#include <stdio.h>

#include <string.h>

#include <stdlib.h>

#include <unistd.h>

#define ROWS 10

#define COLS 10

Void display\_matrix(int matrix[ROWS][COLS]) {

Printf(“\nCurrent State:\n”);

For (int I = 0; I < ROWS; i++) {

For (int j = 0; j < COLS; j++) {

Printf(“%3d “, matrix[i][j]);

}

Printf(“\n”);

}

}

Int count\_neighbors(int matrix[ROWS][COLS], int x, int y) {

Int count = 0;

For (int I = -1; I <= 1; i++) {

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

Int newX = x + I;

Int newY = y + j;

If ((I != 0 || j != 0) && newX >= 0 && newX < ROWS && newY >= 0 && newY < COLS) {

Count += matrix[newX][newY];

}

}

}

Return count;

}

Void update\_matrix(int matrix[ROWS][COLS]) {

Int newMatrix[ROWS][COLS] = {0};

For (int I = 0; I < ROWS; i++) {

For (int j = 0; j < COLS; j++) {

Int neighbors = count\_neighbors(matrix, I, j);

If (matrix[i][j] == 1) {

newMatrix[i][j] = (neighbors == 2 || neighbors == 3) ? 1 : 0;

} else {

newMatrix[i][j] = (neighbors == 3) ? 1 : 0;

}

}

}

Memcpy(matrix, newMatrix, sizeof(newMatrix));

}

Void get\_user\_input(int matrix[ROWS][COLS]) {

Char input[100];

Int position;

Printf(“Enter positions (1-100). Enter 0 to start simulation.\n”);

While (1) {

Display\_matrix(matrix);

Printf(“Enter positions: “);

Fgets(input, sizeof(input), stdin);

Char \*token = strtok(input, “ “);

While (token != NULL) {

Position = atoi(token);

If (position == 0) return;

If (position >= 1 && position <= 100) {

Int row = (position – 1) / COLS;

Int col = (position – 1) % COLS;

Matrix[row][col] = 1;

} else {

Printf(“Invalid input: %d (Use values 1-100)\n”, position);

}

Token = strtok(NULL, “ “);

}

}

}

Void run\_simulation(int matrix[ROWS][COLS], int iterations) {

For (int I = 0; I < iterations; i++) {

Printf(“\nIteration %d:\n”, I + 1);

Update\_matrix(matrix);

Display\_matrix(matrix);

Sleep(1);

}

Printf(“Simulation Complete!! \nThank you!\n”);

}

Int main() {

Int matrix[ROWS][COLS] = {0};

Int iterations;

Get\_user\_input(matrix);

Printf(“How many times should it run? “);

Scanf(“%d”, &iterations);

Getchar();

Run\_simulation(matrix, iterations);

Return 0;

}