#include <bits/stdc++.h>

#include <iostream>

#include <time.h>

using namespace std;

const int N=9;

class SudokuSolver

{

private:

    int board[N][N] = {{3, 0, 6, 5, 7, 8, 4, 9, 2},

                       {5, 2, 9, 1, 3, 4, 0, 6, 8},

                       {4, 8, 7, 6, 2, 9, 5, 3, 1},

                       {2, 6, 3, 4, 1, 5, 9, 8, 7},

                       {9, 7, 4, 8, 6, 3, 1, 2, 5},

                       {8, 5, 1, 7, 9, 2, 6, 4, 3},

                       {1, 3, 8, 9, 4, 7, 2, 5, 6},

                       {6, 9, 2, 3, 5, 1, 8, 7, 4},

                       {7, 4, 5, 2, 8, 6, 3, 1, 0}

    };

public:

    void menu();

    void inputSudoku(int N);

    bool isvalid(int i, int j, int val);

    bool solve(int arr[N][N], int i, int j);

    void boundry(int arr[N][N]);

    bool single\_player(string);

    bool compare\_sudoku(int userinput[N][N], int solution[N][N]);

};

bool SudokuSolver::compare\_sudoku(int userinput[N][N], int solution[N][N])

{

    int flag = 0;

    for (int i = 0; i < N; i++)

    {

        for (int j = 0; j < N; j++)

        {

            if (userinput[i][j] == solution[i][j])

            {

                flag = 0;

            }

            else

            {

                flag = 1;

            }

        }

    }

    if (flag == 1)

    {

        return false;

    }

    else

    {

        return true;

    }

}

void SudokuSolver::inputSudoku(int N)

{

    for (int i = 0; i < N; i++)

    {

        for (int j = 0; j < N; j++)

        {

            cin >> board[i][j];

        }

    }

}

bool SudokuSolver::single\_player(string player\_name)

{

    cout << "Below is the given Sudoku :- ";

    vector<pair<int, int>> vec(100);

    int k = 0;

    int sol[N][N];

    int temp[N][N];

    for (int i = 0; i < N; i++)

    {

        for (int j = 0; j < N; j++)

        {

            sol[i][j] = board[i][j]; // Copying the sudoku

            temp[i][j] = board[i][j]; // Copying the sudoku

            if (board[i][j] == 0)

            {

                vec[k].first = i;

                vec[k].second = j;

                k++;

            }

        }

    }

    int row, column;

    int value;

    for (int i = 0; i < N; i++)

    {

        for (int j = 0; j < N; j++)

        {

            system("cls");

            cout << "\n\n-------------- Sudoku Game -------------------\n\n"

                 << endl;

            cout << "\n\n-------------- "<<player\_name<< " turn:- -------------------\n\n"

                 << endl;

            boundry(sol);

            cout << "Incomplete indexes " << endl;

            for (int i = 0; i < k; i++)

            {

                cout << "{ " << vec[i].first << "," << vec[i].second << " } ";

            }

            cout << endl;

            cout << "If you have filled all indexes then enter -1 -1 to see your result " << endl;

        start:

            cout << "In which index you want to enter value  ::";

            cin >> row >> column;

            if (row == -1 || column == -1)

            {

                goto end;

            }

            else

            {

                if (sol[row][column] == 0)

                {

                    cout << "Enter Value ";

                    cin >> value;

                    sol[row][column] = value;

                }

                else

                {

                    cout << "This row and column is already filled " << endl;

                    goto start;

                }

            }

        }

    }

end:

    int flag = 0;

    for (int i = 0; i < k; i++)

    {

        if (sol[vec[i].first][vec[i].second] == 0)

        {

            flag = 1;

        }

        else

        {

            flag = 0;

        }

    }

    if (flag == 1)

    {

        cout << "Incompletely Filled Sudoku " << endl;

    }

    else

    {

        solve(temp, 0, 0);

        if (compare\_sudoku(sol, temp) == true)

        {

            cout << "Congrats " << player\_name << ", you are a winner " << endl;

            return true;

        }

        else

        {

            cout << "You lost this game please try again " << endl;

        }

    }

    return false;

}

void SudokuSolver::menu()

{

start:

    cout << "--------------Welcome to Sudoku Solver Game:--------------------" << endl;

    cout << "1)Single Player " << endl;

    cout << "2)Multiplayer Mode" << endl;

    cout << "3)Check your own sudoku:-" << endl;

    cout << "4)Exit " << endl;

    int n;

    cin >> n;

    if (n == 1)

    {

        time\_t now = time(0); // get current dat/time with respect to system

        char \*dt = ctime(&now); // convert it into string

        fstream player;

        player.open("singleplayer.txt", std::ios\_base::app);

        string player\_name;

        cout << "Enter Player Name:-";

        cin >> player\_name;

        clock\_t t;

        t = clock();

        if (single\_player(player\_name) == true)

        {

            t = clock() - t;

            double time\_taken = ((double)t) / CLOCKS\_PER\_SEC; // in seconds

            cout << player\_name << " took " << time\_taken << " seconds to solve the sudoku. \n";

            player << dt << player\_name << " took " << time\_taken << " seconds to solve the sudoku. \n";

        }

        else

        {

            player << dt << player\_name << ", is unable to solve the sudoku " << endl;

            boundry(board);

            cout << "Press 1 to See the solution:-";

            int a;

            cin >> a;

            if (a == 1)

            {

                cout << "Sudoku solution:-" << endl;

                if (solve(board, 0, 0))

                {

                    boundry(board);

                }

                else

                {

                    cout << "Unable to solve ..." << endl;

                }

            }

        }

        cout<<"Thank you for playing "<<endl;

        return ;

    }

    else if (n == 2)

    {

        fstream multi;

        multi.open("multipleplayer.txt", std::ios\_base::app);

        string player1;

        string player2;

        cout << "Enter Player 1 Name ::";

        cin >> player1;

        cout << "Enter Player 2 Name ::";

        cin>>player2;

        clock\_t t1;

        t1 = clock();

        double time\_taken1;

        if (single\_player(player1) == true)

        {

            t1 = clock() - t1;

            time\_taken1 = ((double)t1) / CLOCKS\_PER\_SEC; // in seconds

            cout << player1 << " took " << time\_taken1 << " seconds to solve the sudoku \n";

        }

        int l;

        cout<<"Are "<<player2<<" Ready?? ";

        system("pause");

            clock\_t t2;

            t2 = clock();

            double time\_taken2;

            if (single\_player(player2) == true)

            {

                t2 = clock() - t2;

                time\_taken2 = ((double)t2) / CLOCKS\_PER\_SEC; // in seconds

                cout << player2 << " took " << time\_taken2 << " seconds to solve the sudoku \n";

            }

            multi << "Player 1:" << player1 << ":  tooks " << time\_taken1 << " to solve the sudoku " << endl;

            multi << "Player 2:" << player2 << ":  tooks " << time\_taken2 << " to solve the sudoku " << endl;

            if (t1 < t2)

            {

                cout << player1 << " are Winner " << endl;

                multi << player1 << " are Winner " << endl;

            }

            else if (t1 > t2)

            {

                cout << player2 << " are Winner " << endl;

                multi << player2 << " are Winner " << endl;

            }

            multi << "----------------------------" << endl;

    }

    else if (n == 3)

    {   cout<<"Input your own Sudoku..."<<endl;

        inputSudoku(N);

        cout << "Press 1 to See the solution:-";

        int a;

        cin >> a;

        if (a == 1)

        {

            cout << "Sudoku solution:-" << endl;

            if (solve(board, 0, 0))

            {

                boundry(board);

            }

            else

            {

                cout << "Sudoku pattern is not valid , Unable to solve ..." << endl;

            }

        }

        goto start;

    }

    else

    {

        cout << "Thank you...";

        return;

    }

}

void SudokuSolver::boundry(int arr[N][N])

{

    cout << "------------------" << endl;

    for (int i = 0; i < N; i++)

    {

        for (int j = 0; j < N; j++)

        {

            cout << arr[i][j] << "|";

        }

        cout << endl;

    }

    cout << "------------------" << endl;

}

bool SudokuSolver::isvalid(int i, int j, int val)

{

    for (int k = 0; k < N; k++)

    {

        if (board[i][k] == val)

            return false;

    }

    for (int k = 0; k < N; k++)

    {

        if (board[k][j] == val)

            return false;

    }

    int strow = i - i % 3;

    int stcol = j - j % 3;

    for (int a = 0; a < 3; a++)

    {

        for (int b = 0; b < 3; b++)

        {

            if (board[a + strow][b + stcol] == val)

                return false;

        }

    }

    return true;

}

bool SudokuSolver::solve(int arr[N][N], int i, int j)

{

    if (i == N - 1 && j == N)

    {

        return true;

    }

    if (j == N)

    {

        i++;

        j = 0;

    }

    if (arr[i][j] != 0)

    {

        return solve(arr, i, j + 1);

    }

    else

    {

        for (int num = 1; num <= 9; num++)

        {

            if (isvalid(i, j, num))

            {

                arr[i][j] = num;

                if (solve(arr, i, j + 1))

                {

                    return true;

                }

                arr[i][j] = 0;

            }

        }

    }

    return false;

}

int main()

{

    SudokuSolver sudoku1;

    sudoku1.menu();

    return 0;

}