



NATIONAL UNIVERSITY OF COMPUTER AND EMERGING
SCIENCES—FAST—NUCES--CFD

PROGRAMMING FUNDAMENTALS

FINAL PROJECT DOCUMENTATION

GROUP PROJECT

MOEEZ AHMAD (21F – 9617)

FASIH-UL-HASAN (21F- 9110)

ALI SHAIKH (21F – 9631)

BCS – 1F

2K 21 BATCH



Department of **COMPUTER SCIENCES**

Table of Contents

2048	3
Console Based game in C++	3
Introduction:	3
About System:	3
Project Techniques:	4
Project Glimpse:	4
Description of codes and their S.S	5
Work before main function:	5
Description:	6
Working in the main function:	7
Description:	10
Update Highest Score Function:	11
Description:	11
Main Display Function:	12
Description:	12
Initialization Function:	13
Description:	13
Move upward Function:	14
Up Arrow key Function:	15
SCREEN SHOTS	30
COMPLETE CODE	34

2048

Console Based game in C++

Introduction:

2048 Game is based on a concept of colliding the same two numbers to create a new number with the total sum of the two numbers. This whole system is in C++ language. This game is a single player interesting game. It is played on a 4×4 grid, with numbers that slide smoothly when a player moves them using the four arrow keys. Every turn, a new number will randomly appear in an empty spot on the board with a value of 2

About System:

2048 Game project is designed in 'C++' language. This system is based on a concept to move numbers on a grid to merge them to form the number 2048. However, you can keep on playing the game by creating larger numbers. In each move, a new number of displays (2). The player has to move the numbers using arrow keys and try to collide the same two numbers which create a new number with the total sum of the two numbers. Numbers slide as far as possible in the chosen direction until they strike on either another number or the edge of the grid.

Project Techniques:

We will try to cover up all the contents and techniques that we have studied throughout the semester. In fact, the game will cover up all contents as follow:

- ✓ Different data types
- ✓ Identifiers
- ✓ Keywords
- ✓ Operators
- ✓ Conditional structures
- ✓ Repetition structures
- ✓ File handling
- ✓ Functions
- ✓ Use of global and local variables
- ✓ Arrays (2D and 3D)

Project Glimpse:

- ✓ The game will only start when the user enters **(n)**, otherwise it will not.
 - ✓ The game will be operated by using **ARROWS KEYS**.
 - ✓ It will count on the score and moves with the game.
 - ✓ The user will lose the game in the following conditions:
 - 1) if score reaches the maximum value.
 - 2) If moves are out of order.
 - 3) If the box is full and there is no another movement of numbers and the user is unable to make the tile of the title **2048**.
 - ✓ And he will win the game if:
 - 1) He scored the tile of **2048** within the moves limit and score limit.
-

Description of codes and their S.S

Work before main function:

```
#include<iostream>
#include<string>
#include<conio.h>
#include<fstream>
#include<stdlib.h>
#include<ctime>
using namespace std;

const int m = 4;
const int n = 4;
int Array[m][n];
int score = 0, maxscore, moves = 0, rem_moves = 0;
bool lost = 0;

void initialization();
void board();
void up();
void down();
void left();
void right();
bool winner();
void UpdateHeighestScore();
bool up_down_check();
bool left_right_check();
void generate_random();
void display();
```

Description:

- ✓ First of all, here we included the header file of different built-in functions and proper functionality in the program.
- ✓ Then we declare and initialized the mostly used variables as **GLOBALLY**. Because these are the variables that will be commonly used by the many functions in the program.
- ✓ So, avoid the repeating declaration, we used some variables **GLOBALLY**.
- ✓ Then we declare the functions prototypes before defining the `main()` function.

Working in the main function:

```
int main()
{
    system("color 0a");

    int n, j;
    char option = ' ';
    char input;
    int start = 0;

    while (true)
    {
        system("cls");
        display();

        if (start == 0)
        {
            while (option != 'n')
            {
                option = _getch();

                initialization();

                score = 0;
                moves = 0;

                for (int a = 0; a <= 1; a++)
                {
                    n = rand() % 4;
                    j = rand() % 4;
                    Array[n][j] = 2;
                }
                start++;
            }
        }

        UpdateHeighestScore();
        board();

        input = _getch();

        if (input == 72)
        {
```

```

        up();
    }
    else if (input == 80)
    {
        down();
    }
    else if (input == 75)
    {
        left();
    }
    else if(input == 77)
    {
        right();
    }
    else if(input == 's')
    {
        while (input != 'r')
        {
            system("cls");
            cout << "Game Paused!!" << endl;
            cout << "Press r To Resume Game" << endl;

            input = _getch();

            if (input == 'r')
            {
                break;
            }
        }
    }
    else if(input == 'q')
    {
        break;
    }
    else if (score >= 15000)
    {
        system("cls");
        cout << endl << endl;
        cout << "\t\t\t\tYou Reach Maximum Score Limit!!!" << endl;
        break;
    }
    else if (moves == 1000)
    {
        system("cls");
        cout << endl << endl;
        cout << "\t\t\t\tYou Reach Maximum Move Limit!!!" << endl;
        break;
    }
}

```



```

else if (winner())
{
    system("cls");
    cout << endl << endl << endl << endl << endl;
    cout << "\n\t\t\t\t\t " << " +=====+ " <<
endl;
    cout << "\t\t\t\t\t " << " |      CONGRATULATIONS      | " <<
endl;
    cout << "\t\t\t\t\t " << " |      YOU WON THE GAME      | " <<
endl;
    cout << "\t\t\t\t\t " << " +=====+ " <<
endl;
    cout << endl << endl << endl;
    break;
}
else if (up_down_check())
{
    system("cls");
    cout << endl << endl << endl << endl << endl;
    cout << "\n\t\t\t\t\t " << " +=====+ " <<
endl;
    cout << "\t\t\t\t\t " << " |      ALAS!      | " <<
endl;
    cout << "\t\t\t\t\t " << " |      YOU LOSE THE GAME      | " <<
endl;
    cout << "\t\t\t\t\t " << " +=====+ " <<
endl;
    cout << endl << endl << endl;
    break;
}
}
}

```

HYPHEN

Description:

- ✓ Here we declare and initialized some local variables of main() function
- ✓ Then build a loop to execute the main function till the condition remains true.
- ✓ And that while loop will break down when condition becomes false.
- ✓ Here calls a display function to display **INSTRUCTIONS LIST**.
- ✓ Here starts condition will evaluates and it will terminate when start increments.
- ✓ Then it will take input from the user to start the game.
- ✓ Then **rand()%4** will generate only **2** on any random position in the grid.
- ✓ In this main function we will also check the winning, losing, game over and grid filled conditions and will display the messages on the output screen.

Update Highest Score Function:

```
void UpdateHighestScore(){
    {
        if (score > maxscore)
        {
            maxscore = score;
            ofstream out;
            out.open("maxscore.txt");
            out << maxscore;
            out.close();
        }
        ifstream in;
        in.open("maxscore.txt");
        in >> maxscore;
        in.close();
    }
}
```

Description:

- ✓ This function updates the highest score and store that score in **.TXT** file.
- ✓ This function will be called when the user crosses the previous highest score.

Main Display Function:

```
void display()
{
    cout << "\t\t\t\t\t" << "+=====+" << endl;
    cout << "\t\t\t\t\t" << "| THE 2048 GAME |" << endl;
    cout << "\t\t\t\t\t" << "+=====+" << endl;
    cout << endl;
    cout << "\t\t\t\t\t" << "+=====+" <<
endl;
    cout << "\t\t\t\t\t" << "    SCORE: " << score << endl;
    cout << "\t\t\t\t\t" << "    MOVES: " << moves << endl;
    cout << "\t\t\t\t\t" << "    REMAINING MOVES: " << (131038 -
moves) << endl;
    cout << "\t\t\t\t\t" << "    HIGHEST SCORE: " << maxscore << endl;
    cout << "\t\t\t\t\t" << "+=====+" <<
endl;
    cout << endl;
    cout << "\t\t\t\t\t" << " [ INSTRUCTIONS ]" << endl;
    cout << endl;
    cout << "\t\t\t\t" << "--MOVE AND JOIN THE NUMBERS TO GET TO THE
2048 TILE !" << endl;
    cout << "\t\t\t\t" << "--USE ARROW KEYS FOR THE MOVEMENT OF THE
NUMBERS THROUGH TILES !" << endl << endl << endl;
    cout << "\t\t\t\t" << "--PRESS  (n)  TO START  THE NEW GAME !" <<
endl;
    cout << "\t\t\t\t" << "--PRESS  (s)  TO PAUSE  THE GAME !" << endl;
    cout << "\t\t\t\t" << "--PRESS  (r)  TO RESUME THE PAUSED GAME !"
<< endl;
    cout << "\t\t\t\t" << "--PRESS  (q)  TO QUIT   THE GAME !" << endl;
    cout << endl << endl;
}
```

Description:

- ✓ This functions the main screen of the game.
- ✓ Also, it will print the **INSTRUCTIONS LINES** on the screen.

Initialization Function:

```
void initialization()
{
    for (int i = 0; i <= 3; i++)
    {
        for (int j = 0; j <= 3; j++)
        {
            Array[i][j] = 0;
        }
    }
}
```

Description:

- ✓ This function will initialize all boxes in the grid to zero.
- ✓ It will do the same process every-time it is called.

Move upward Function:

```
void moveup()
{
    for (int col = 0; col <= 3; col++)
    {
        for (int count = 0; count <= 3; count++)
        {
            if (Array[0][col] == 0)
            {
                Array[0][col] = Array[1][col];
                Array[1][col] = 0;
            }
            if (Array[1][col] == 0)
            {
                Array[1][col] = Array[2][col];
                Array[2][col] = 0;
            }
            if (Array[2][col] == 0)
            {
                Array[2][col] = Array[3][col];
                Array[3][col] = 0;
            }
        }
    }
}
```

Up Arrow key Function:

```
void up()
{
    moveup();

    for (int col = 0; col <= 3; col++)
    {
        if (Array[0][col] == Array[1][col])
        {
            Array[0][col] = Array[1][col] + Array[0][col];
            Array[1][col] = 0;
            score += Array[0][col];
        }
        if (Array[1][col] == Array[2][col])
        {
            Array[1][col] = Array[1][col] + Array[2][col];
            Array[2][col] = 0;
            score += Array[1][col];
        }
        if (Array[2][col] == Array[3][col])
        {
            Array[2][col] = Array[3][col] + Array[2][col];
            Array[3][col] = 0;
            score += Array[2][col];
        }
    }

    moveup();
    moves++;
    generate_random();
    board();
}
```

Moving downward Function:

```
void movedown()
{
    for (int col = 0; col <= 3; col++)
    {
        for (int count = 0; count <= 3; count++)
        {
            if (Array[3][col] == 0)
            {
                Array[3][col] = Array[2][col];
                Array[2][col] = 0;
            }
            if (Array[2][col] == 0)
            {
                Array[2][col] = Array[1][col];
                Array[1][col] = 0;
            }
            if (Array[1][col] == 0)
            {
                Array[1][col] = Array[0][col];
                Array[0][col] = 0;
            }
        }
    }
}
```


Down arrow key Function:

```
void down()
{
    movedown();

    for (int col = 0; col <= 3; col++)
    {
        if (Array[3][col] == Array[2][col])
        {
            Array[3][col] = Array[2][col] + Array[3][col];
            Array[2][col] = 0;
            score += Array[3][col];
        }
        if (Array[2][col] == Array[1][col])
        {
            Array[2][col] = Array[2][col] + Array[1][col];
            Array[1][col] = 0;
            score += Array[2][col];
        }
        if (Array[1][col] == Array[0][col])
        {
            Array[1][col] = Array[0][col] + Array[1][col];
            Array[0][col] = 0;
            score += Array[1][col];
        }
    }

    movedown();
    moves++;
    generate_random();
    board();
}
```

Moving leftward Function:

```
void moveleft()
{
    for (int row = 0; row <= 3; row++)
    {
        for (int count = 0; count <= 3; count++)
        {
            if (Array[row][0] == 0)
            {
                Array[row][0] = Array[row][1];
                Array[row][1] = 0;
            }
            if (Array[row][1] == 0)
            {
                Array[row][1] = Array[row][2];
                Array[row][2] = 0;
            }
            if (Array[row][2] == 0)
            {
                Array[row][2] = Array[row][3];
                Array[row][3] = 0;
            }
        }
    }
}
```

Left arrow key Function:

```
void left()
{
    moveleft();
    for (int row = 0; row <= 3; row++)
    {
        if (Array[row][0] == Array[row][1])
        {
            Array[row][0] += Array[row][1];
            Array[row][1] = 0;
            score = score + Array[row][0];
        }
        if (Array[row][1] == Array[row][2])
        {
            Array[row][1] += Array[row][2];
            Array[row][2] = 0;
            score += Array[row][1];
        }
        if (Array[row][2] == Array[row][3])
        {
            Array[row][2] += Array[row][3];
            Array[row][3] = 0;
            score = score + Array[row][2];
        }
    }

    moveleft();
    moves++;
    generate_random();
    board();
}
```

Moving rightwards Function:

```
void moveright()  
{  
    for (int row = 0; row <= 3; row++)  
    {  
        for (int count = 0; count <= 3; count++)  
        {  
            if (Array[row][3] == 0)  
            {  
                Array[row][3] = Array[row][2];  
                Array[row][2] = 0;  
            }  
            if (Array[row][2] == 0)  
            {  
                Array[row][2] = Array[row][1];  
                Array[row][1] = 0;  
            }  
            if (Array[row][1] == 0)  
            {  
                Array[row][1] = Array[row][0];  
                Array[row][0] = 0;  
            }  
        }  
    }  
}
```

Right arrow key Function:

```
void right()
{
    moveright();
    for (int col = 3; col >= 3; col--)
    {
        for (int row = 0; row <= 3; row++)
        {
            if (Array[row][3] == Array[row][2])
            {
                Array[row][3] += Array[row][2];
                Array[row][2] = 0;
                score = score + Array[row][3];
            }
            if (Array[row][2] == Array[row][1])
            {
                Array[row][2] += Array[row][1];
                Array[row][1] = 0;
                score += Array[row][2];
            }
            if (Array[row][1] == Array[row][0])
            {
                Array[row][1] += Array[row][0];
                Array[row][0] = 0;
                score = score + Array[row][1];
            }
        }
    }
    moveright();
    moves++;
    generate_random();
    board();
}
```

Random generate Function:

```
void generate_random()
{
    int p = 0, y = 0, twofour;
    if (moves < 131038)
    {
        for (int i = 0; i < 16; i++)
        {
            p = rand() % 4;
            y = rand() % 4;
            twofour = rand() % 4;
            if (twofour >= 2)
            {
                if (Array[p][y] == 0)
                {
                    Array[p][y] = 2;
                    i = 16;
                }
            }
            else
            {
                if (Array[p][y] == 0)
                {
                    Array[p][y] = 4;
                    i = 16;
                }
            }
        }
    }
    else
    {
        lost = 1;
    }
}
```

Board Display Function:

```
void board()
{
    system("cls");
    display();
    for (int row = 0; row <= 3; row++)
    {
        cout << "\t\t\t\t\t";
        for (int col = 0; col <= 3; col++)
        {
            if (Array[row][col] == 0)
            {
                cout << " | ";
            }
            else
            {
                if (Array[row][col] < 10)
                {
                    cout << " | " << Array[row][col] << " |";
                }
                else if (Array[row][col] < 100)
                {
                    cout << " | " << Array[row][col] << " |";
                }
                else if (Array[row][col] < 1000)
                {
                    cout << " | " << Array[row][col] << " |";
                }
                else if (Array[row][col] < 10000)
                {
                    cout << " | " << Array[row][col] << " |";
                }
                else
                {
                    cout << " | " << Array[row][col] << " |";
                }
            }
        }
        cout << endl;
    }
}
```

Description:

- ✓ This function displays the grid on the screen.
- ✓ It will print the grid in the form of rows and columns of the order of 4x4.
- ✓ The game will be operate on arrow keys so the grid will play an important role in the game as it is the essential part of the game.

Win check Function:

```
bool winner()
{
    bool win = false;
    for (int row = 0; row <= 3; row++)
    {
        for (int col = 0; col <= 3; col++)
        {
            if (Array[row][col] >= 2048)
            {
                win = true;
                return win;
            }
            else
            {
                win = false;
            }
        }
    }
    return win;
}
```

Description:

- ✓ This function will check the win condition in the game.
- ✓ If any one of the 16 block becomes the tile of **2048**, then the function will be called and it will be executed.
- ✓ After this the message of game winning will be printed on the screen

Up - Down check Function:

```
bool up_down_check()
{
    int a = 0, b = 0, c = 0;
    bool ans;
    for (int row = 0; row <= 3; row++)
    {
        for (int col = 0; col <= 3; col++)
        {
            if (Array[row][col] != 0)
            {
                a++;
            }
        }
    }
    if (a == 16)
    {
        for (int col = 0; col <= 3; col++)
        {
            if (Array[0][col] != Array[1][col])
            {
                b++;
            }
            if (Array[1][col] != Array[2][col])
            {
                b++;
            }
            if (Array[2][col] != Array[3][col])
            {
                b++;
            }
        }
    }

    if (b == 12)
    {
        bool ans = true;
        bool ans2 = left_right_check();
        if (ans == ans2)
        {
            return ans;
        }
    }
}
```

```
else
{
    ans = false;
    return ans;
}
```

Description:

- ✓ This function will check all the values in the all columns from top and bottom and from bottom to top.
- ✓ The variable **a** will be incremented when the if block executes truly.
- ✓ If all if blocks executed truly then **a becomes 12** and it will return bool as true and function also becomes true.
- ✓ Otherwise, it will return false and function also becomes false.

Left - Right check Function:

```
bool left_right_check()
{
    int a = 0, b = 0;
    bool ans3;

    for (int row = 0; row <= 3; row++)
    {
        if (Array[row][3] != Array[row][2])
        {
            a++;
        }
        if (Array[row][2] != Array[row][1])
        {
            a++;
        }
        if (Array[row][1] != Array[row][0])
        {
            a++;
        }
    }

    if (a == 12)
    {
        ans3 = true;
        return ans3;
    }
    else
    {
        ans3 = false;
        return ans3;
    }
}
```

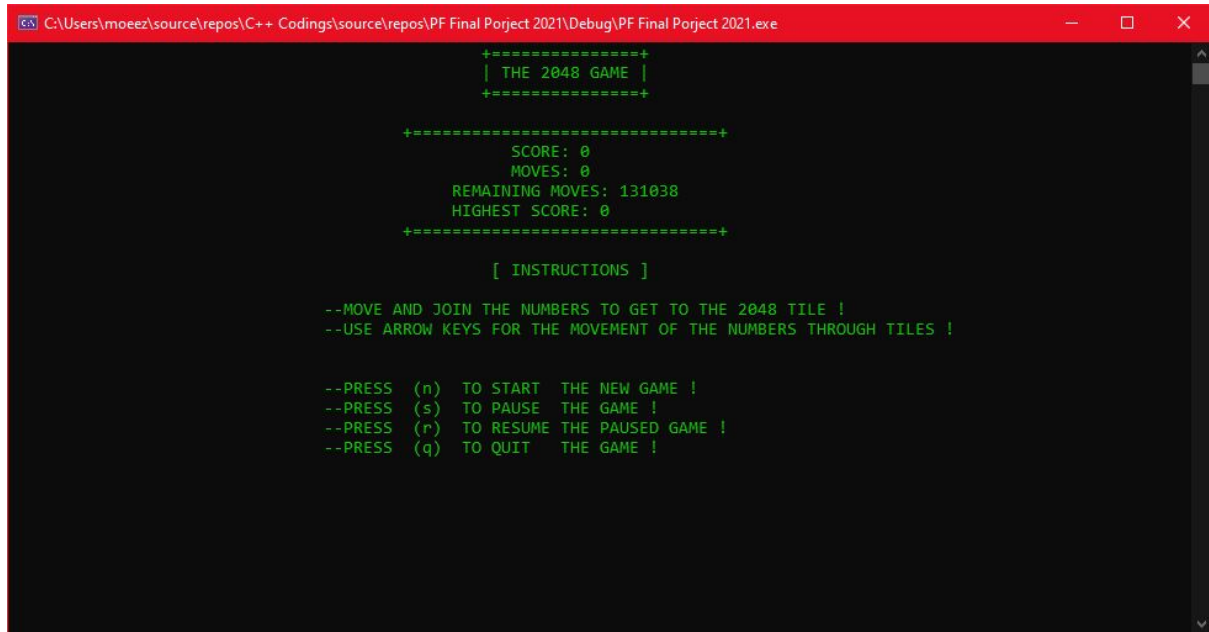
Description:

- ✓ This function will check all the values in the all rows from left and right and from right to left.
- ✓ The variable **a** will be incremented when the if block executes truly.
- ✓ If all if blocks executed truly then **a becomes 12** and it will return bool as true and function also becomes true.
- ✓ Otherwise, it will return false and function also becomes false.

HYPER PROGRAMMERS

SCREEN SHOTS

Main home screen of game:



```
+=====+
| THE 2048 GAME |
+=====+

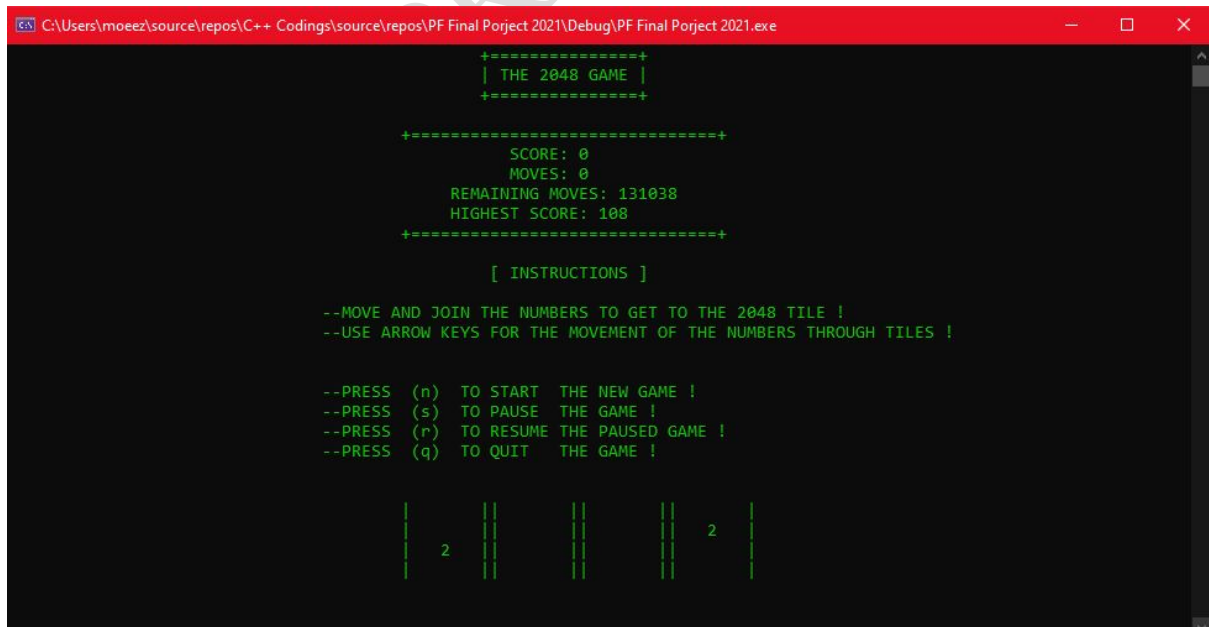
+=====+
          SCORE: 0
          MOVES: 0
    REMAINING MOVES: 131038
          HIGHEST SCORE: 0
+=====+

[ INSTRUCTIONS ]

--MOVE AND JOIN THE NUMBERS TO GET TO THE 2048 TILE !
--USE ARROW KEYS FOR THE MOVEMENT OF THE NUMBERS THROUGH TILES !

--PRESS (n) TO START THE NEW GAME !
--PRESS (s) TO PAUSE THE GAME !
--PRESS (r) TO RESUME THE PAUSED GAME !
--PRESS (q) TO QUIT THE GAME !
```

When the game starts:



```
+=====+
| THE 2048 GAME |
+=====+

+=====+
          SCORE: 0
          MOVES: 0
    REMAINING MOVES: 131038
          HIGHEST SCORE: 108
+=====+

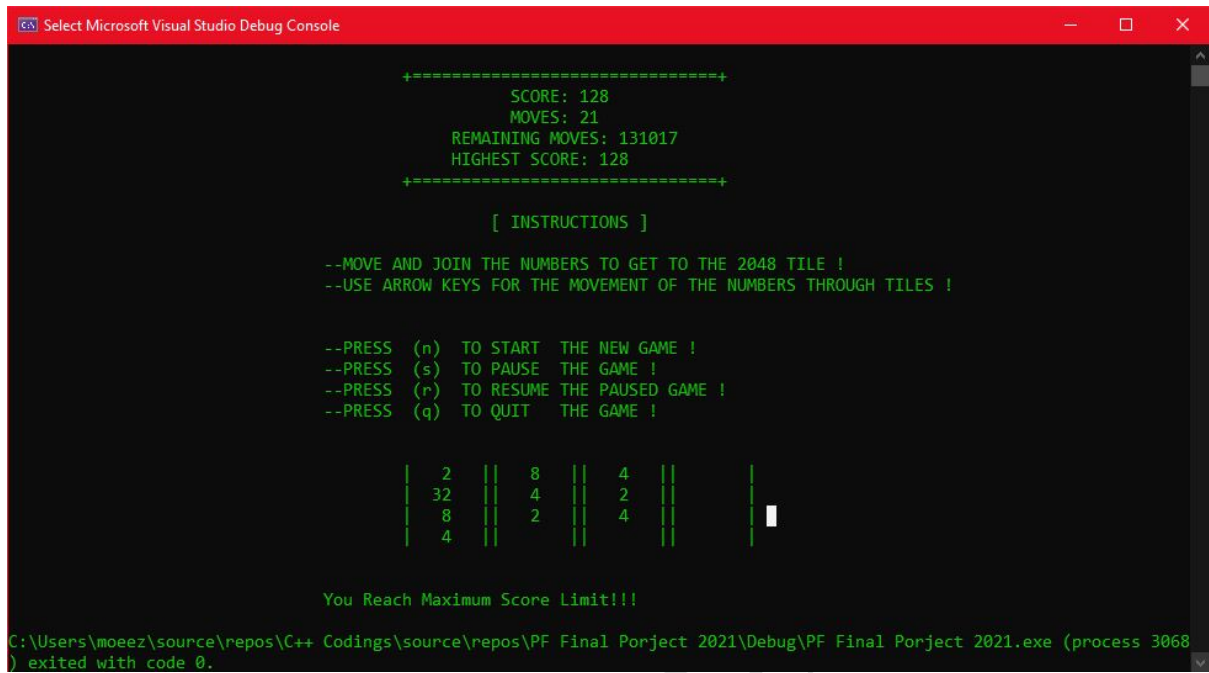
[ INSTRUCTIONS ]

--MOVE AND JOIN THE NUMBERS TO GET TO THE 2048 TILE !
--USE ARROW KEYS FOR THE MOVEMENT OF THE NUMBERS THROUGH TILES !

--PRESS (n) TO START THE NEW GAME !
--PRESS (s) TO PAUSE THE GAME !
--PRESS (r) TO RESUME THE PAUSED GAME !
--PRESS (q) TO QUIT THE GAME !

      | | | |
      | 2 | | |
      | | | |
      | | | |
```

When you reached the maximum score:



```
Select Microsoft Visual Studio Debug Console

+=====+
          SCORE: 128
          MOVES: 21
    REMAINING MOVES: 131017
          HIGHEST SCORE: 128
+=====+

[ INSTRUCTIONS ]

--MOVE AND JOIN THE NUMBERS TO GET TO THE 2048 TILE !
--USE ARROW KEYS FOR THE MOVEMENT OF THE NUMBERS THROUGH TILES !

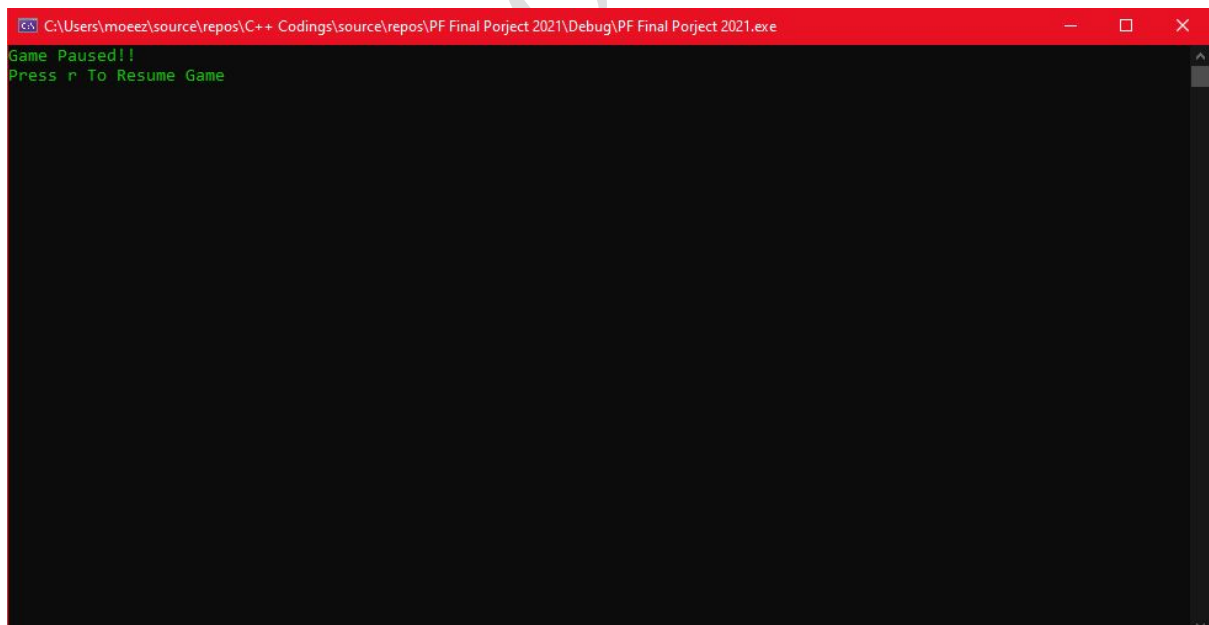
--PRESS (n) TO START THE NEW GAME !
--PRESS (s) TO PAUSE THE GAME !
--PRESS (r) TO RESUME THE PAUSED GAME !
--PRESS (q) TO QUIT THE GAME !

      | 2  || 8  || 4  ||   |
      | 32 || 4  || 2  ||   |
      | 8  || 2  || 4  ||   |
      | 4  ||   ||   ||   |

      You Reach Maximum Score Limit!!!

C:\Users\moeez\source\repos\C++ Codings\source\repos\PF Final Porject 2021\Debug\PF Final Porject 2021.exe (process 3068)
exited with code 0.
```

When the game is paused:



```
C:\Users\moeez\source\repos\C++ Codings\source\repos\PF Final Porject 2021\Debug\PF Final Porject 2021.exe
Game Paused!!
Press r To Resume Game
```

When the game is resumed:

```

C:\Users\moezz\source\repos\C++ Codings\source\repos\PF Final Porject 2021\Debug\PF Final Porject 2021.exe
+=====+
| THE 2048 GAME |
+=====+

+=====+
          SCORE: 24
          MOVES: 10
    REMAINING MOVES: 131028
    HIGHEST SCORE: 128
+=====+

[ INSTRUCTIONS ]

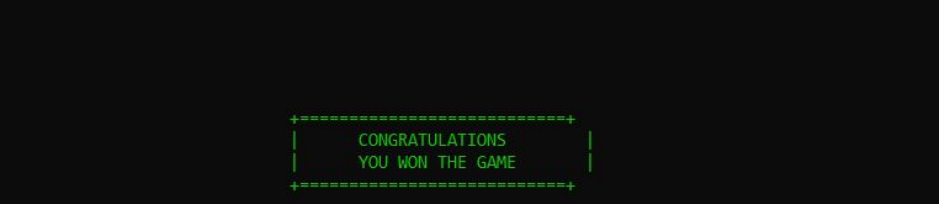
--MOVE AND JOIN THE NUMBERS TO GET TO THE 2048 TILE !
--USE ARROW KEYS FOR THE MOVEMENT OF THE NUMBERS THROUGH TILES !


--PRESS (n) TO START THE NEW GAME !
--PRESS (s) TO PAUSE THE GAME !
--PRESS (r) TO RESUME THE PAUSED GAME !
--PRESS (q) TO QUIT THE GAME !


|   |   |   |   | |
| 2 |   | 8 |   | 8 |
|   |   | 4 |   | 2 |
|   |   | 4 |   | 4 |

```

When you won the game:



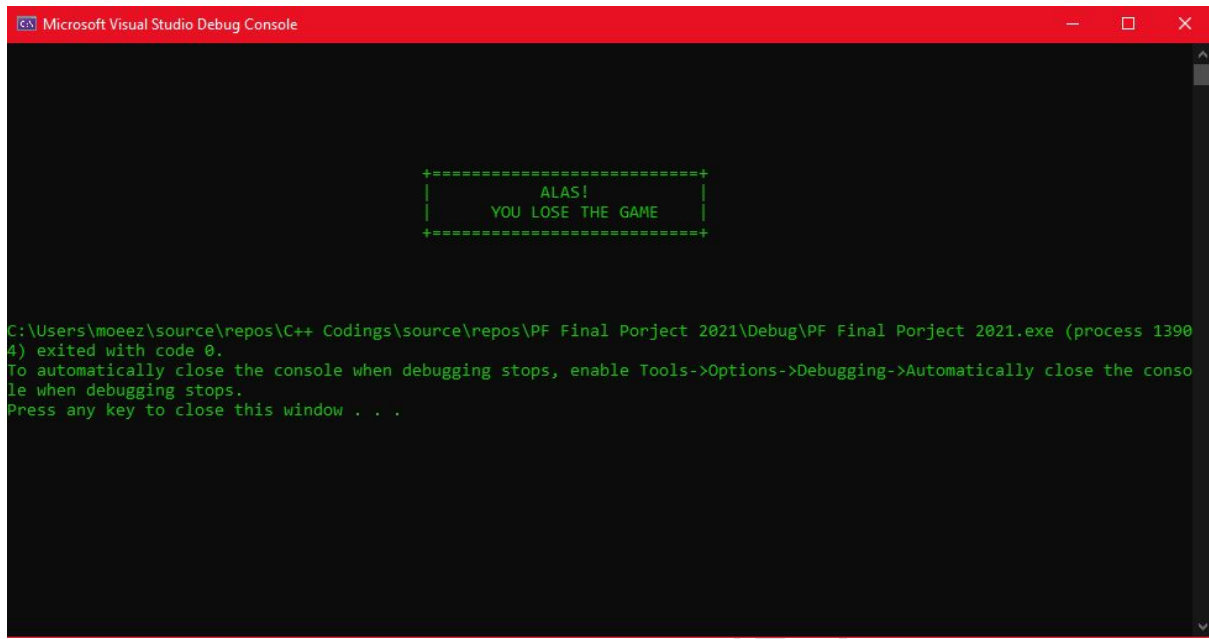
The screenshot shows the Microsoft Visual Studio Debug Console window. The title bar is red and contains the Visual Studio logo, the text "Microsoft Visual Studio Debug Console", and standard window controls (minimize, maximize, close). The console area has a black background with green text. The text displayed is as follows:

```
+=====+  
|          CONGRATULATIONS          |  
|          YOU WON THE GAME          |  
+=====+
```

Below the message, the following text is shown:

```
C:\Users\moeez\source\repos\C++ Codings\source\repos\PF Final Porject 2021\Debug\PF Final Porject 2021.exe (process 6720)  
) exited with code 0.  
To automatically close the console when debugging stops, enable Tools->Options->Debugging->Automatically close the console  
when debugging stops.  
Press any key to close this window . . .
```


When you lost the game:




The screenshot shows the Microsoft Visual Studio Debug Console window. The title bar is red and says "Microsoft Visual Studio Debug Console". The console output is as follows:

```
+=====+  
|      ALAS!  
|  YOU LOSE THE GAME  |  
+=====+
```

Below the message, the console shows the following text:

```
C:\Users\moeez\source\repos\C++ Codings\source\repos\PF Final Porject 2021\Debug\PF Final Porject 2021.exe (process 13904) exited with code 0.  
To automatically close the console when debugging stops, enable Tools->Options->Debugging->Automatically close the console when debugging stops.  
Press any key to close this window . . .
```

.TXT file for updating highest score:



The screenshot shows a Notepad window titled "maxscore.txt - Notepad". The menu bar includes File, Edit, Format, View, and Help. The text content of the file is:

```
128
```

The status bar at the bottom indicates "Ln 1, Col 1", "100%", "Windows (CRLF)", and "UTF-8".

COMPLETE CODE

```
#include<iostream>
#include<string>
#include<conio.h>
#include<fstream>
#include<stdlib.h>
#include<ctime>
using namespace std;

const int m = 4;
const int n = 4;
int Array[m][n];
int score = 0, maxscore, moves = 0, rem_moves = 0;
bool lost = 0;

void initialization();
void board();
void up();
void down();
void left();
void right();
bool winner();
void UpdateHeighestScore();
bool up_down_check();
bool left_right_check();
void generate_random();
void display();

int main()
{
    system("color 0a");

    int n, j;
    char option = ' ';
    char input;
    int start = 0;

    while (true)
    {
        system("cls");
```

```

display();

if (start == 0)
{
    while (option != 'n')
    {
        option = _getch();

        initialization();

        score = 0;
        moves = 0;

        for (int a = 0; a <= 1; a++)
        {
            n = rand() % 4;
            j = rand() % 4;
            Array[n][j] = 2;
        }
        start++;
    }
}

UpdateHeighestScore();
board();

input = _getch();

if (input == 72)
{
    up();
}
else if (input == 80)
{
    down();
}
else if (input == 75)
{
    left();
}
else if (input == 77)
{
    right();
}
else if (input == 's')
{
    while (input != 'r')
    {

```

```

        system("cls");
        cout << "Game Paused!!" << endl;
        cout << "Press r To Resume Game" << endl;

        input = _getch();

        if (input == 'r')
        {
            break;
        }
    }
}
else if(input == 'q')
{
    break;
}
else if (score >= 15000)
{
    system("cls");
    cout << endl << endl;
    cout << "\t\t\t\t\tYou Reach Maximum Score Limit!!!" << endl;
    break;
}
else if (moves == 1000)
{
    system("cls");
    cout << endl << endl;
    cout << "\t\t\t\t\tYou Reach Maximum Move Limit!!!" << endl;
    break;
}
else if (winner())
{
    system("cls");
    cout << endl << endl << endl << endl << endl;
    cout << "\n\t\t\t\t\t\t " << " +=====+ " <<
endl;
    cout << "\t\t\t\t\t\t " << " |          CONGRATULATIONS          | " <<
endl;
    cout << "\t\t\t\t\t\t " << " |          YOU WON THE GAME          | " <<
endl;
    cout << "\t\t\t\t\t\t " << " +=====+ " <<
endl;
    cout << endl << endl << endl;
    break;
}
else if (up_down_check())
{
    system("cls");

```

```

        cout << endl << endl << endl << endl << endl;
        cout << "\t\t\t\t\t" << " +=====+ " <<
endl;
        cout << "\t\t\t\t\t" << " |          ALAS!          | " <<
endl;
        cout << "\t\t\t\t\t" << " |        YOU LOSE THE GAME        | " <<
endl;
        cout << "\t\t\t\t\t" << " +=====+ " <<
endl;

        cout << endl << endl << endl;
        break;
    }
}
}

```

```

void UpdateHeighestScore(){
{
    if (score > maxscore)
    {
        maxscore = score;
        ofstream out;
        out.open("maxscore.txt");
        out << maxscore;
        out.close();
    }
    ifstream in;
    in.open("maxscore.txt");
    in >> maxscore;
    in.close();
}
}

```

```

void display()
{
    cout << "\t\t\t\t\t" << "+=====+" << endl;
    cout << "\t\t\t\t\t" << "| THE 2048 GAME |" << endl;
    cout << "\t\t\t\t\t" << "+=====+" << endl;
    cout << endl;
    cout << "\t\t\t\t\t" << "+=====+" << endl;
    cout << "\t\t\t\t\t" << "    SCORE: " << score << endl;
    cout << "\t\t\t\t\t" << "    MOVES: " << moves << endl;
    cout << "\t\t\t\t\t" << "    REMAINING MOVES: " << (131038 - moves) <<
endl;
}

```

```

    cout << "\\t\\t\\t\\t\\t" << "          HIGHEST SCORE: " << maxscore << endl;
    cout << "\\t\\t\\t\\t\\t" << "+=====+" << endl;
    cout << endl;
    cout << "\\t\\t\\t\\t\\t\\t" << " [ INSTRUCTIONS ]" << endl;
    cout << endl;
    cout << "\\t\\t\\t\\t\\t" << "--MOVE AND JOIN THE NUMBERS TO GET TO THE 2048 TILE !" << endl;
    cout << "\\t\\t\\t\\t\\t" << "--USE ARROW KEYS FOR THE MOVEMENT OF THE NUMBERS THROUGH TILES !" << endl << endl << endl;
    cout << "\\t\\t\\t\\t\\t" << "--PRESS (n) TO START THE NEW GAME !" << endl;
    cout << "\\t\\t\\t\\t\\t" << "--PRESS (s) TO PAUSE THE GAME !" << endl;
    cout << "\\t\\t\\t\\t\\t" << "--PRESS (r) TO RESUME THE PAUSED GAME !" << endl;
    cout << "\\t\\t\\t\\t\\t" << "--PRESS (q) TO QUIT THE GAME !" << endl;
    cout << endl << endl;
}

void initialization()
{
    for (int i = 0; i <= 3; i++)
    {
        for (int j = 0; j <= 3; j++)
        {
            Array[i][j] = 0;
        }
    }
}

void moveup()
{
    for (int col = 0; col <= 3; col++)
    {
        for (int count = 0; count <= 3; count++)
        {
            if (Array[0][col] == 0)
            {
                Array[0][col] = Array[1][col];
                Array[1][col] = 0;
            }
            if (Array[1][col] == 0)
            {
                Array[1][col] = Array[2][col];
                Array[2][col] = 0;
            }
            if (Array[2][col] == 0)
            {
                Array[2][col] = Array[3][col];
                Array[3][col] = 0;
            }
        }
    }
}

```

```

    }
}

}

}

void up()
{
    moveup();

    for (int col = 0; col <= 3; col++)
    {
        if (Array[0][col] == Array[1][col])
        {
            Array[0][col] = Array[1][col] + Array[0][col];
            Array[1][col] = 0;
            score += Array[0][col];
        }
        if (Array[1][col] == Array[2][col])
        {
            Array[1][col] = Array[1][col] + Array[2][col];
            Array[2][col] = 0;
            score += Array[1][col];
        }
        if (Array[2][col] == Array[3][col])
        {
            Array[2][col] = Array[3][col] + Array[2][col];
            Array[3][col] = 0;
            score += Array[2][col];
        }
    }

    moveup();
    moves++;
    generate_random();
    board();
}

void movedown()
{
    for (int col = 0; col <= 3; col++)
    {
        for (int count = 0; count <= 3; count++)
        {
            if (Array[3][col] == 0)
            {
                Array[3][col] = Array[2][col];
            }
        }
    }
}

```

```

        Array[2][col] = 0;
    }
    if (Array[2][col] == 0)
    {
        Array[2][col] = Array[1][col];
        Array[1][col] = 0;
    }
    if (Array[1][col] == 0)
    {
        Array[1][col] = Array[0][col];
        Array[0][col] = 0;
    }
    }
}

void down()
{
    movedown();

    for (int col = 0; col <= 3; col++)
    {
        if (Array[3][col] == Array[2][col])
        {
            Array[3][col] = Array[2][col] + Array[3][col];
            Array[2][col] = 0;
            score += Array[3][col];
        }
        if (Array[2][col] == Array[1][col])
        {
            Array[2][col] = Array[2][col] + Array[1][col];
            Array[1][col] = 0;
            score += Array[2][col];
        }
        if (Array[1][col] == Array[0][col])
        {
            Array[1][col] = Array[0][col] + Array[1][col];
            Array[0][col] = 0;
            score += Array[1][col];
        }
    }

    movedown();
    moves++;
    generate_random();
    board();
}

```



```

void moveleft()
{
    for (int row = 0; row <= 3; row++)
    {
        for (int count = 0; count <= 3; count++)
        {
            if (Array[row][0] == 0)
            {
                Array[row][0] = Array[row][1];
                Array[row][1] = 0;
            }
            if (Array[row][1] == 0)
            {
                Array[row][1] = Array[row][2];
                Array[row][2] = 0;
            }
            if (Array[row][2] == 0)
            {
                Array[row][2] = Array[row][3];
                Array[row][3] = 0;
            }
        }
    }
}

void left()
{
    moveleft();
    for (int row = 0; row <= 3; row++)
    {
        if (Array[row][0] == Array[row][1])
        {
            Array[row][0] += Array[row][1];
            Array[row][1] = 0;
            score = score + Array[row][0];
        }
        if (Array[row][1] == Array[row][2])
        {
            Array[row][1] += Array[row][2];
            Array[row][2] = 0;
            score += Array[row][1];
        }
        if (Array[row][2] == Array[row][3])
        {

```

```

        Array[row][2] += Array[row][3];
        Array[row][3] = 0;
        score = score + Array[row][2];
    }
}

moveleft();
moves++;
generate_random();
board();
}

void moveright()
{
    for (int row = 0; row <= 3; row++)
    {
        for (int count = 0; count <= 3; count++)
        {
            if (Array[row][3] == 0)
            {
                Array[row][3] = Array[row][2];
                Array[row][2] = 0;
            }
            if (Array[row][2] == 0)
            {
                Array[row][2] = Array[row][1];
                Array[row][1] = 0;
            }
            if (Array[row][1] == 0)
            {
                Array[row][1] = Array[row][0];
                Array[row][0] = 0;
            }
        }
    }
}

void right()
{
    moveright();
    for (int col = 3; col >= 3; col--)
    {
        for (int row = 0; row <= 3; row++)
        {
            if (Array[row][3] == Array[row][2])
            {

```

```

        Array[row][3] += Array[row][2];
        Array[row][2] = 0;
        score = score + Array[row][3];
    }
    if (Array[row][2] == Array[row][1])
    {
        Array[row][2] += Array[row][1];
        Array[row][1] = 0;
        score += Array[row][2];
    }
    if (Array[row][1] == Array[row][0])
    {
        Array[row][1] += Array[row][0];
        Array[row][0] = 0;
        score = score + Array[row][1];
    }
}
}
moveright();
moves++;
generate_random();
board();
}
void generate_random()
{
    int p = 0, y = 0, twofour;
    if (moves < 131038)
    {
        for (int i = 0; i < 16; i++)
        {
            p = rand() % 4;
            y = rand() % 4;
            twofour = rand() % 4;
            if (twofour >= 2)
            {
                if (Array[p][y] == 0)
                {
                    Array[p][y] = 2;
                    i = 16;
                }
            }
            else
            {
                if (Array[p][y] == 0)
                {
                    Array[p][y] = 4;
                    i = 16;
                }
            }
        }
    }
}

```

```

    }
}
else
{
    lost = 1;
}
}

void board()
{

    system("cls");
    display();
    for (int row = 0; row <= 3; row++)
    {
        cout << "\t\t\t\t\t";
        for (int col = 0; col <= 3; col++)
        {
            if (Array[row][col] == 0)
            {
                cout << " |      | ";
            }
            else
            {
                if (Array[row][col] < 10)
                {
                    cout << " |  " << Array[row][col] << "  | ";
                }
                else if (Array[row][col] < 100)
                {
                    cout << " |   " << Array[row][col] << "   | ";
                }
                else if (Array[row][col] < 1000)
                {
                    cout << " |    " << Array[row][col] << "    | ";
                }
                else if (Array[row][col] < 10000)
                {
                    cout << " |     " << Array[row][col] << "     | ";
                }
                else
                {
                    cout << " |      " << Array[row][col] << "      | ";
                }
            }
        }
        cout << endl;
    }
}

```

```

    }
}

bool winner()
{
    bool win = false;
    for (int row = 0; row <= 3; row++)
    {
        for (int col = 0; col <= 3; col++)
        {
            if (Array[row][col] >= 2048)
            {
                win = true;
                return win;
            }
            else
            {
                win = false;
            }
        }
    }
    return win;
}

bool up_down_check()
{
    int a = 0, b = 0, c = 0;
    bool ans;
    for (int row = 0; row <= 3; row++)
    {
        for (int col = 0; col <= 3; col++)
        {
            if (Array[row][col] != 0)
            {
                a++;
            }
        }
    }
    if (a == 16)
    {
        for (int col = 0; col <= 3; col++)
        {
            if (Array[0][col] != Array[1][col])
            {
                b++;
            }
            if (Array[1][col] != Array[2][col])

```

```

        {
            b++;
        }
        if (Array[2][col] != Array[3][col])
        {
            b++;
        }
    }

}

if (b == 12)
{
    bool ans = true;
    bool ans2 = left_right_check();
    if (ans == ans2)
    {
        return ans;
    }
}

else
{
    ans = false;
    return ans;
}
}

bool left_right_check()
{
    int a = 0, b = 0;
    bool ans3;

    for (int row = 0; row <= 3; row++)
    {
        if (Array[row][3] != Array[row][2])
        {
            a++;
        }
        if (Array[row][2] != Array[row][1])
        {
            a++;
        }
        if (Array[row][1] != Array[row][0])
        {
            a++;
        }
    }
}

```

```
if (a == 12)
{
    ans3 = true;
    return ans3;
}
else
{
    ans3 = false;
    return ans3;
}
}
```

THE END
