|  |  |
| --- | --- |
| C++  Task 2 of Internship | Microsoft Word logo |

# Explanation of Tasks

**Task 1: Quiz Game**

I create a small quiz app where:

The user enters their name to register.

The quiz shows multiple-choice questions.

The user answers by choosing options (1 to 4).

The app tells if the answer is correct or wrong.

At the end, it shows the correct answers and the user's score.

🙪 🙪🙪 🙪 🙪🙪 🙪

**Task 2: Game of Life**

In this I create a simulation of Conway’s Game of Life:

A 5x5 grid shows live (\*) and dead (.) cells.

A predefined pattern (a line of stars) is used.

The grid updates automatically based on rules.

The user can watch how the cells evolve over time.

It simulates how cells live or die depending on their neighbors.

🙪 🙪🙪 🙪 🙪🙪 🙪

C++ PROGRAM FOR ONLINE QUIZ GAME

/\*Your code has some issues outside the loop like answer and score being

used without declaration, and logic outside the quiz loop.

Here's the fixed and complete version\*/

#include <iostream>

#include <vector>

#include <string>

using namespace std;

// Structure for each question

struct Question {

string question;

string options[4];

int correctOption; // index 0-3

Question(string q, string op1, string op2, string op3, string op4, int correct) {

question = q;

options[0] = op1;

options[1] = op2;

options[2] = op3;

options[3] = op4;

correctOption = correct;

}

};

int main() {

string username;

int score = 0, answer;

cout << "Welcome to the Quiz Game!\n";

cout << "Enter your name to register: ";

cin >> username;

vector<Question> quiz;

quiz.push\_back(Question("What is the capital of France?", "1) Berlin",

"2) London", "3) Paris", "4) Rome", 2));

quiz.push\_back(Question("Which planet is known as the Red Planet?",

"1) Earth", "2) Mars", "3) Jupiter", "4) Saturn", 1));

quiz.push\_back(Question("Which language is used for web apps?",

"1) Python", "2) JavaScript", "3) C++", "4) Java", 1));

for (int i = 0; i < quiz.size(); i++) {

cout << "\nQ" << i + 1 << ": " << quiz[i].question << endl;

for (int j = 0; j < 4; j++)

cout << quiz[i].options[j] << endl;

cout << "Your answer (1-4): ";

cin >> answer;

if (answer - 1 == quiz[i].correctOption) {

cout << "Correct!\n";

score++;

} else {

cout << "Wrong. Correct answer: " << quiz[i].options[quiz[i].correctOption] << "\n";

}

}

cout << "\n=== Quiz Complete ===\n";

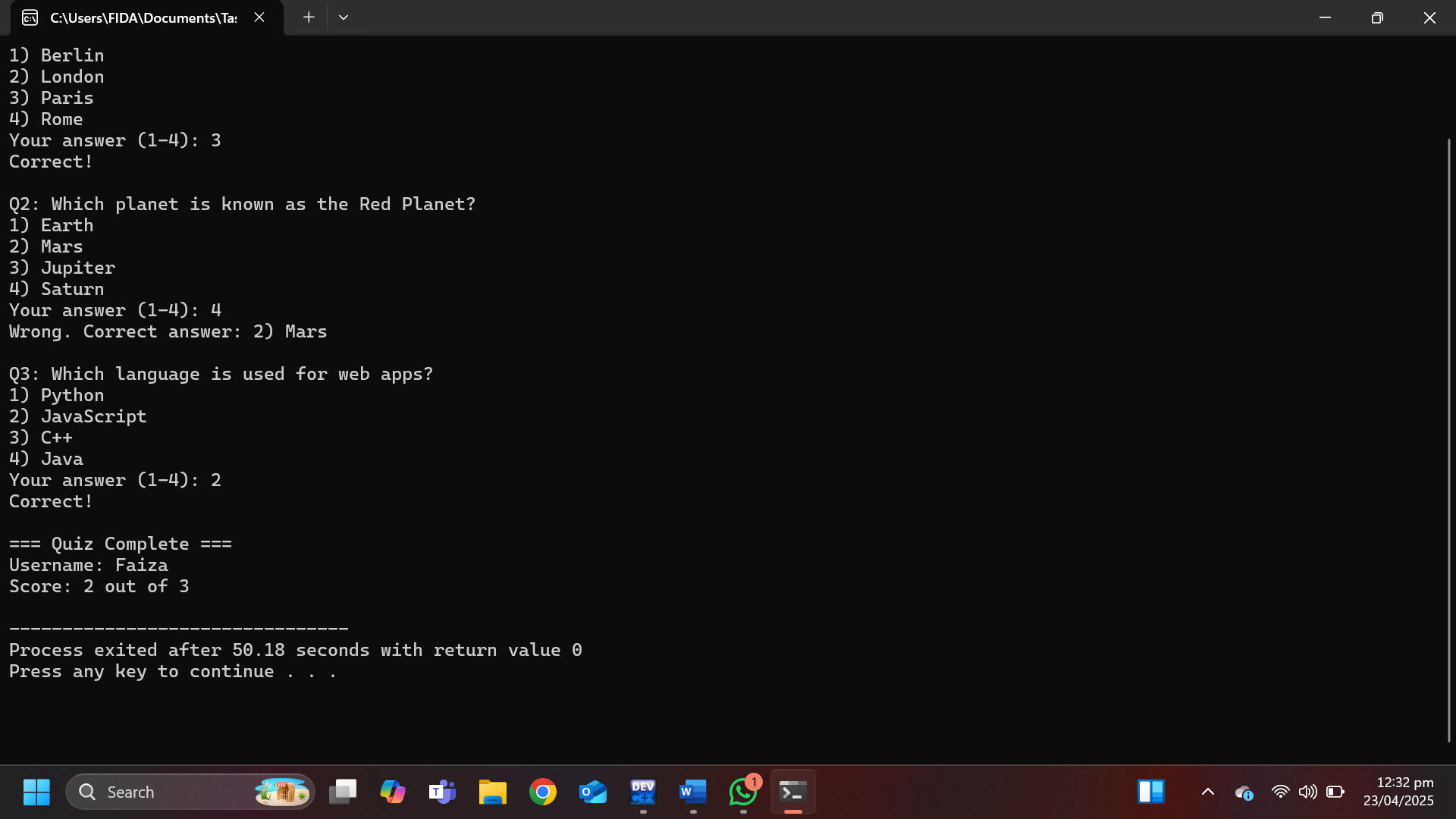
cout << "Username: " << username << endl;

cout << "Score: " << score << " out of " << quiz.size() << endl;

return 0;

}

# Console Output



SC++ PROGRAM FOR GAME OF LIFE SIMULATION

/\* Conway’s Game of Life with a fixed 5x5 grid and a predefined pattern\*/

#include <iostream>

#include <windows.h>

using namespace std;

const int GRID\_SIZE = 5;

void printGrid(int grid[GRID\_SIZE][GRID\_SIZE]) {

system("cls");

for (int i = 0; i < GRID\_SIZE; i++) {

for (int j = 0; j < GRID\_SIZE; j++)

cout << (grid[i][j] ? '\*' : '.') << " ";

cout << endl;

}

}

int countNeighbors(int grid[GRID\_SIZE][GRID\_SIZE], int x, int y) {

int count = 0;

for (int i = -1; i <= 1; i++)

for (int j = -1; j <= 1; j++) {

if (i == 0 && j == 0) continue;

int r = x + i, c = y + j;

if (r >= 0 && r < GRID\_SIZE && c >= 0 && c < GRID\_SIZE)

count += grid[r][c];

}

return count;

}

void nextGen(int grid[GRID\_SIZE][GRID\_SIZE]) {

int temp[SIZE][SIZE] = {0};

for (int i = 0; i < GRID\_SIZE; i++)

for (int j = 0; j < GRID\_SIZE; j++) {

int n = countNeighbors(grid, i, j);

if (grid[i][j] && (n == 2 || n == 3)) temp[i][j] = 1;

if (!grid[i][j] && n == 3) temp[i][j] = 1;

}

for (int i = 0; i < GRID\_SIZE; i++)

for (int j = 0; j < GRID\_SIZE; j++)

grid[i][j] = temp[i][j];

}

int main() {

int grid[GRID\_SIZE][GRID\_SIZE] = {0};

// Simple pattern

grid[1][2] = grid[2][2] = grid[3][2] = 1;

for (int gen = 0; gen < 10; gen++) {

printGrid(grid);

nextGen(grid);

Sleep(500); // half second delay

}

return 0;

}

/\*How it works:

Fixed 5x5 grid

Simple vertical line pattern

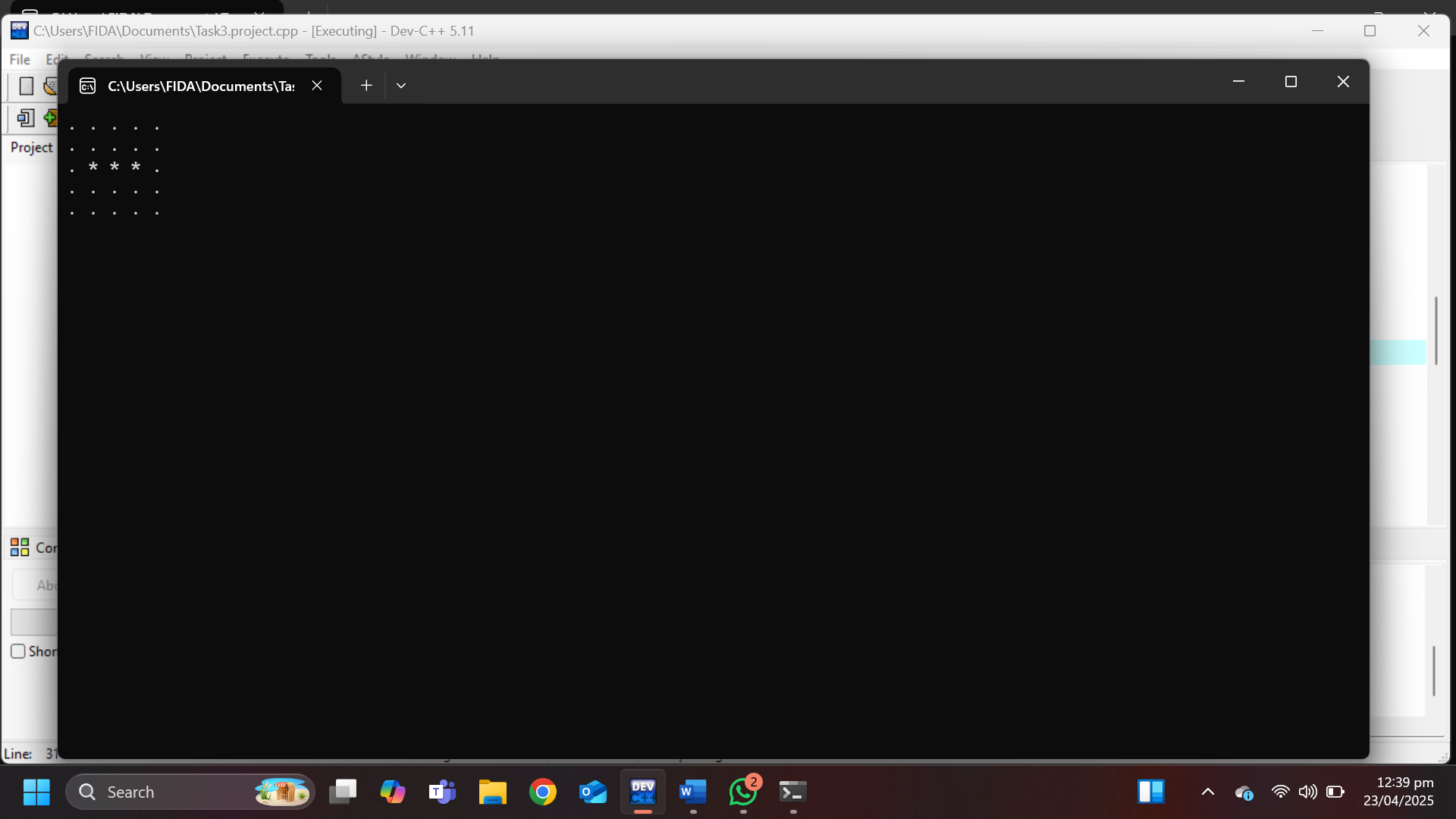
Updates for 10 generations

Uses \* for alive and . for dead

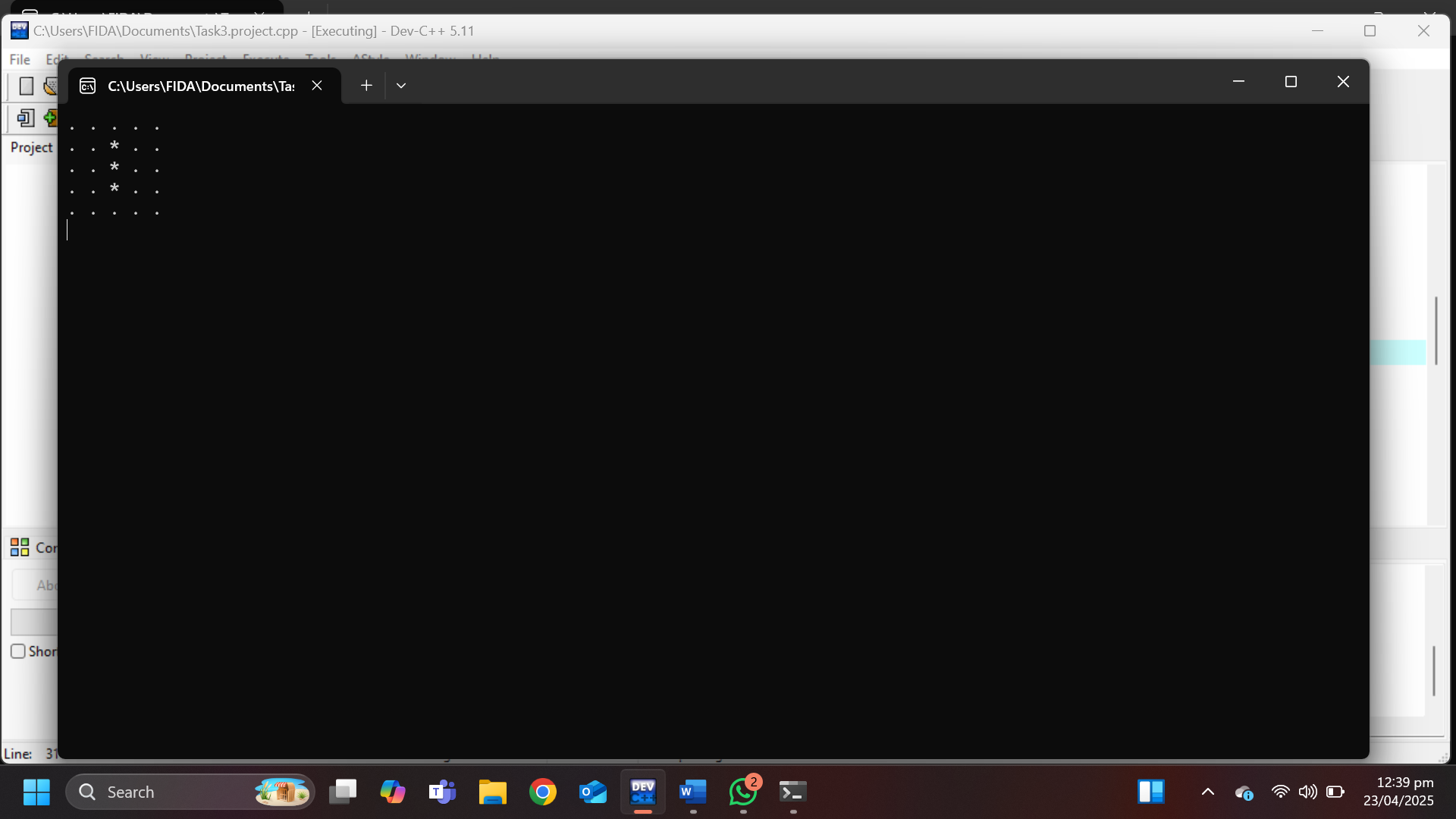
Ready to run on Windows.\*/

# Console Output

|  |
| --- |
|  |
| Initial Pattern(Generation 0): |
|  |
| Generation 1: |



Generation 2:



Generation 3:

