CSE 102 ASSIGNMENT 7 REPORT

MUHAMMET FATİH ALBAYIN

THE CODE

```c
#include <stdio.h>
#include <string.h>
#include <strings.h>
#include <stdlib.h>
#include <time.h>

#define MAX_SIZE 10
#define MAX_MOVES 100

int move_rows[MAX_MOVES];               /* To store the move's rows     */
int move_cols[MAX_MOVES];               /* To store the move's columns  */
int move_start[MAX_MOVES];              /* For undo functionality       */
int move_top = -1;                      /* To keep the top move         */
int move_count = 0;                     /* To keep the move count       */
int total_cells, revealed_cells = 0;    /* To check if the game ended   */


int randomize_size();
int randomize_board(char board[][MAX_SIZE], int size);
int mine_amount(int size);
void print_board(char board[][MAX_SIZE], int size);
int update_board(char board[][MAX_SIZE], int row, int col, int mine_amount);
int save_move(int move_row, int move_col);
int check_move(int row, int col, int size);
int play_game();
void check_around(char board[][MAX_SIZE], int row, int col, int size);
void push_move(int row, int col);
void start_player_move();
void undo_move(char board[][MAX_SIZE]);
void remove_last_move();
void save_total_moves();


int main() {
    play_game();

    return 0;
}


int play_game() {                                   /* General function for
managing the whole game */
    char move[10];                                  /* The move
*/
    int row, col, size;                             /* Row, column input and size
of the grid       */
    int isFaulty = 1, gameon = 1;                   /* For the input and game
loops                 */
    int mines;                                      /* Amount of mines
*/

    size = randomize_size();                        /* Randomizes the grid size
*/
    char board[size][MAX_SIZE];
```

```c
    randomize_board(board, size);                   /* Randomizes the board */
    mines = mine_amount(size);                      /* Counts the mines in the board */

    total_cells = (size*size) - mines;              /* Amount of empty cells */


    printf("≡≡Welcome to the Minesweeper Game≡≡\n");

    FILE *file = fopen("moves.txt", "w");           /* Opens the file to store moves */

    fprintf(file, "---Game Moves---\n");

    while(gameon) {
        print_board(board, size);                   /* Prints the board before each move */

        while(isFaulty) {                           /* Checks the input and directs to the needed function  */
            printf("Enter the row and the column(or type undo): ");
            fgets(move, sizeof(move), stdin);

            move[strcspn(move, "\n")] = 0;

            if(sscanf(move, "%d %d", &row, &col) == 2 && row ⩽ size - 1 && col ⩽ size - 1 && row ⩾ 0 && col ⩾ 0) {
                int n = check_move(row, col, size);
                if (n == 1) {
                    printf("BOOM! You hit a mine. Game Over.\n");
                    gameon = 0;
                }
                else {
                    start_player_move();
                    check_around(board, row, col, size);
                    save_move(row, col);
                }

                isFaulty = 0;
            }
            else if(strcasecmp(move, "undo") == 0) {
                undo_move(board);
                remove_last_move();
                isFaulty = 0;
            }
            else {
                printf("Please enter a valid input!\n");
                isFaulty = 1;
            }
            if (revealed_cells == total_cells) {
                printf("Congratulations! You cleared the board without hitting a mine!\n");
                gameon = 0;
```

```
            }

        }
        isFaulty = 1;
    }
    if(!gameon)
        save_total_moves();
}

int randomize_size() {                              /* Randomize the board size
*/
    int size;
    srand(time(NULL));

    size = rand() % 8 + 2;                          /* Between 2 and 10
*/
    return size;
}


int randomize_board(char board[][MAX_SIZE], int size){  /* Randomly places mines
and prints it to the file */
    int i, j;
    srand(time(NULL));
    FILE *fptr = fopen("map.txt", "w");

    for(i = 0; i < size; i++){
        for (j = 0; j < size; j++) {
            if (rand() % 5 == 0){                       /* 20% chance of mine
*/
                board[i][j] = '*';
                fprintf(fptr, "%c ", board[i][j]);
                if (j == size - 1 && i != size - 1)
                    fprintf(fptr, "\n");
                }
            else {
                board[i][j] = '.';
                fprintf(fptr, "%c ", board[i][j]);
                if (j == size - 1 && i != size - 1)
                    fprintf(fptr, "\n");
            }
            board[i][j] = '#';                          /* Covers the board with
'#'                         */
        }
    }
    fclose(fptr);
}


void print_board(char board[][MAX_SIZE], int size){     /* Prints the board to the
console                    */
    int row, col;
    printf("  ");
    for (col = 0; col < size; col++) {
```

```
144         printf("%d ", col);
145     }
146     printf("\n");
147 
148     for (row = 0; row < size; row++) {
149         printf("%d ", row);
150         for (col = 0; col < size; col++) {
151             printf("%c ", board[row][col]);
152             if(col == size - 1) {
153                 printf("\n");
154             }
155         }
156     }
157 }
158 
159 
160 int update_board(char board[][MAX_SIZE], int row, int col, int mine_amount) {
161     board[row][col] = mine_amount + '0';                /* Places the number
    values to where it is told to  */
162 }
163 
164 
165 void check_around(char board[][MAX_SIZE], int row, int col, int size) {
166     int mines_around = 0, i, j;
167     int directions[8][2] = { {-1,-1}, {-1,0}, {-1,1}, {0,-1}, {0,1}, {1,-1},
    {1,0}, {1,1} };
168     char array[MAX_SIZE][MAX_SIZE];
169     int first_call = 1;
170 
171     if (first_call) {                                   /* Copies the board from
    the file at first run      */
172         FILE *fptr = fopen("map.txt", "r");
173         for (i = 0; i < size; i++) {
174             for (j = 0; j < size; j++) {
175                 fscanf(fptr, " %c", &array[i][j]);
176             }
177         }
178         fclose(fptr);
179         first_call = 0;
180     }
181 
182     if (board[row][col] ≠ '#')                          /* Doesn't repeat the
    process if it's already opened    */
183         return;
184 
185     revealed_cells++;                                   /* Increments the revealed
    cell count               */
186 
187     for(i = 0; i < 8; i++) {                            /* Checks the adjacent
    coordinates                   */
188         int adj_row = row + directions[i][0];
189         int adj_col = col + directions[i][1];
190 
191         if(adj_row ≥ 0 && adj_row < size && adj_col ≥ 0 && adj_col < size) {
```

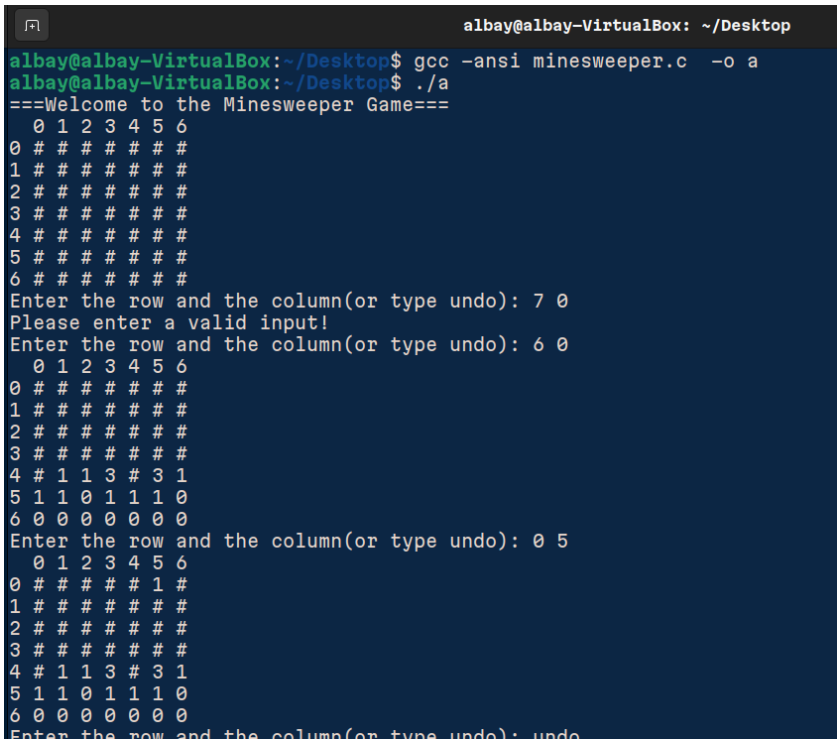
```
            if(array[adj_row][adj_col] == '*')
                mines_around++;
        }
    }

    update_board(board, row, col, mines_around);        /* Updates the board coordinate with the mine count */
    push_move(row, col);                                /* Pushes move to stack */

    if(mines_around == 0) {                             /* Recursively calls the function if there are no mines around  */
        for(i = 0; i < 8; i++) {
            int adj_row = row + directions[i][0];
            int adj_col = col + directions[i][1];

            if(adj_row >= 0 && adj_row < size && adj_col >= 0 && adj_col < size) {
                check_around(board, adj_row, adj_col, size);
            }
        }
    }
}



int check_move(int row, int col, int size) {
    char array[size][size];
    int i, j;

    FILE *fptr = fopen("map.txt", "r");                 /* Checks the move from the file                                */
    for (i = 0; i < size; i++) {
        for(j = 0; j < size; j++) {
            fscanf(fptr, " %c", &array[i][j]);
        }
    }
    if(array[row][col] == '*')
        return 1;
    else
        return 0;

}


int save_move(int move_row, int move_col) {             /* Saves the move to the file                       */
    FILE *file = fopen("moves.txt", "r");

    char lines[MAX_MOVES][50];
    int line_count = 0, i;

    while (fgets(lines[line_count], sizeof(lines[0]), file)) {      /* Counts the lines                     */
        if (strncmp(lines[line_count], "Total Moves", 11) != 0) {
```

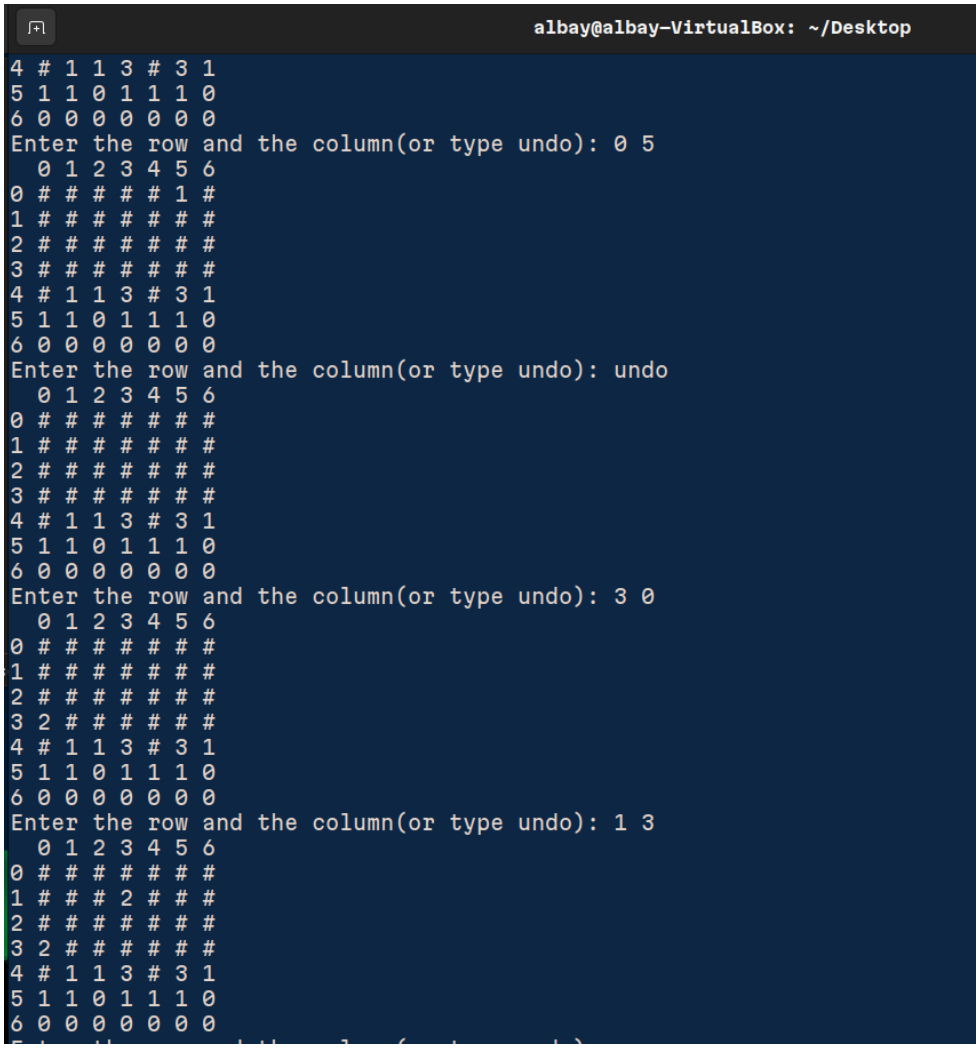
```
            line_count++;
        }
    }
    fclose(file);

    file = fopen("moves.txt", "w");

    fprintf(file, "---Game Moves---\n");                            /* Prints the move info to the file     */
    for (i = 1; i < line_count; i++) {
        fprintf(file, "%s", lines[i]);
    }

    move_count++;
    fprintf(file, "Move %d: (Row %d, Col %d)\n", move_count, move_row, move_col);

    fprintf(file, "Total Moves: %d\n", move_count);

    fclose(file);
    return 0;
}


int mine_amount(int size) {                             /* Counts the mines the board                      */
    int i, j, mines = 0;
    char check;
    FILE *file = fopen("map.txt", "r");

    for(i = 0; i < size; i++) {
        for(j = 0; j < size; j++){
            fscanf(file, " %c", &check);
            if(check == '*')
                ++mines;
        }
    }
    return mines;
}


void push_move(int row, int col) {                      /* Pushes the move to the stack with coordinate info    */
    if (move_top < MAX_MOVES - 1) {
        move_top++;
        move_rows[move_top] = row;
        move_cols[move_top] = col;
    }
}


void start_player_move() {                              /* Starts the move saving process           */
    move_start[move_count] = move_top + 1;
}
```

```c

void undo_move(char board[][MAX_SIZE]) {
    if (move_count == 0) {
        printf("There is no move to undo!\n");          /* Checks the move count */
        return;
    }

    int start = move_start[move_count - 1];

    while (move_top >= start) {                         /* Hides the opened cells */
        int row = move_rows[move_top];
        int col = move_cols[move_top];
        board[row][col] = '#';
        move_top--;
    }

    move_count--;                                       /* Decrements the move count               */
}


void remove_last_move() {                               /* Removes the last move from the stack     */
    FILE *file = fopen("moves.txt", "r");

    char lines[MAX_MOVES][50];
    int total_lines = 0, i;

    while (total_lines < MAX_MOVES && fgets(lines[total_lines], sizeof(lines[0]), file)) {
        total_lines++;                                  /* Count lines in the file */
    }
    fclose(file);

    if (total_lines < 3) {
        return;
    }

    int moveLineIndex = total_lines - 2;                /* Copies the file info excluding last two lines and paste it to the file   */

    for (i = moveLineIndex; i < total_lines - 1; i++) {
        strcpy(lines[i], lines[i + 1]);
    }
    total_lines--;

    sprintf(lines[total_lines - 1], "Total Moves: %d\n", move_count);

    file = fopen("moves.txt", "w");

```

```
337     for (i = 0; i < total_lines; i++) {
338         fprintf(file, "%s", lines[i]);
339     }
340     fclose(file);
341 }
342
343
344 void save_total_moves() {                               /* Prints the total moves
    to the file       */
345     FILE *file = fopen("moves.txt", "a");
346
347     fprintf(file, "\nTotal Moves: %d\n", move_count);
348     fclose(file);
349 }
```

OUTPUTS OF THE CODE





THE FILES' CONTENT