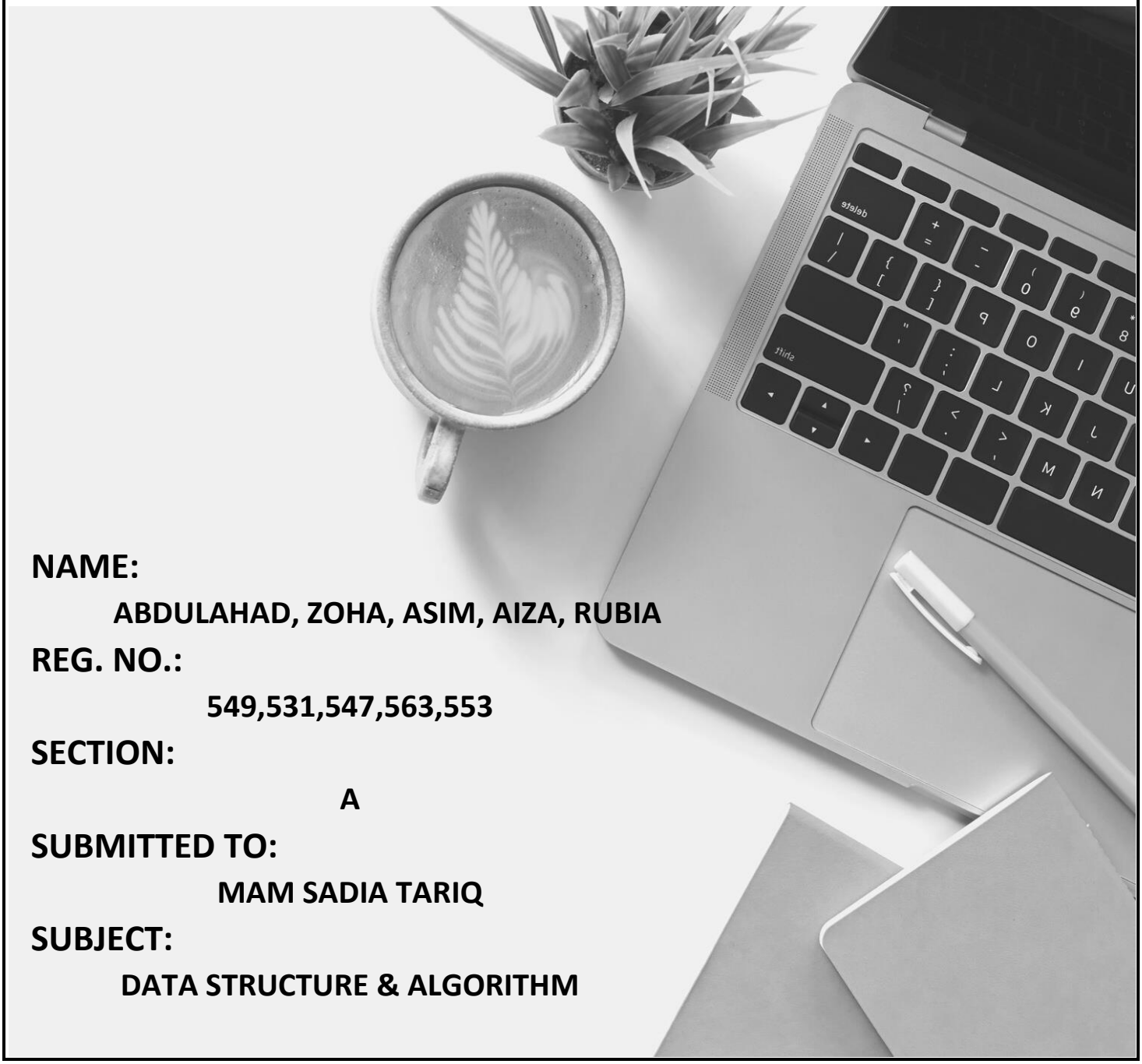
UNIVERSITY OF ENGINEERING AND TECHNOLOGY, LAHORE (NAROWAL CAMPUS)

PROJECT REPORT



NAME:

ABDULAHAD, ZOHA, ASIM, AIZA, RUBIA

REG. NO.:

549,531,547,563,553

SECTION:

A

SUBMITTED TO:

MAM SADIA TARIQ

SUBJECT:

DATA STRUCTURE & ALGORITHM

SUDOKU SOLVER

Sudoku solver mini projects on data structures and algorithms use the backtracking algorithm to resolve Sudoku puzzles. Sudoku is a logic-based number puzzle game that requires you to fill in a 9x9 grid with numbers from 1 to 9, with no repetitions allowed in the rows, columns, or 3x3 sub grids.

The first step in the Sudoku Solver project is to draw the puzzle as a 9x9 grid and fill it in with the specified numbers. The puzzle is then solved using the backtracking approach by iteratively trying out various values in each cell until a solution is discovered.

When the backtracking algorithm runs into a dead end, that is, when it comes into a contradiction that renders the problem impossible to solve, it will turn around and try another possible solution for each cell in the grid. Once a viable answer is found, the algorithm goes back and tests different values in the earlier cells.

The Sudoku Solver data structures and algorithms projects in C++ show how the backtracking algorithm can be used in real-world situations to solve challenging puzzles like Sudoku. It is beneficial for puzzle fans, game creators, and anybody else interested in logic-based problem-solving.

PROJECT CODE

```
#include <iostream>
#include <cstdlib>
#include <cstring>
using namespace std;

const int N = 9;

bool isSafe(int board[N][N], int row, int col, int num)
{
    // Check if 'num' is already in the same row
    for (int i = 0; i < N; i++)
        if (board[row][i] == num)
            return false;

    // Check if 'num' is already in the same column
    for (int i = 0; i < N; i++)
        if (board[i][col] == num)
            return false;

    // Check if 'num' is already in the same 3x3 box
    int boxRowStart = row - row % 3;
    int boxColStart = col - col % 3;
```

```
    for (int i = 0; i < 3; i++)
        for (int j = 0; j < 3; j++)
            if (board[i + boxRowStart][j + boxColStart] == num)
                return false;

    return true;
}

void printBoard(int board[N][N])
{

    system("cls");

    cout << "\n\n\n";
    cout <<
"\t\t\t\t\t*******************************************************" <<
endl;
    cout << "\t\t\t\t\t*     *     *     *     *     *     *     *     *
*" << endl;
    cout << "\t\t\t\t\t*  " << board[0][0] << "  *  " << board[0][1] <<
"  *  " << board[0][2] << "  *  " << board[0][3] << "  *  " << board[0][4]
<< "  *  " << board[0][5] << "  *  " << board[0][6] << "  *  " <<
board[0][7] << "  *  " << board[0][8] << "  *  " << endl;

    cout <<
"\t\t\t\t\t*******************************************************" <<
endl;
    cout << "\t\t\t\t\t*     *     *     *     *     *     *     *     *
*" << endl;

    cout << "\t\t\t\t\t*  " << board[1][0] << "  *  " << board[1][1] <<
"  *  " << board[1][2] << "  *  " << board[1][3] << "  *  " << board[1][4]
<< "  *  " << board[1][5] << "  *  " << board[1][6] << "  *  " <<
board[1][7] << "  *  " << board[1][8] << "  *  " << endl;

    cout <<
"\t\t\t\t\t*******************************************************" <<
endl;
    cout << "\t\t\t\t\t*     *     *     *     *     *     *     *     *
*" << endl;

    cout << "\t\t\t\t\t*  " << board[2][0] << "  *  " << board[2][1] <<
"  *  " << board[2][2] << "  *  " << board[2][3] << "  *  " << board[2][4]
<< "  *  " << board[2][5] << "  *  " << board[2][6] << "  *  " <<
board[2][7] << "  *  " << board[2][8] << "  *  " << endl;

    cout <<
"\t\t\t\t\t*******************************************************" <<
endl;
```

```
    cout << "\t\t\t\t\t\t*     *     *     *     *     *     *     *     *
*" << endl;

    cout << "\t\t\t\t\t\t*  " << board[3][0] << "  *  " << board[3][1] <<
"  *  " << board[3][2] << "  *  " << board[3][3] << "  *  " << board[3][4]
<< "  *  " << board[3][5] << "  *  " << board[3][6] << "  *  " <<
board[3][7] << "  *  " << board[3][8] << "  *  " << endl;

    cout <<
"\t\t\t\t\t\t*****************************************************" <<
endl;
    cout << "\t\t\t\t\t\t*     *     *     *     *     *     *     *     *
*" << endl;

    cout << "\t\t\t\t\t\t*  " << board[4][0] << "  *  " << board[4][1] <<
"  *  " << board[4][2] << "  *  " << board[4][3] << "  *  " << board[4][4]
<< "  *  " << board[4][5] << "  *  " << board[4][6] << "  *  " <<
board[4][7] << "  *  " << board[4][8] << "  *  " << endl;

    cout <<
"\t\t\t\t\t\t*****************************************************" <<
endl;
    cout << "\t\t\t\t\t\t*     *     *     *     *     *     *     *     *
*" << endl;

    cout << "\t\t\t\t\t\t*  " << board[5][0] << "  *  " << board[5][1] <<
"  *  " << board[5][2] << "  *  " << board[5][3] << "  *  " << board[5][4]
<< "  *  " << board[5][5] << "  *  " << board[5][6] << "  *  " <<
board[5][7] << "  *  " << board[5][8] << "  *  " << endl;

    cout <<
"\t\t\t\t\t\t*****************************************************" <<
endl;
    cout << "\t\t\t\t\t\t*     *     *     *     *     *     *     *     *
*" << endl;

    cout << "\t\t\t\t\t\t*  " << board[6][0] << "  *  " << board[6][1] <<
"  *  " << board[6][2] << "  *  " << board[6][3] << "  *  " << board[6][4]
<< "  *  " << board[6][5] << "  *  " << board[6][6] << "  *  " <<
board[6][7] << "  *  " << board[6][8] << "  *  " << endl;

    cout <<
"\t\t\t\t\t\t*****************************************************" <<
endl;
    cout << "\t\t\t\t\t\t*     *     *     *     *     *     *     *     *
*" << endl;

    cout << "\t\t\t\t\t\t*  " << board[7][0] << "  *  " << board[7][1] <<
"  *  " << board[7][2] << "  *  " << board[7][3] << "  *  " << board[7][4]
```

```
<< "  *  " << board[7][5] << "  *  " << board[7][6] << "  *  " <<
board[7][7] << "  *  " << board[7][8] << "  *  " << endl;

    cout <<
"\t\t\t\t\t\t*****************************************************" <<
endl;
    cout << "\t\t\t\t\t\t*     *     *     *     *     *     *     *     *
*" << endl;

    cout << "\t\t\t\t\t\t*  " << board[8][0] << "  *  " << board[8][1] <<
"  *  " << board[8][2] << "  *  " << board[8][3] << "  *  " << board[8][4]
<< "  *  " << board[8][5] << "  *  " << board[8][6] << "  *  " <<
board[8][7] << "  *  " << board[8][8] << "  *  " << endl;

    cout <<
"\t\t\t\t\t\t*****************************************************" <<
endl;
}

bool solveSudoku(int board[N][N], int row, int col)
{
    // If all cells are filled, the puzzle is solved
    if (row == N - 1 && col == N)
        return true;

    // Move to the next row if the current column is N
    if (col == N)
    {
        row++;
        col = 0;
    }

    // Skip the cells that already have a value
    if (board[row][col] != 0)
        return solveSudoku(board, row, col + 1);

    // Try filling the current cell with a number from 1 to 9
    for (int num = 1; num <= 9; num++)
    {
        if (isSafe(board, row, col, num))
        {
            board[row][col] = num;

            if (solveSudoku(board, row, col + 1))
                return true;

            board[row][col] = 0;
        }
    }
    return false;
```

```
}

bool isSolvedCompletely(int grid[N][N])
{
    for (int row = 0; row < N; row++)
        for (int col = 0; col < N; col++)
            if (grid[row][col] == 0)
                return false;

    return true;
}

void playGame(int board[N][N])
{
    int ch;
    int row, col, num;
    while (true)
    {
        printBoard(board);
        cout << endl
             << endl;
        cout << "Unable to solve? Enter -1 as row, col and num to view the
solved sudoku." << endl;
        cout << "Enter row: ";
        cin >> row;
        cout << "Enter column: ";
        cin >> col;
        cout << "Enter number: ";
        cin >> num;

        if (row == -1 || col == -1 || num == -1)
        {
            solveSudoku(board, 0, 0);
            printBoard(board);
            cout << endl;
            cout << "Better luck next time!!!" << endl;
            return;
        }
        if (isSolvedCompletely(board))
            break;
        row--;
        col--;
        if (!isSafe(board, row, col, num))
        {
            cout << "Invalid move. Try again." << endl;
            continue;
        }
        board[row][col] = num;
    }
```

```
    // Check if the user has solved it correctly or not
    bool solved = true;
    for (int i = 0; i < N; i++)
    {
        for (int j = 0; j < N; j++)
        {
            if (board[i][j] == 0)
            {
                solved = false;
                break;
            }
        }
    }

    if (solved)
    {
        cout << "Congratulations! You have solved the puzzle." << endl;
        printBoard(board);
    }
    else
    {
        cout << "Puzzle not solved. Better luck next time." << endl;
    }
}

int main()
{

    int board[N][N] = {
        {3, 0, 6, 5, 0, 8, 4, 0, 0},
        {5, 2, 0, 0, 0, 0, 0, 0, 0},
        {0, 8, 7, 0, 0, 0, 0, 3, 1},
        {0, 0, 3, 0, 1, 0, 0, 8, 0},
        {9, 0, 0, 8, 6, 3, 0, 0, 5},
        {0, 5, 0, 0, 9, 0, 6, 0, 0},
        {1, 3, 0, 0, 0, 0, 2, 5, 0},
        {0, 0, 0, 0, 0, 0, 0, 7, 4},
        {0, 0, 5, 2, 0, 6, 3, 0, 0}};

    while (true)
    {

        int choice;
        cout << endl
             << endl;
        cout << "\t\t[1] Solve the Sudoku" << endl;
        cout << "\t\t[2] Unable to solve? View the solved Sudoku" << endl;
        cout << "\t\t[3] Exit" << endl;
        cout << "\t\tEnter your choice: ";
        cin >> choice;
```

```
        switch (choice)
        {
        case 1:
            playGame(board);
            break;
        case 2:
            if (solveSudoku(board, 0, 0))
            {
                cout << "Completely Solved Sudoku is: " << endl;
                cout << endl
                     << endl;
                printBoard(board);
                cout << endl;
                cout << "Better luck next time!!!" << endl;
            }
            else
                cout << "No solution found" << endl;
            break;
        case 3:
            exit(0);
        default:
            cout << "Invalid choice" << endl;
        }
        return 0;
    }
}
```

.....END.....