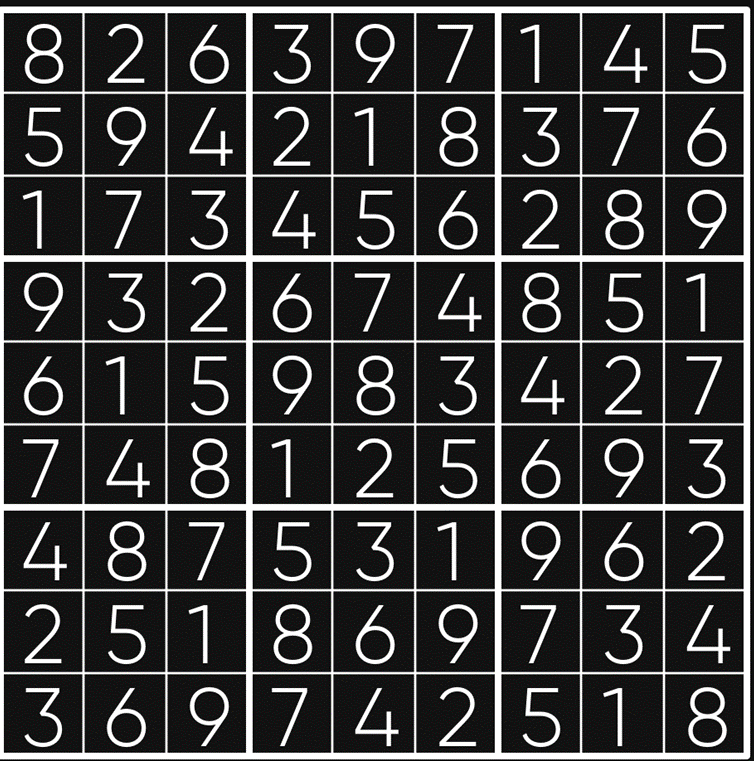
**INTRODUCTION**

SUDOKU, also known as Su Doku, popular form of number game. In its simplest and most common configuration, sudoku consists of a 9 × 9 grid with numbers appearing in some of the squares. The object of the puzzle is to fill the remaining squares, using all the numbers 1–9 exactly once in each row, column, and the nine 3 × 3 sub grids.

****

**RULES FOR SUDOKU**

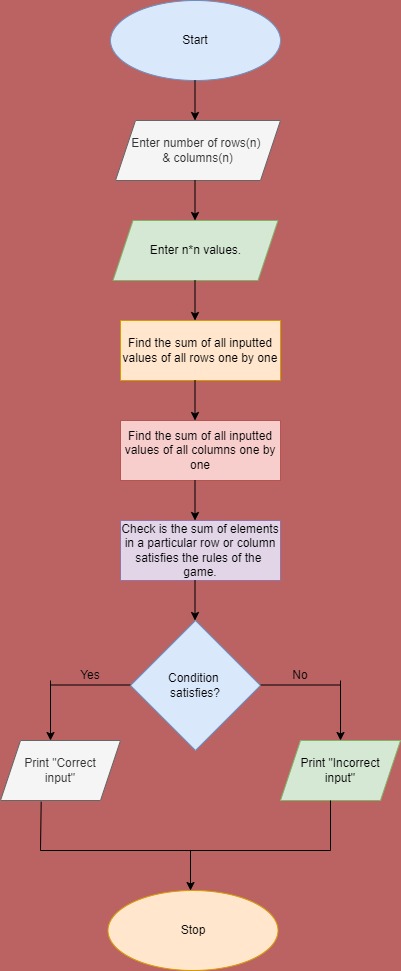
* The rules for sudoku are simple. A 9×9 square must be filled in with numbers from 1-9 with no repeated numbers in each line, horizontally or vertically. To challenge you more, there are 3×3 squares marked out in the grid, and each of these squares can't have any repeat numbers either.
* The ‘45 rule’ in Sudoku is a basic solving strategy that comes from the fact that the digits in each row, column and 3×3 box will always sum to 45.



**ALGORITHM**

* We begin by asking users to enter the number of rows(n) and column(n) they want for the Sudoku game.
* After that the user is asked to enter values. For example, if a user is playing a 9x9 sudoku game, he/she will have to enter 81 values.
* After entering all the values, the user clicks on Enter and the inputted values are displayed in the form of a table with n numbers of rows and columns.
* Along with this the result i.e., if the inputted values satisfy the rules of the game or not. For ex, in a 9x9 sudoku, we have the ‘rule of 45’ which says that the digits in each row, column and 3×3 box will always sum to 45.
* So, if the condition is satisfied, it will display correct input otherwise it will display incorrect input.

**FLOWCHART**



**CODE FOR SUDOKU GAME**

#include<stdio.h> //header files

int main()

{

int i,j,n,s; //declaring variables

int k,l; //declaring variables

int value[9][9]; //initializing matrix

int total\_row[9]; //initializing array for row

int total\_col[9]; //initializing array for column

printf("'enter the inputs of a soduko,and check whether its correct or not'\n");

printf("\n input no of 'rows x column' \n");

scanf("\n%d",&k); //taking input from user

scanf("\t%d",&l); //taking input from user

printf("values\n");

for(i=0;i<k;i++) //for loop for rows

{

total\_row[i]=0; //initializing array for row

for(j=0;j<l;j++) //for loop for columns

{

scanf("%d",&value[i][j]); //taking input from user

total\_row[i]=total\_row[i]+value[i][j]; //adding the values of row

}

}

for(j=0;j<l;j++) //for loop for columns

{

total\_col[j]=0; //initializing array for column

for(i=0;i<k;i++) //for loop for rows

{

total\_col[j]=total\_col[j]+value[i][j]; //adding the values of column

}

}

for(i=0;i<k;i++) //for loop for rows

{

printf("\n"); //for new line

for(j=0;j<l;j++) //for loop for columns

{

printf(" \t %d ",value[i][j]); //printing the values of matrix

}

}

s=((k\*(k+1))/2); //formula for 45 Rule in Sudoku

for(n=0;n<k;n++)

{

if((total\_col[n])!=s) //checking the condition

printf("\n incorrect i/p's col\_%d ",n+1 ); //printing the output

else if((total\_row[n])!=s)

printf(" incorrect i/p's row\_%d ",n+1);

else if(total\_row[n] && s && total\_col[n])

printf("\n correct i/p for both %d row and col ",n+1);

else if(total\_row[n] && s )

printf("\n correct i/p for %d row ",n+1);

else if(s && total\_col[n])

printf("\n correct i/p for %d col ",n+1);

else

{

}

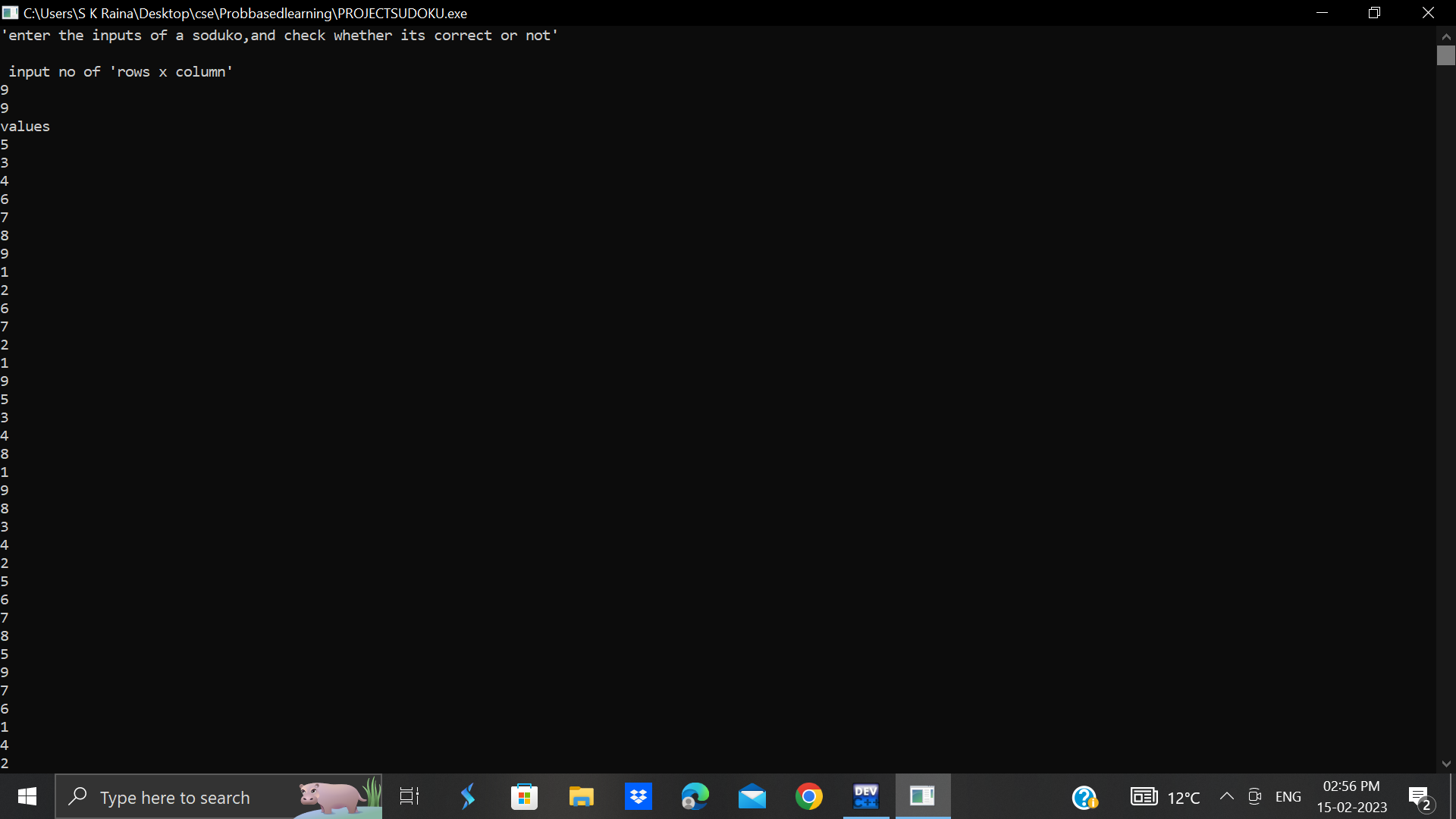
}

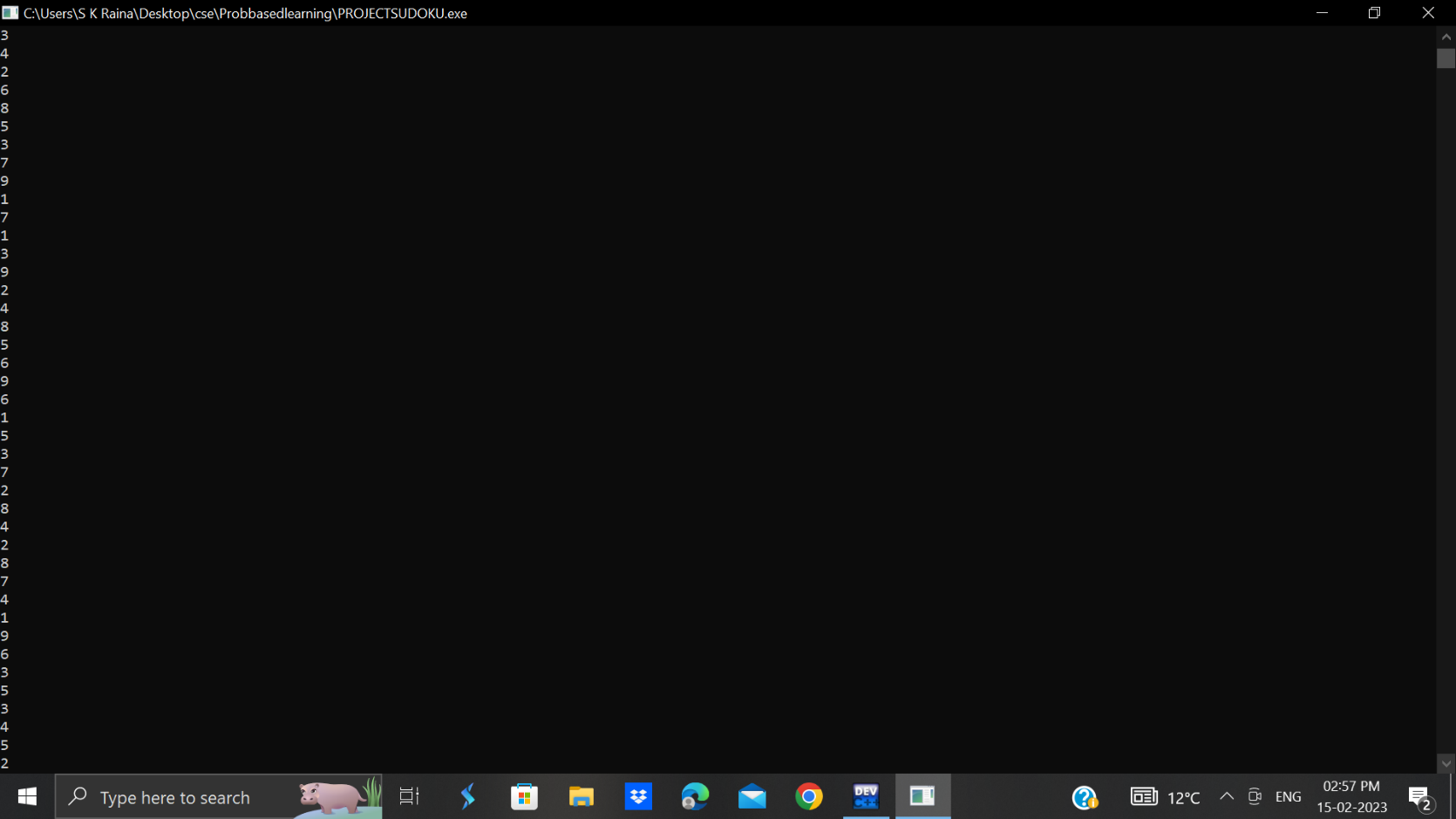
}

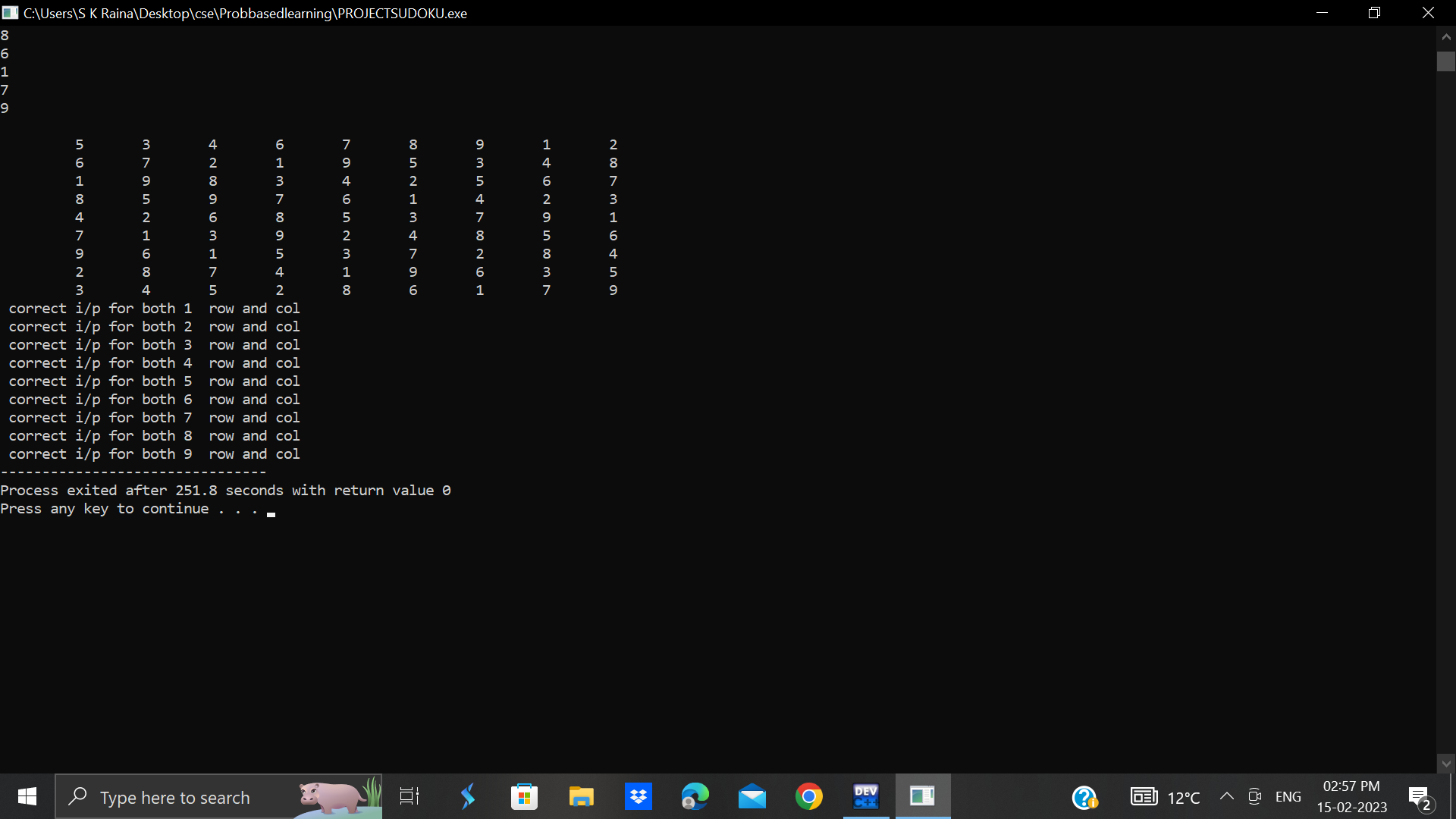
**EXPLAINATION OF THE CODE:**

* #include<stdio.h> - stdio.h is a header file which has the necessary information to include the input/output related functions in our program. Example printf, scanf etc. If we want to use printf or scanf function in our program, we should include the stdio.h header file in our source code.
* int main() - The main function serves as the starting point for program execution. It usually controls program execution by directing the calls to other functions in the program.
* int i,j,n,s; - here, we are declaring four variables i,j,n,s of integer datatype. We shall need them later for controlling the looping statements.
* int k,l; - here, we are declaring two variables k,l of integer datatype. k is for the number of rows and l is for number of columns.
* int value[9][9] – here, we are declaring a 2D array with 9 rows and 9 columns
* int total\_row[9]; - here, we are declaring a 1D array of size=9, datatype=integer and name= total\_row
* int total\_col[9]; - here, we are declaring a 1D array of size=9, datatype=integer and name= total\_col
* printf("'enter the inputs of a soduko, and check whether its correct or not'\n"); - The printf() is a library function to send formatted output to the screen. The function prints the string inside quotations. Here, it will display 'enter the inputs of a soduko, and check whether its correct or not’.
* printf("\n input no of 'rows x column' \n"); - here, it will display, ‘input no of 'rows x column' on the screen.
* scanf("\n%d",&k); - it will take the input from the user for the number of rows.
* scanf("\n%d",&l); - it will take the input from the user for the number of columns.
* printf("values\n"); - here, it will display ‘values’ on the screen.
* for(i=0;i<k;i++) - here, we are using a for loop, i is the looping variable initialized by 0, condition is that i should be less k and every time the loop is executed, the value of i will be incremented by 1.
* total\_row[i]=0; - here, we are initializing array for row with 0.
* for(j=0;j<l;j++) - here, we are using a for loop, j is the looping variable initialized by o, condition is that j should be less l and every time the loop is executed, the value of j will be incremented by 1.
* scanf("%d",&value[i][j]); - it will take input from users for the 2D array declared earlier. It will fill elements row wise, like the 0th row first, then 1st row and so on.
* total\_row[i]=total\_row[i]+value[i][j]; - it will calculate the sum of elements in a particular row. We will need it later for checking the condition.
* for(j=0;j<l;j++) - here, we are using a for loop, j is the looping variable initialized by o, condition is that j should be less l and every time the loop is executed, the value of j will be incremented by 1.
* total\_col[i]=0; - here, we are initializing array for column with 0.
* for(i=0;i<k;i++) - here, we are using a for loop, i is the looping variable initialized by 0, condition is that i should be less k and every time the loop is executed, the value of i will be incremented by 1.
* total\_col[j]=total\_row[j]+value[i][j]; - it will calculate the sum of elements in a particular column. We will need it later for checking the condition.
* for(i=0;i<k;i++) - here, we are using a for loop, i is the looping variable initialized by 0, condition is that i should be less k and every time the loop is executed, the value of i will be incremented by 1.
* printf("\n"); - it will print a new line every time the loop executes till the condition remains true.
* for(j=0;j<l;j++) - here, we are using a for loop, j is the looping variable initialized by o, condition is that j should be less l and every time the loop is executed, the value of j will be incremented by 1.
* printf(" \t %d ",value[i][j]); - it will display the elements stored in array named value.
* s=((k\*(k+1))/2); - this is the formula for the ‘rule of 45’ in sudoku. The 45 rule in Sudoku is a basic solving strategy that comes from the fact that the digits in each row, column and 3×3 box will always sum to 45. We will need it to check if the user has imputed values correctly or not.
* for(n=0;n<k;n++) - here, we are using a for loop, n is the looping variable initialized by 0, condition is that n should be less k and every time the loop is executed, the value of n will be incremented by 1.
* if((total\_col[n])!=s) – here, we are using an ‘if’ statement with the condition that the sum of elements entered in any column is not equal to s. If this condition turns out to be true, then the statements following it will be executed.
* printf("\n incorrect i/p's col\_%d ",n+1 ); - it will display ‘incorrect i/p's col\_%d’ on the screen but instead of %d, the value of n+1 will be printed.
* else if((total\_row[n])!=s) - here, we are using an ‘else if’ statement with the condition that the sum of elements entered in any row is not equal to s. If this condition turns out to be true, then the statements following it will be executed.
* printf(" incorrect i/p's row\_%d ",n+1); - it will display ‘incorrect i/p's row\_%d’ on the screen but instead of %d, the value of n+1 will be printed.
* else if(total\_row[n] && s && total\_col[n]) - here, we are using an ‘else if’ statement with the condition that the sum of elements entered in any row as well as column is equal to s. If this condition turns out to be true, then the statements following it will be executed.
* printf("\n correct i/p for both %d row and col ",n+1); - it will display ‘correct i/p for both both %d row and col’ on the screen but instead of %d, the value of n+1 will be printed.
* else if(total\_row[n] && s ) - here, we are using an ‘else if’ statement with the condition that the sum of elements entered in any row is equal to s. If this condition turns out to be true, then the statements following it will be executed.
* printf("\n correct i/p for %d row ",n+1); - it will display ‘correct i/p for %d row’ on the screen but instead of %d, the value of n+1 will be printed.
* else if(s && total\_col[n]) - here, we are using an ‘else if’ statement with the condition that the sum of elements entered in any column is equal to s. If this condition turns out to be true, then the statements following it will be executed.
* printf("\n correct i/p for %d col ",n+1); - it will display ‘correct i/p for %d col’ on the screen but instead of %d, the value of n+1 will be printed.
* else {} – here, we have an empty else block.

**OUTPUT**

****

****

****

\*\*\*