

# Report Quiz 2

---

## MINESWEEPER

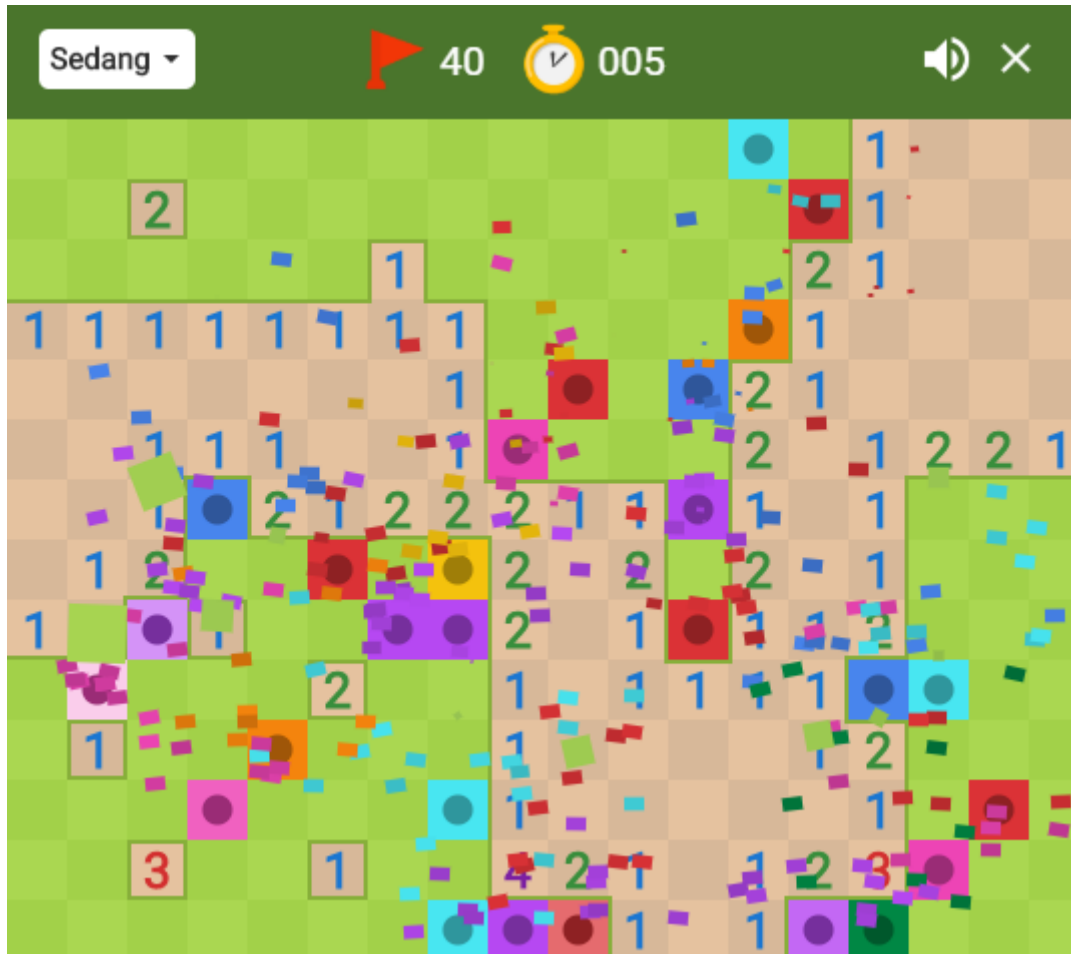
---

*By Group 5:*

---

1. *Sherly Rosa Anggraeni*  
05111740000018
  2. *Adzra Zaky Haura*  
05111740000037
  3. *Indira Nursyamsina Hazimi*  
05111740000082
-

## Concept of Minesweeper



**Minesweeper** is a game for one person. The aim of the game is to clear the land of mines without touching the 'bombs'.

### How to play :

- Understand the basic principles of the Minesweeper game. Each game session begins with a grid formed from plain plots. After one of the tiles is clicked, some tiles will disappear. There are also a number of plots that are still displayed plainly, as well as several plots that display numbers. Your task is to use the numbers shown to find out which blanks have mines, and which blanks are "safe" to click.
- Use the left and right buttons on the mouse. Mouse is the only device needed to play Minesweeper. The left button is used to click plots that do not contain mines, while the right button is used to mark plots that contain mines.
- The first plot clicked will never make a bomb. Once clicked, several tiles will open, while other tiles will display numbers.
- Recognize the meaning of the numbers that appear. The number on the plot refers to the number of bombs that currently touch the numbered plot. For example, if there are two plots next to each other, and one of the plots is marked with the number "1", one plot next to the plot has a bomb.

## Source Code

```
#include <stdio.h>
#include <strings.h>
#include <conio.h>
#include <time.h>

int main(){
    while(1){
        char grid[6][6]={}, field[6][6]={};
        char m[2], tmpstr[10];
        int gridn[6][6] = {0};
        int i, j, k, l;
        int x, y, xs, ys;
        int mine = 0;
        int nof;
        int a = 0;
        char menu;
        printf("\t\t\t\t\t\t\t\t\t\tMinesweeper 6x6\n\n");
        printf("\t\t\t\t\t\t\t\t\t\t1.Play Minesweeper\n\t\t\t\t\t\t\t\t\t\t2.Rules\n");
        menu = getch();
        if(menu == 50){
            system("cls");
            printf("This is the classic game of minesweeper written in
C.\n\nRules for minesweeper:\n");
            printf("1. You are in a mine field and you have to successfully
flag all the places which have a mine(%c). If you flag all the mines, You
win!\n",42);
            printf("2.In your first turn, you have to choose a starting
square. You can open a square by entering its row number(x) and column
number(y)\n(Note: row and column number starts from 1)\n");
            printf("3.There are three modes of operation:\n");
            printf("\t(i) open mode. type 'o' in mode option. This mode
lets the user to open a square\n");
            printf("\t(ii) flag mode. type 'f' in mode option. This mode
lets the user to flag a particular square. Flagged square is denoted by an
'F'\n");
            printf("\t(iii) remove flag mode. type 'r' in mode option. This
mode lets the user to remove a particular flag from a flagged square\n");
            printf("4.If you open a square with a mine, you lose\n");
            printf("5.If you open a square with a number written on it. The
number shows that how many mines are there in the adjacent 8 squares\n");
            printf("\nFor eg:\n%c %c %c\n\n%c 4 %c\n\n%c %c
%c",177,177,177,177,177,177,177,177,177);
            printf("\nHere 4 denoted that there are 4 mines in the
remaining uncovered squares");
            printf("\n\nPress enter to continue.....");
            fflush(stdin);
            gets(tmpstr);
        }

        system("cls");
        for(i = 0; i < 6; i++){
            for(j = 0; j < 6; j++){
                field[i][j] = 177;
            }
        }

        for(i = 0; i < 6; i++){
```

```

        for(j = 0; j < 30; j++){
            printf(" ");
        }
        for(j = 0; j < 6; j++){
            printf("%c  ", field[i][j]);
        }
        printf("\n\n");
    }

    int starting;
    starting:
    printf("\n Open the starting square.\n");
    printf("x: ");
    scanf("%d",&xs);
    printf("y: ");
    scanf("%d", &ys);
    if(xs>6 || ys>6 || xs<1 || ys<1){
        printf("Row or column not defined. Try again\n"); goto
starting;
    }

    xs = xs-1;
    ys = ys-1;
    srand(time(NULL));
    while(1){
        i=rand()%6;
        j=rand()%6;
        if(grid[i][j]!=42 && i!=xs && j!=ys) grid[i][j]=42;
        else continue;
        mine++;
        if(mine == 9) break;
    }

    nof = mine;
    for(i = 0; i < 6; i++){
        for(j = 0; j< 6; j++){
            if(grid[i][j] != 42){
                for(k = i-1; k <= i+1; k++){
                    for(l = j-1; l <= j+1; l++){
                        if(grid[k][l] == 42 && k >= 0 && l >= 0 && k
<=5 && l <= 5)
                            grid[i][j]++;
                    }
                }
                grid[i][j] = gridn[i][j] + 48;
            }
        }
    }

    for(i = xs-1; i <= xs+1; i++)
    {
        for(j = ys-1; j <= ys+1; j++)
        {
            if(grid[i][j] != 42){
                if (yf == 1 && j==-1) continue;
                else if (yf == 6 && j==6) continue;
                field[i][j] = grid[i][j];
                if(field[i][j] == 48){
                    field[i-1][j] = grid[i-1][j];
                    field[i+1][j] = grid[i+1][j];

```

```

        field[i][j+1] = grid[i][j+1];
        field[i][j-1] = grid[i][j-1];
        field[i-1][j+1] = grid[i-1][j+1];
        field[i-1][j-1] = grid[i-1][j-1];
        field[i+1][j+1] = grid[i+1][j+1];
        field[i+1][j-1] = grid[i+1][j-1];
    }
    printf("i: %d j: %d value: %c\n", i, j, field[i][j]);
}

x=xs;
y=ys;
while(1){
    system("cls");
    for(i = 0; i < 6; i++)
    {
        for(j = 0; j < 30; j++)
            printf(" ");
        for(j = 0; j < 6; j++)
            printf("%c ", field[i][j]);
        printf("\n\n");
    }

    printf("Number of leftover flags: %d\n", nof);
    printf("Game mode:\n");
    printf("\t'o' for opening a square.\n\t'f' for flaging a
square.\n\t'r' for removing square's flag.\n");

    if(grid[x][y] == 42 && strcmp(m, "o") == 0)
    {
        printf("\n\t\t\t\t\tYou opened a bomb!");
        printf("\n\t\t\t\t\tGAME OVER\n");
        break;
    }

    for(i = 0; i < 6; i++)
    {
        for(j = 0; j < 6; j++)
        {
            if(grid[i][j] == 42 && field[i][j] == 70)
                a++;
        }
    }

    if(a == mine)
    {
        printf("You win\n");
        break;
    }
    a = 0;
    printf("Mode chosen: ");
    scanf("%s", &m);
    printf("(x,y): ");
    scanf("%d %d", &x, &y);
    x = x-1;
    y = y-1;

    if(strcmp(m, "o") == 0) field[x][y] = grid[x][y];
    else if(strcmp(m, "f") == 0 && field[x][y] != 70 && field[x][y]
== -79){

```

```

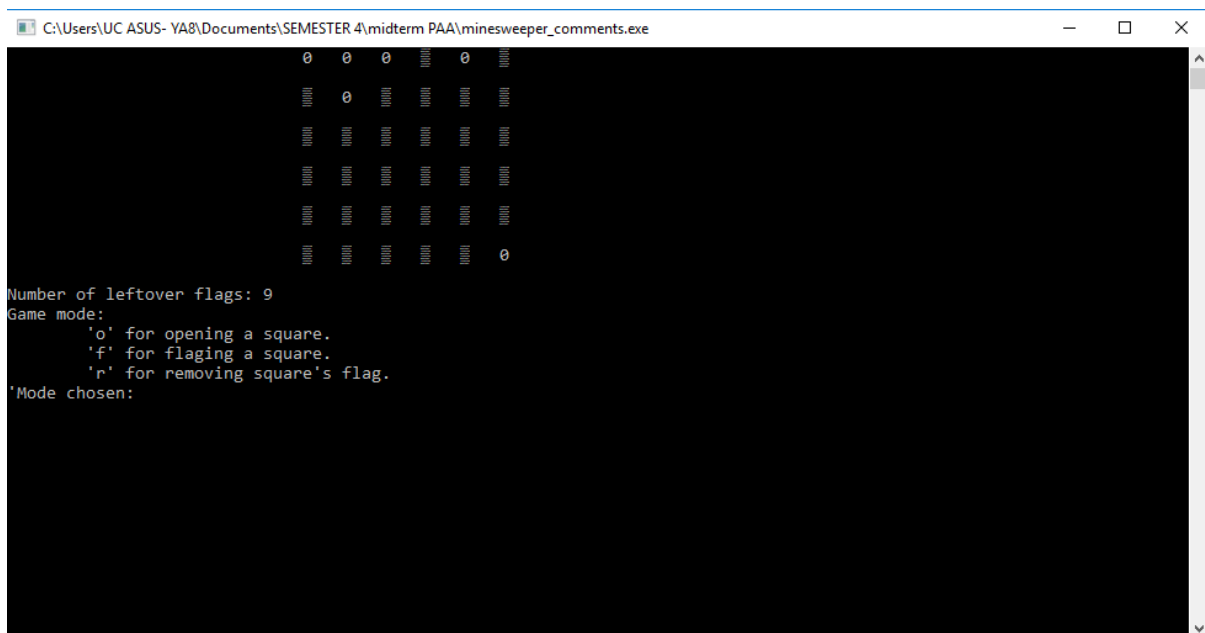
        field[x][y] = 70;
        nof--;
    }
    else if(strcmp(m, "r") == 0 && field[x][y] == 70){
        field[x][y] = 177;
        nof++;
    }
    if(grid[x][y] == 42 && strcmp(m, "o") == 0)
    {
        for(i = 0; i < 6; i++)
        {
            for(j = 0; j < 6; j++)
            {
                if(grid[i][j] == 42)
                    field[i][j] = grid[i][j];
            }
        }
    }

    //option to play the game again or exit
    int opt;
    printf("Play again?\n1. Yes\n2. No\n");
    scanf("%d", &opt);
    if(opt==2) break;
}
return 0;
}

```

## Bug no 1

So, when we try to play the game. The number to identify the field that already open only said 0, it supposed to be 0,1,2,etc.



## Source Code 2

```
#include <stdio.h>
#include <strings.h>
#include <conio.h>
#include <time.h>

//plan of the minesweeper's contruction
//not the finished code

int main(){
    while(1){
        char grid[6][6], field[6][6];

        char m[2], tmpstr[10];
        int gridn[6][6] = {0};
        int i, j, k, l;
        int x, y, xs, ys;
        int mine = 0;
        int nof;
        int a = 0;
        char menu;
        //the added part
        for(i = 0; i < 6; i++) {
            for(j = 0; j < 6; j++) {
                grid[i][j] = 0;
            }
        }
        printf("\t\t\t\t\t\t\t\tMinesweeper 6x6\n\n");
        printf("\t\t\t\t\t\t\t\t1.Play Minesweeper\n\t\t\t\t\t\t\t\t2.Rules\n");
        scanf("%c",&menu);
        if(menu == 50){
            system("cls");
```

```

        printf("This is the classic game of minesweeper written in
C.\n\nRules for minesweeper:\n");
        printf("1. You are in a mine field and you have to successfully
flag all the places which have a mine(%c). If you flag all the mines, You
win!\n",42);
        printf("2.In your first turn, you have to choose a starting
square. You can open a square by entering its row number(x) and column
number(y)\n(Note: row and column number starts from 1)\n");
        printf("3.There are three modes of operation:\n");
        printf("\t(i) open mode. type 'o' in mode option. This mode
lets the user to open a square\n");
        printf("\t(ii) flag mode. type 'f' in mode option. This mode
lets the user to flag a particular square. Flagged square is denoted by an
'F'\n");
        printf("\t(iii) remove flag mode. type 'r' in mode option. This
mode lets the user to remove a particular flag from a flagged square\n");
        printf("4.If you open a square with a mine, you lose\n");
        printf("5.If you open a square with a number written on it. The
number shows that how many mines are there in the adjacent 8 squares\n");
        printf("\nFor eg:\n%c %c %c\n\n%c 4 %c\n\n%c %c
%c",177,177,177,177,177,177,177,177);
        printf("\nHere 4 denoted that there are 4 mines in the
remaining uncovered squares");
        printf("\n\nPress enter to continue.....");
        fflush(stdin);
        gets(tmpstr);
    }

    system("cls");
    for(i = 0; i < 6; i++){
        for(j = 0; j < 6; j++){
            field[i][j] = 177;
        }
    }

    for(i = 0; i < 6; i++){
        for(j = 0; j < 30; j++){
            printf(" ");
        }

        for(j = 0; j < 6; j++){
            printf("%c  ", field[i][j]);
        }
        printf("\n\n");
    }
    //the part that we change
    for(;;){
        printf("\n Open the starting square.\n");
        printf("x: ");
        scanf("%d",&xs);
        printf("y: ");
        scanf("%d", &ys);
        if(xs>6 || ys>6 || xs<1 || ys<1){
            printf("Row or column not defined. Try again\n"); continue;
        }
        else break;
    }

    xs = xs-1;
    ys = ys-1;
    srand(time(NULL));

```



```

while(1){
    i=rand()%6;
    j=rand()%6;
    if(grid[i][j]!=42 && i!=xs && j!=ys) grid[i][j]=42;
    else continue;
    mine++;
    if(mine == 9) break;
}

nof = mine;
for(i = 0; i < 6; i++){
    for(j = 0; j < 6; j++){
        if(grid[i][j] != 42){
            for(k = i-1; k <= i+1; k++){
                for(l = j-1; l <= j+1; l++){
                    if(grid[k][l] == 42 && k >= 0 && l >= 0 && k
<=5 && l <= 5)
                        gridn[i][j]++;
                }
            }
            grid[i][j] = gridn[i][j] + 48;
        }
    }
}

for(i = xs-1; i <= xs+1; i++)
{
    for(j = ys-1; j <= ys+1; j++)
    {
        if(grid[i][j] != 42)field[i][j] = grid[i][j];
    }
}

x=x;
y=y;
while(1){
    system("cls");
    for(i = 0; i < 6; i++)
    {
        for(j = 0; j < 30; j++)
            printf(" ");
        for(j = 0; j < 6; j++)
            printf("%c  ", field[i][j]);
        printf("\n\n");
    }
    printf("Number of leftover flags: %d\n", nof);
    printf("Game mode:\n");
    printf("\t'o' for opening a square.\n\t'f' for flaging a
square.\n\t'r' for removing square's flag.\n");
    if(grid[x][y] == 42 && strcmp(m, "o") == 0)
    {
        printf("\n\t\t\t\t\tYou opened a bomb!");
        printf("\n\t\t\t\t\tGAME OVER\n");
        break;
    }
    for(i = 0; i < 6; i++)
    {
        for(j = 0; j < 6; j++)
        {
            if(grid[i][j] == 42 && field[i][j] == 70)

```

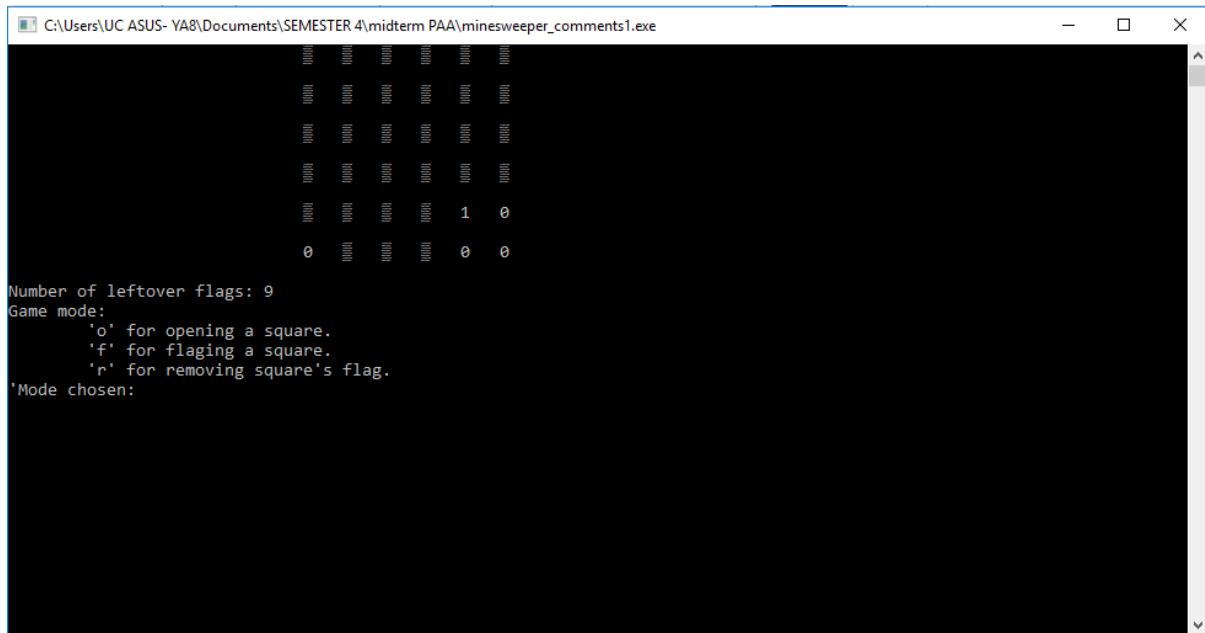
```

        a++;
    }
}
if(a == mine)
{
    printf("You win\n");
    break;
}
//the part that we change
a = 0;
printf("Mode chosen: ");
scanf("%s", &m);
printf("x: ");
scanf("%d",&x);
printf("y: ");
scanf("%d",&y);
x = x-1;
y = y-1;
printf("%d\n", field[x][y]);
if(strcmp(m, "o") == 0) field[x][y] = grid[x][y];
else if(strcmp(m, "f") == 0 && field[x][y] != 70 && field[x][y]
== -79){
    field[x][y] = 70;
    nof--;
}
else if(strcmp(m, "r") == 0 && field[x][y] == 70){
    field[x][y] = 177;
    nof++;
}
if(grid[x][y] == 42 && strcmp(m, "o") == 0)
{
    for(i = 0; i < 6; i++)
    {
        for(j = 0; j < 6; j++)
        {
            if(grid[i][j] == 42)
                field[i][j] = grid[i][j];
        }
    }
}
int opt;
printf("Play again?\n1. Yes\n2. No\n");
scanf("%d", &opt);
if(opt==2) break;
}
return 0;}

```

## Bug no 2

If we choose coordinate as example (6,6), the other side field will open too.



### Source Code 3

```
#include <stdio.h>
#include <strings.h>
#include <conio.h>
#include <time.h>

int main(){
    while(1){
        char grid[6][6], field[6][6];
        char m[2], tmpstr[10];
        int gridn[6][6] = {0};
        int i, j, k, l;
        int x, y, xs, ys;
        int mine = 0;
        int nof;
        int a = 0;
        char menu;
        for(i = 0; i < 6; i++) {
            for(j = 0; j < 6; j++) {
                grid[i][j] = 0;
            }
        }
        printf("\t\t\t\t\t\t\t\tMinesweeper 6x6\n\n");
        printf("\t\t\t\t\t\t\t\t1.Play Minesweeper\n\t\t\t\t\t\t\t\t2.Rules\n");
        scanf("%c",&menu);
        if(menu == 50){
            system("cls");
            printf("This is the classic game of minesweeper written in  
C.\n\nRules for minesweeper:\n");
            printf("1. You are in a mine field and you have to successfully  
flag all the places which have a mine(%c). If you flag all the mines, You  
win!\n",42);
        }
    }
}
```

```

        printf("2.In your first turn, you have to choose a starting
square. You can open a square by entering its row number(x) and column
number(y)\n(Note: row and column number starts from 1)\n");
        printf("3.There are three modes of operation:\n");
        printf("\t(i) open mode. type 'o' in mode option. This mode
lets the user to open a square\n");
        printf("\t(ii) flag mode. type 'f' in mode option. This mode
lets the user to flag a particular square. Flagged square is denoted by an
'F'\n");
        printf("\t(iii) remove flag mode. type 'r' in mode option. This
mode lets the user to remove a particular flag from a flagged square\n");
        printf("4.If you open a square with a mine, you lose\n");
        printf("5.If you open a square with a number written on it. The
number shows that how many mines are there in the adjacent 8 squares\n");
        printf("\nFor eg:\n%c %c %c\n\n%c 4 %c\n\n%c %c
%c",177,177,177,177,177,177,177,177);
        printf("\nHere 4 denoted that there are 4 mines in the
remaining uncovered squares");
        printf("\n\nPress enter to continue.....");
        fflush(stdin);
        gets(tmpstr);
    }

    system("cls");
    for(i = 0; i < 6; i++){
        for(j = 0; j < 6; j++){
            field[i][j] = 177;
        }
    }
    for(i = 0; i < 6; i++){
        for(j = 0; j < 30; j++){
            printf(" ");
        }
        for(j = 0; j < 6; j++){
            printf("%c  ", field[i][j]);
        }
        printf("\n\n");
    }

    for(;;){
        printf("\n Open the starting square.\n");
        printf("x: ");
        scanf("%d",&xs);
        printf("y: ");
        scanf("%d", &ys);
        if(xs>6 || ys>6 || xs<1 || ys<1){
            printf("Row or column not defined. Try again\n"); continue;
        }
        else break;
    }

    xs = xs-1;
    ys = ys-1;
    srand(time(NULL));
    while(1){
        i=rand()%6;
        j=rand()%6;
        if(grid[i][j]!=42 && i!=xs && j!=ys) grid[i][j]=42;
        else continue;
        mine++;
        if(mine == 9) break;
    }

```

```

}

nof = mine;
for(i = 0; i < 6; i++){
    for(j = 0; j< 6; j++){
        if(grid[i][j] != 42){
            for(k = i-1; k <= i+1; k++){
                for(l = j-1; l <= j+1; l++){
                    if(grid[k][l] == 42 && k >= 0 && l >= 0 && k
<=5 && l <= 5)
                        gridn[i][j]++;
                }
            }
            grid[i][j] = gridn[i][j] + 48;
        }
    }
}

int yf;
//the part that we change
for(i = xs-1; i <= xs+1; i++)
{
    for(j = ys-1; j <= ys+1; j++)
    {
        if(grid[i][j] != 42){

            if (yf == 1 && j==-1) continue;
            else if (yf == 6 && j==6) continue;
            field[i][j] = grid[i][j];
        }
    }
}

x=xS;
y=yS;
while(1){

    system("cls");
    for(i = 0; i < 6; i++)
    {
        for(j = 0; j < 30; j++)
            printf(" ");
        for(j = 0; j < 6; j++)
            printf("%c   ", field[i][j]);
        printf("\n\n");
    }

    printf("Number of leftover flags: %d\n", nof);
    printf("Game mode:\n");
    printf("\t'o' for opening a square.\n\t'f' for flaging a
square.\n\t'r' for removing square's flag.\n");

    if(grid[x][y] == 42 && strcmp(m, "o") == 0)
    {
        printf("\n\t\t\t\t\tYou opened a bomb!");
        printf("\n\t\t\t\t\tGAME OVER\n");
        break;
    }
    for(i = 0; i < 6; i++)
    {
        for(j = 0; j < 6; j++)

```

```

        {
            if(grid[i][j] == 42 && field[i][j] == 70)
                a++;
        }
    }
    if(a == mine)
    {
        printf("You win\n");
        break;
    }
    a = 0;
    printf("Mode chosen: ");
    scanf("%s", &m);
    printf("x: ");
    scanf("%d", &x);
    printf("y: ");
    scanf("%d", &y);
    x = x-1;
    y = y-1;
    printf("%d\n", field[x][y]);
    if(strcmp(m, "o") == 0) field[x][y] = grid[x][y];
    else if(strcmp(m, "f") == 0 && field[x][y] != 70 && field[x][y]
== -79){
        field[x][y] = 70;
        nof--;
    }
    else if(strcmp(m, "r") == 0 && field[x][y] == 70){
        field[x][y] = 177;
        nof++;
    }
    if(grid[x][y] == 42 && strcmp(m, "o") == 0)
    {
        for(i = 0; i < 6; i++)
        {
            for(j = 0; j < 6; j++)
            {
                if(grid[i][j] == 42)
                    field[i][j] = grid[i][j];
            }
        }
    }
    }
    int opt;
    printf("Play again?\n1. Yes\n2. No\n");
    scanf("%d", &opt);
    if(opt==2) break;
}
return 0;
}

```

## Bug no 3

So apparently we dont have any dfs implementation in our source code.

## Source Code 4

```
#include <stdio.h>
#include <strings.h>
#include <conio.h>
#include <time.h>

int main(){
    while(1){
        char grid[6][6], field[6][6];
        char m[2], tmpstr[10];
        int gridn[6][6] = {0};
        int i, j, k, l;
        int x, y, xs, ys;
        int mine = 0;
        int nof;
        int a = 0;
        char menu;
        for(i = 0; i < 6; i++) {
            for(j = 0; j < 6; j++) {
                grid[i][j] = 0;
            }
        }
        printf("\t\t\t\t\t\t\tMinesweeper 6x6\n\n");
        printf("\t\t\t\t\t\t\t1.Play Minesweeper\n\t\t\t\t\t\t\t2.Rules\n");
        scanf("%c",&menu);
        if(menu == 50){
            system("cls");
            printf("This is the classic game of minesweeper written in C.\n\nRules for minesweeper:\n");
            printf("1. You are in a mine field and you have to successfully flag all the places which have a mine(%c). If you flag all the mines, You win!\n",42);
            printf("2.In your first turn, you have to choose a starting square. You can open a square by entering its row number(x) and column number(y)\n(Note: row and column number starts from 1)\n");
            printf("3.There are three modes of operation:\n");
            printf("\t(i) open mode. type 'o' in mode option. This mode lets the user to open a square\n");
            printf("\t(ii) flag mode. type 'f' in mode option. This mode lets the user to flag a particular square. Flagged square is denoted by an 'F'\n");
            printf("\t(iii) remove flag mode. type 'r' in mode option. This mode lets the user to remove a particular flag from a flagged square\n");
            printf("4.If you open a square with a mine, you lose\n");
            printf("5.If you open a square with a number written on it. The number shows that how many mines are there in the adjacent 8 squares\n");
            printf("\nFor eg:\n%c %c %c\n\n%c 4 %c\n\n%c %c %c",177,177,177,177,177,177,177,177);
            printf("\nHere 4 denoted that there are 4 mines in the remaining uncovered squares");
            printf("\n\nPress enter to continue.....");
            fflush(stdin);
            gets(tmpstr);
        }

        system("cls");
```

```

    for(i = 0; i < 6; i++){
        for(j = 0; j < 6; j++){
            field[i][j] = 177;
        }
    }
    for(i = 0; i < 6; i++){
        for(j = 0; j < 30; j++){
            printf(" ");
        }
        for(j = 0; j < 6; j++){
            printf("%c  ", field[i][j]);
        }
        printf("\n\n");
    }

    for(;;){
        printf("\n Open the starting square.\n");
        printf("x: ");
        scanf("%d",&xs);
        printf("y: ");
        scanf("%d", &ys);
        if(xs>6 || ys>6 || xs<1 || ys<1){
            printf("Row or column not defined. Try again\n"); continue;
        }
        else break;
    }

    xs = xs-1;
    ys = ys-1;
    srand(time(NULL));
    while(1){
        i=rand()%6;
        j=rand()%6;
        if(grid[i][j]!=42 && i!=xs && j!=ys) grid[i][j]=42;
        else continue;
        mine++;
        if(mine == 9) break;
    }

    nof = mine;
    for(i = 0; i < 6; i++){
        for(j = 0; j < 6; j++){
            if(grid[i][j] != 42){
                for(k = i-1; k <= i+1; k++){
                    for(l = j-1; l <= j+1; l++){
                        if(grid[k][l] == 42 && k >= 0 && l >= 0 && k
<=5 && l <= 5)
                            gridn[i][j]++;
                    }
                }
                grid[i][j] = gridn[i][j] + 48;
            }
        }
    }
    int yf;
    //the part that we change
    for(i = xs-1; i <= xs+1; i++)
    {
        for(j = ys-1; j <= ys+1; j++)
        {
            if(grid[i][j] != 42){

```



```

        if (yf == 1 && j==1) continue;
        else if (yf == 6 && j==6) continue;
        field[i][j] = grid[i][j];
        if(field[i][j] == 48){
            field[i-1][j] = grid[i-1][j];
            field[i+1][j] = grid[i+1][j];
            field[i][j+1] = grid[i][j+1];
            field[i][j-1] = grid[i][j-1];
            field[i-1][j+1] = grid[i-1][j+1];
            field[i-1][j-1] = grid[i-1][j-1];
            field[i+1][j+1] = grid[i+1][j+1];
            field[i+1][j-1] = grid[i+1][j-1];
        }
        printf("i: %d j: %d value: %c\n", i, j, field[i][j]);
    }

}

x=xs;
y=ys;
while(1){

    system("cls");
    for(i = 0; i < 6; i++)
    {
        for(j = 0; j < 30; j++)
            printf(" ");
        for(j = 0; j < 6; j++)
            printf("%c ", field[i][j]);
        printf("\n\n");
    }

    printf("Number of leftover flags: %d\n", nof);
    printf("Game mode:\n");
    printf("\t'o' for opening a square.\n\t'f' for flagging a
square.\n\t'r' for removing square's flag.\n");

    if(grid[x][y] == 42 && strcmp(m, "o") == 0)
    {
        printf("\n\t\t\t\t\tYou opened a bomb!");
        printf("\n\t\t\t\t\tGAME OVER\n");
        break;
    }
    for(i = 0; i < 6; i++)
    {
        for(j = 0; j < 6; j++)
        {
            if(grid[i][j] == 42 && field[i][j] == 70)
                a++;
        }
    }
    if(a == mine)
    {
        printf("You win\n");
        break;
    }
    a = 0;
    printf("Mode chosen: ");
    scanf("%s", &m);
    printf("x: ");

```

```

scanf("%d",&x);
printf("y: ");
scanf("%d",&y);
x = x-1;
y = y-1;
printf("%d\n", field[x][y]);
if(strcmp(m, "o") == 0) field[x][y] = grid[x][y];
else if(strcmp(m, "f") == 0 && field[x][y] != 70 && field[x][y]
== -79){
    field[x][y] = 70;
    nof--;
}
else if(strcmp(m, "r") == 0 && field[x][y] == 70){
    field[x][y] = 177;
    nof++;
}
if(grid[x][y] == 42 && strcmp(m, "o") == 0)
{
    for(i = 0; i < 6; i++)
    {
        for(j = 0; j < 6; j++)
        {
            if(grid[i][j] == 42)
                field[i][j] = grid[i][j];
        }
    }
}
int opt;
printf("Play again?\n1. Yes\n2. No\n");
scanf("%d", &opt);
if(opt==2) break;
}
return 0;
}

```

## Improvement

We improve our source code to be ‘a little bit OOP’.

- Board to define the fields

### Board.cpp

```
#include "pch.h"
#include "Board.h"

using namespace std;

Board::Board()
{
    for (i = 0; i < 6; i++) {
        for (j = 0; j < 6; j++) {
            grid[i][j] = 0;
        }
    }
    for(i = 0; i < 6; i++){
        for(j = 0; j < 6; j++){
            field[i][j] = SQUARE;
        }
    }
}

Board::~Board()
{
}

void Board::PrintFirstBoard()
{
    system("CLS");
    for (i = 0; i < 6; i++) {
        for (j = 0; j < 30; j++) {
            printf(" ");
        }
        for (j = 0; j < 6; j++) {
            printf("%c  ", field[i][j]);
        }
        printf("\n\n");
    }
}

void Board::FirstStep()
{
    for (;;) {
        printf("\n Open the starting square.\n");
        printf("x: ");
        scanf_s("%d", &ys);
        printf("y: ");
        scanf_s("%d", &xs);
        if (xs > 6 || ys > 6 || xs < 1 || ys < 1) {
            printf("Row or column not defined. Try again\n"); continue;
        }
        else break;
    }
    xs = xs - 1;
    ys = ys - 1;
    srand(time(NULL));
}
```

```

}

int Board::SpreadTheMine()
{
    while (1) {
        i = rand() % 6;
        j = rand() % 6;
        if (grid[i][j] != 42 && i != xs && j != ys) grid[i][j] = 42;
        else continue;
        mine++;
        if (mine == 9) return 1;
    }
}

void Board::MakeHints()
{
    nof = mine;
    for (i = 0; i < 6; i++) {
        for (j = 0; j < 6; j++) {
            if (grid[i][j] != 42) {
                for (k = i - 1; k <= i + 1; k++) {
                    for (l = j - 1; l <= j + 1; l++) {
                        if (grid[k][l] == 42 && k >= 0 && l >= 0 && k <= 5
&& l <= 5)
                            gridn[i][j]++;
                    }
                }
                grid[i][j] = gridn[i][j] + 48;
            }
        }
    }
}

int yf;
void Board::MakeMap()
{
    for (i = xs - 1; i <= xs + 1; i++)
    {
        for (j = ys - 1; j <= ys + 1; j++)
        {
            if (grid[i][j] != 42) {
                if (yf == 1 && j == -1) continue;
                else if (yf == 6 && j == 6) continue;
                field[i][j] = grid[i][j];
                if (field[i][j] == 48) {
                    field[i-1][j] = grid[i-1][j];
                    field[i+1][j] = grid[i+1][j];
                    field[i][j+1] = grid[i][j+1];
                    field[i][j-1] = grid[i][j-1];
                    field[i-1][j+1] = grid[i-1][j+1];
                    field[i-1][j-1] = grid[i-1][j-1];
                    field[i+1][j+1] = grid[i+1][j+1];
                    field[i+1][j-1] = grid[i+1][j-1];
                }
                printf("i: %d j: %d value: %c\n", i, j, field[i][j]);
            }
        }
    }

    x = xs;
    y = ys;
}

```

```
void Board::PrintBoard()
{
    system("cls");
    for (i = 0; i < 6; i++)
    {
        for (j = 0; j < 30; j++)
            printf(" ");
        for (j = 0; j < 6; j++)
            printf("%c ", field[i][j]);
        printf("\n\n");
    }
}

void Board::Details()
{
    printf("Number of leftover flags: %d\n", nof);
    printf("Game mode:\n");
    printf("\t'o' for opening a square.\n\t'f' for flagging a square.\n\t'r'  
for removing square's flag.\n");
}

int Board::ChoiceCheck()
{
    if (grid[x][y] == 42 && strcmp(m, "o") == 0)
    {
        printf("\n\t\t\t\t\tYou opened a bomb!");
        printf("\n\t\t\t\t\tGAME OVER\n");
        return 1;
    }
    for (i = 0; i < 6; i++)
    {
        for (j = 0; j < 6; j++)
        {
            if (grid[i][j] == 42 && field[i][j] == 70)
                a++;
        }
    }
    if (a == mine)
    {
        printf("You win\n");
        return 1;
    }
}

void Board::ChoiceGetter()
{
    a = 0;
    printf("Mode chosen: ");
    cin >> m;
    printf("x: ");
    scanf_s("%d", &y);
    printf("y: ");
    scanf_s("%d", &x);
    x = x - 1;
    y = y - 1;
    printf("%d\n", field[x][y]);
}

void Board::OpenSquare()
{

```

```

        field[x][y] = grid[x][y];
    }

void Board::FlagSquare()
{
    if(field[x][y] != 70 && field[x][y] == -79)
    {
        field[x][y] = 70;
        nof--;
    }
}

void Board::RemoveFlag()
{
    if (field[x][y] == 70)
    {
        field[x][y] = SQUARE;
        nof++;
    }
}

void Board::RevealAllBombs()
{
    if (grid[x][y] == 42 && strcmp(m, "o") == 0)
    {
        for (i = 0; i < 6; i++)
        {
            for (j = 0; j < 6; j++)
            {
                if (grid[i][j] == 42)
                    field[i][j] = grid[i][j];
            }
        }
    }
}

```

## Board.h

```

#pragma once
#include <iostream>
#include <string.h>
#include <conio.h>
#include <time.h>

#define SQUARE 177;

class Board
{
public:
    Board();
    ~Board();
    char grid[6][6], field[6][6];
    char m[2], tmpstr[10];
    int gridn[6][6] = { 0 };
    int i, j, k, l;
    int x, y, xs, ys;
    int mine = 0;
    int nof;
    int a = 0;

    void PrintFirstBoard();
    void FirstStep();

```



```

        printf("\nHere 4 denoted that there are 4 mines in the remaining
uncovered squares\n");
        system("PAUSE");
    }

int Menu::Finish()
{
    int opt;
    printf("Play again?\n1. Yes\n2. No\n");
    scanf_s("%d", &opt);
    if (opt == 1) { return 0; }
    else if (opt == 2) { return 1; }
}

void Menu::Unknown()
{
    printf("Sorry, unknown command\nPlease try again\n");
    Sleep(2500);
}

```

## Menu.h

```

#pragma once
#include <windows.h>
#include <string.h>
#include <iostream>
#define SQUARE 177

class Menu
{
public:
    Menu();
    ~Menu();
    int intmenu;
    void Start();
    void Rules();
    int Finish();
    void Unknown();
};

```

- We make file because we need pre-compiled header

## Pch.cpp

```

// pch.cpp: source file corresponding to pre-compiled header; necessary for
compilation to succeed

```

```

#include "pch.h"

```

```

// In general, ignore this file, but keep it around if you are using pre-
compiled headers.

```

## Pch.h

```

// Tips for Getting Started:
// 1. Use the Solution Explorer window to add/manage files
// 2. Use the Team Explorer window to connect to source control
// 3. Use the Output window to see build output and other messages
// 4. Use the Error List window to view errors
// 5. Go to Project > Add New Item to create new code files, or Project >
Add Existing Item to add existing code files to the project

```



```
// 6. In the future, to open this project again, go to File > Open >
Project and select the .sln file
```

```
#ifndef PCH_H
#define PCH_H
```

```
// TODO: add headers that you want to pre-compile here
```

```
#endif //PCH_H
```

- We make file to do the main function of this game

### **Minesweeper.cpp**

```
#include "pch.h"
#include "Menu.h"
#include "Board.h"
#include <iostream>
```

```
using namespace std;
```

```
int main()
{
    while (1)
    {
        //std::cout << "Hello World!\n";
        Menu ui;
        ui.Start();
        if (ui.intmenu == 1)
        {
            printf("Mainnnnn\n");
            Board minefield;
            minefield.PrintFirstBoard();
            minefield.FirstStep();
            while (1)
            {
                if(minefield.SpreadTheMine()) break;
            }
            minefield.MakeHints();
            minefield.MakeMap();
            while (1)
            {
                minefield.PrintBoard();
                OutputDebugString(L"Masuk\n");
                minefield.Details();
                OutputDebugString(L"Masuk2\n");
                if(minefield.ChoiceCheck()==1) break;
                OutputDebugString(L"Masuk3\n");
                minefield.ChoiceGetter();
                OutputDebugString(L"Masuk4\n");
                if (strcmp(minefield.m, "o") == 0) minefield.OpenSquare();
                else if (strcmp(minefield.m, "f") == 0)
                    minefield.FlagSquare();
                else if (strcmp(minefield.m, "r") == 0)
                    minefield.RemoveFlag();
                OutputDebugString(L"Masuk5\n");
                minefield.RevealAllBombs();
                OutputDebugString(L"Masuk6\n");
            }
            if (ui.Finish()) break;
        }
        if (ui.intmenu == 2)
```

```

        {
            ui.Rules();
        }
        if (ui.intmenu == 3)
        {
            break;
        }
        // else
        // {
        //     ui.Unknown();
        //     system("cls");
        //     ui.Start();
        // }
        //system("Pause");
    }
    return 0;
}

```

### Minesweeper.vcxproj

```

<?xml version="1.0" encoding="utf-8"?>
<Project DefaultTargets="Build" ToolsVersion="15.0"
xmlns="http://schemas.microsoft.com/developer/msbuild/2003">
  <ItemGroup Label="ProjectConfigurations">
    <ProjectConfiguration Include="Debug|Win32">
      <Configuration>Debug</Configuration>
      <Platform>Win32</Platform>
    </ProjectConfiguration>
    <ProjectConfiguration Include="Release|Win32">
      <Configuration>Release</Configuration>
      <Platform>Win32</Platform>
    </ProjectConfiguration>
    <ProjectConfiguration Include="Debug|x64">
      <Configuration>Debug</Configuration>
      <Platform>x64</Platform>
    </ProjectConfiguration>
    <ProjectConfiguration Include="Release|x64">
      <Configuration>Release</Configuration>
      <Platform>x64</Platform>
    </ProjectConfiguration>
  </ItemGroup>
  <PropertyGroup Label="Globals">
    <VCProjectVersion>15.0</VCProjectVersion>
    <ProjectGuid>{D13CE233-3DCE-4575-8C47-227E1621C12B}</ProjectGuid>
    <Keyword>Win32Proj</Keyword>
    <RootNamespace>Minesweeper</RootNamespace>

    <WindowsTargetPlatformVersion>10.0.17134.0</WindowsTargetPlatformVersion>
  </PropertyGroup>
  <Import Project="$(VCTargetsPath)\Microsoft.Cpp.Default.props" />
  <PropertyGroup Condition="'$(Configuration)|$(Platform)'=='Debug|Win32'"
Label="Configuration">
    <ConfigurationType>Application</ConfigurationType>
    <UseDebugLibraries>true</UseDebugLibraries>
    <PlatformToolset>v141</PlatformToolset>
    <CharacterSet>Unicode</CharacterSet>
  </PropertyGroup>
  <PropertyGroup
Condition="'$(Configuration)|$(Platform)'=='Release|Win32'"
Label="Configuration">

```

```

        <ConfigurationType>Application</ConfigurationType>
        <UseDebugLibraries>>false</UseDebugLibraries>
        <PlatformToolset>v141</PlatformToolset>
        <WholeProgramOptimization>>true</WholeProgramOptimization>
        <CharacterSet>Unicode</CharacterSet>
    </PropertyGroup>
    <PropertyGroup Condition="'$(Configuration)|$(Platform)'=='Debug|x64'"
Label="Configuration">
        <ConfigurationType>Application</ConfigurationType>
        <UseDebugLibraries>>true</UseDebugLibraries>
        <PlatformToolset>v141</PlatformToolset>
        <CharacterSet>Unicode</CharacterSet>
    </PropertyGroup>
    <PropertyGroup Condition="'$(Configuration)|$(Platform)'=='Release|x64'"
Label="Configuration">
        <ConfigurationType>Application</ConfigurationType>
        <UseDebugLibraries>>false</UseDebugLibraries>
        <PlatformToolset>v141</PlatformToolset>
        <WholeProgramOptimization>>true</WholeProgramOptimization>
        <CharacterSet>Unicode</CharacterSet>
    </PropertyGroup>
    <Import Project="$(VCTargetsPath)\Microsoft.Cpp.props" />
    <ImportGroup Label="ExtensionSettings">
    </ImportGroup>
    <ImportGroup Label="Shared">
    </ImportGroup>
    <ImportGroup Label="PropertySheets"
Condition="'$(Configuration)|$(Platform)'=='Debug|Win32'"
        <Import Project="$(UserRootDir)\Microsoft.Cpp.$(Platform).user.props"
Condition="exists('$(UserRootDir)\Microsoft.Cpp.$(Platform).user.props')"
Label="LocalAppDataPlatform" />
    </ImportGroup>
    <ImportGroup Label="PropertySheets"
Condition="'$(Configuration)|$(Platform)'=='Release|Win32'"
        <Import Project="$(UserRootDir)\Microsoft.Cpp.$(Platform).user.props"
Condition="exists('$(UserRootDir)\Microsoft.Cpp.$(Platform).user.props')"
Label="LocalAppDataPlatform" />
    </ImportGroup>
    <ImportGroup Label="PropertySheets"
Condition="'$(Configuration)|$(Platform)'=='Debug|x64'"
        <Import Project="$(UserRootDir)\Microsoft.Cpp.$(Platform).user.props"
Condition="exists('$(UserRootDir)\Microsoft.Cpp.$(Platform).user.props')"
Label="LocalAppDataPlatform" />
    </ImportGroup>
    <ImportGroup Label="PropertySheets"
Condition="'$(Configuration)|$(Platform)'=='Release|x64'"
        <Import Project="$(UserRootDir)\Microsoft.Cpp.$(Platform).user.props"
Condition="exists('$(UserRootDir)\Microsoft.Cpp.$(Platform).user.props')"
Label="LocalAppDataPlatform" />
    </ImportGroup>
    <PropertyGroup Label="UserMacros" />
    <PropertyGroup Condition="'$(Configuration)|$(Platform)'=='Debug|Win32'"
        <LinkIncremental>>true</LinkIncremental>
    </PropertyGroup>
    <PropertyGroup Condition="'$(Configuration)|$(Platform)'=='Debug|x64'"
        <LinkIncremental>>true</LinkIncremental>
    </PropertyGroup>
    <PropertyGroup
Condition="'$(Configuration)|$(Platform)'=='Release|Win32'"
        <LinkIncremental>>false</LinkIncremental>
    </PropertyGroup>

```

```

    <PropertyGroup Condition="'$(Configuration)|$(Platform)'=='Release|x64'">
      <LinkIncremental>>false</LinkIncremental>
    </PropertyGroup>
    <ItemDefinitionGroup
Condition="'$(Configuration)|$(Platform)'=='Debug|Win32'">
      <ClCompile>
        <PrecompiledHeader>Use</PrecompiledHeader>
        <WarningLevel>Level3</WarningLevel>
        <Optimization>Disabled</Optimization>
        <SDLCheck>>true</SDLCheck>

<PreprocessorDefinitions>WIN32;_DEBUG;_CONSOLE;% (PreprocessorDefinitions)</
PreprocessorDefinitions>
        <ConformanceMode>>true</ConformanceMode>
        <PrecompiledHeaderFile>pch.h</PrecompiledHeaderFile>
      </ClCompile>
      <Link>
        <SubSystem>Console</SubSystem>
        <GenerateDebugInformation>>true</GenerateDebugInformation>
      </Link>
    </ItemDefinitionGroup>
    <ItemDefinitionGroup
Condition="'$(Configuration)|$(Platform)'=='Debug|x64'">
      <ClCompile>
        <PrecompiledHeader>Use</PrecompiledHeader>
        <WarningLevel>Level3</WarningLevel>
        <Optimization>Disabled</Optimization>
        <SDLCheck>>true</SDLCheck>

<PreprocessorDefinitions>_DEBUG;_CONSOLE;% (PreprocessorDefinitions)</Prepro
cessorDefinitions>
        <ConformanceMode>>true</ConformanceMode>
        <PrecompiledHeaderFile>pch.h</PrecompiledHeaderFile>
      </ClCompile>
      <Link>
        <SubSystem>Console</SubSystem>
        <GenerateDebugInformation>>true</GenerateDebugInformation>
      </Link>
    </ItemDefinitionGroup>
    <ItemDefinitionGroup
Condition="'$(Configuration)|$(Platform)'=='Release|Win32'">
      <ClCompile>
        <PrecompiledHeader>Use</PrecompiledHeader>
        <WarningLevel>Level3</WarningLevel>
        <Optimization>MaxSpeed</Optimization>
        <FunctionLevelLinking>true</FunctionLevelLinking>
        <IntrinsicFunctions>true</IntrinsicFunctions>
        <SDLCheck>true</SDLCheck>

<PreprocessorDefinitions>WIN32;NDEBUG;_CONSOLE;% (PreprocessorDefinitions)</
PreprocessorDefinitions>
        <ConformanceMode>true</ConformanceMode>
        <PrecompiledHeaderFile>pch.h</PrecompiledHeaderFile>
      </ClCompile>
      <Link>
        <SubSystem>Console</SubSystem>
        <EnableCOMDATFolding>true</EnableCOMDATFolding>
        <OptimizeReferences>true</OptimizeReferences>
        <GenerateDebugInformation>true</GenerateDebugInformation>
      </Link>
    </ItemDefinitionGroup>

```

```

    <ItemDefinitionGroup
      Condition="'$(Configuration)|$(Platform)'=='Release|x64'">
        <ClCompile>
          <PrecompiledHeader>Use</PrecompiledHeader>
          <WarningLevel>Level3</WarningLevel>
          <Optimization>MaxSpeed</Optimization>
          <FunctionLevelLinking>true</FunctionLevelLinking>
          <IntrinsicFunctions>true</IntrinsicFunctions>
          <SDLCheck>true</SDLCheck>

        <PreprocessorDefinitions>NDEBUG;_CONSOLE;%(PreprocessorDefinitions)</PreprocessorDefinitions>
          <ConformanceMode>true</ConformanceMode>
          <PrecompiledHeaderFile>pch.h</PrecompiledHeaderFile>
        </ClCompile>
        <Link>
          <SubSystem>Console</SubSystem>
          <EnableCOMDATFolding>true</EnableCOMDATFolding>
          <OptimizeReferences>true</OptimizeReferences>
          <GenerateDebugInformation>true</GenerateDebugInformation>
        </Link>
      </ItemDefinitionGroup>
    <ItemGroup>
      <ClInclude Include="Board.h" />
      <ClInclude Include="Menu.h" />
      <ClInclude Include="pch.h" />
    </ItemGroup>
    <ItemGroup>
      <ClCompile Include="Board.cpp" />
      <ClCompile Include="Menu.cpp" />
      <ClCompile Include="Minesweeper.cpp" />
      <ClCompile Include="pch.cpp">
        <PrecompiledHeader
          Condition="'$(Configuration)|$(Platform)'=='Debug|Win32'">Create</PrecompiledHeader>
        <PrecompiledHeader
          Condition="'$(Configuration)|$(Platform)'=='Debug|x64'">Create</PrecompiledHeader>
        <PrecompiledHeader
          Condition="'$(Configuration)|$(Platform)'=='Release|Win32'">Create</PrecompiledHeader>
        <PrecompiledHeader
          Condition="'$(Configuration)|$(Platform)'=='Release|x64'">Create</PrecompiledHeader>
      </ClCompile>
    </ItemGroup>
    <Import Project="$(VCTargetsPath)\Microsoft.Cpp.targets" />
    <ImportGroup Label="ExtensionTargets">
    </ImportGroup>
  </Project>

```

### Minesweeper.vcproj.filters

```

<?xml version="1.0" encoding="utf-8"?>
<Project ToolsVersion="4.0"
  xmlns="http://schemas.microsoft.com/developer/msbuild/2003">
  <ItemGroup>
    <Filter Include="Source Files">
      <UniqueIdentifier>{4FC737F1-C7A5-4376-A066-2A32D752A2FF}</UniqueIdentifier>
    </Filter>
  </ItemGroup>

```

```

        <Extensions>cpp;c;cc;cxx;def;odl;idl;hpj;bat;asm;asmx</Extensions>
    </Filter>
    <Filter Include="Header Files">
        <UniqueIdentifier>{93995380-89BD-4b04-88EB-
625FBE52EBFB}</UniqueIdentifier>
        <Extensions>h;hh;hpp;hxx;hm;inl;inc;ipp;xsd</Extensions>
    </Filter>
    <Filter Include="Resource Files">
        <UniqueIdentifier>{67DA6AB6-F800-4c08-8B7A-
83BB121AAD01}</UniqueIdentifier>

<Extensions>rc;ico;cur;bmp;dlg;rc2;rct;bin;rgs;gif;jpg;jpeg;jpe;resx;tiff;t
if;png;wav;mfcribbon-ms</Extensions>
    </Filter>
</ItemGroup>
<ItemGroup>
    <ClInclude Include="pch.h">
        <Filter>Header Files</Filter>
    </ClInclude>
    <ClInclude Include="Menu.h">
        <Filter>Header Files</Filter>
    </ClInclude>
    <ClInclude Include="Board.h">
        <Filter>Header Files</Filter>
    </ClInclude>
</ItemGroup>
<ItemGroup>
    <ClCompile Include="pch.cpp">
        <Filter>Source Files</Filter>
    </ClCompile>
    <ClCompile Include="Minesweeper.cpp">
        <Filter>Source Files</Filter>
    </ClCompile>
    <ClCompile Include="Menu.cpp">
        <Filter>Source Files</Filter>
    </ClCompile>
    <ClCompile Include="Board.cpp">
        <Filter>Source Files</Filter>
    </ClCompile>
</ItemGroup>
</Project>

```

### Minesweeper.vcxproj.user

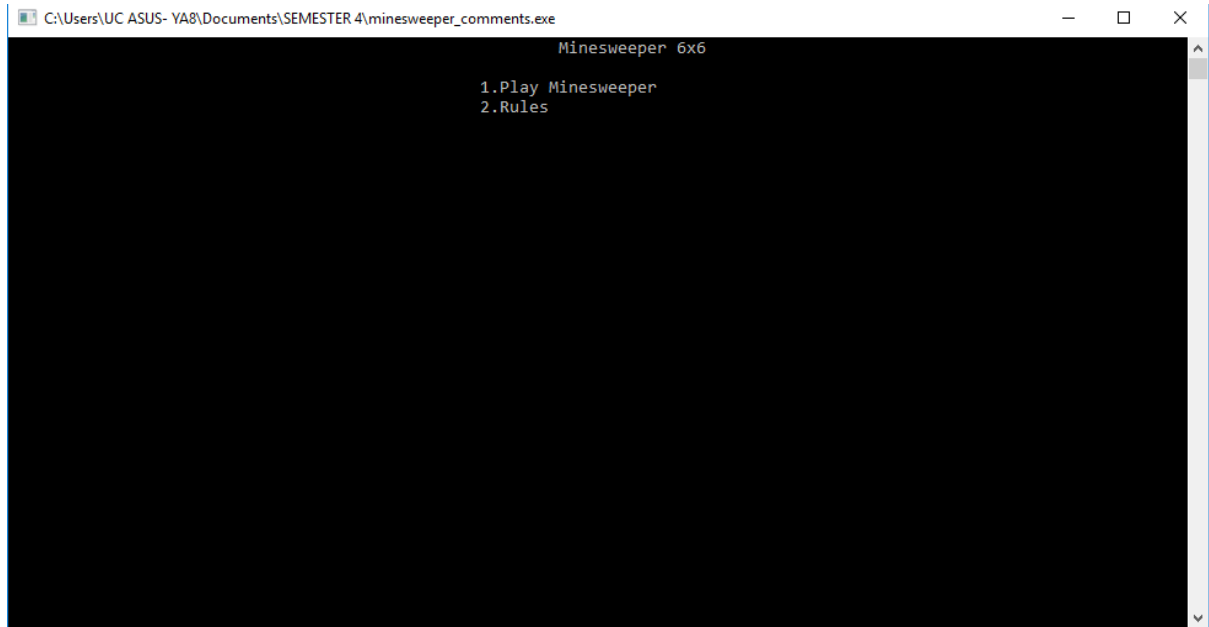
```

<?xml version="1.0" encoding="utf-8"?>
<Project ToolsVersion="15.0"
xmlns="http://schemas.microsoft.com/developer/msbuild/2003">
    <PropertyGroup />
</Project>

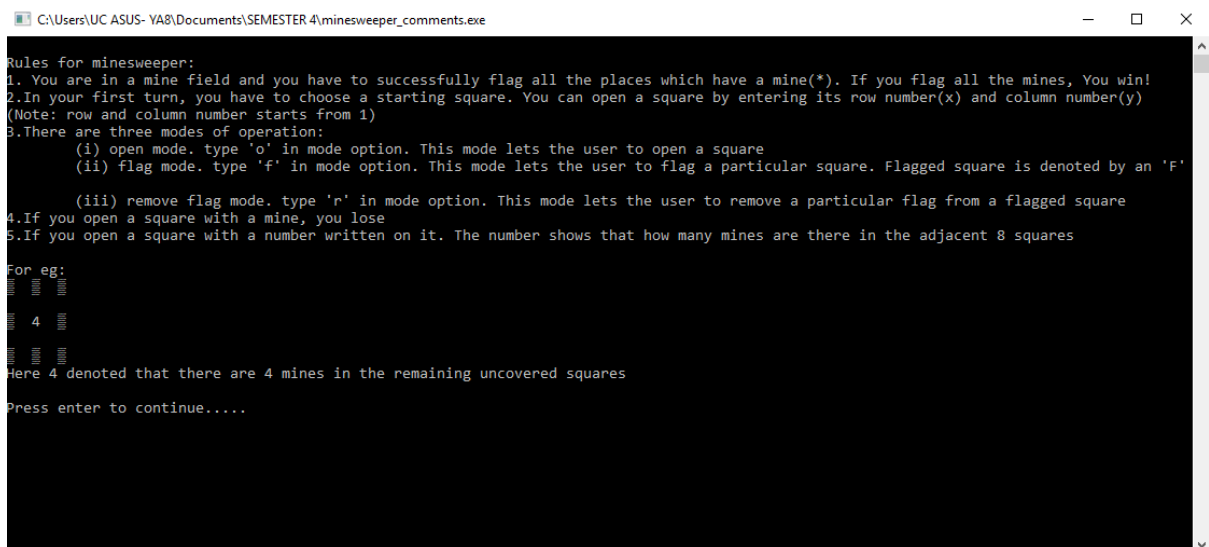
```

# How It Works

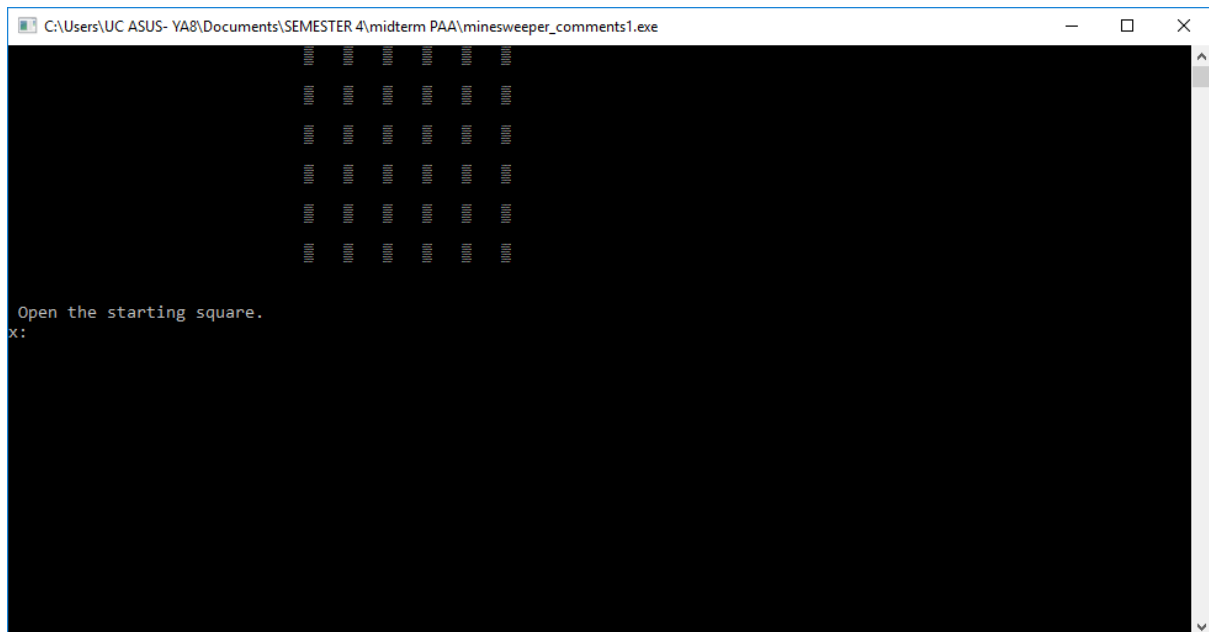
## Initial Display



## Rules

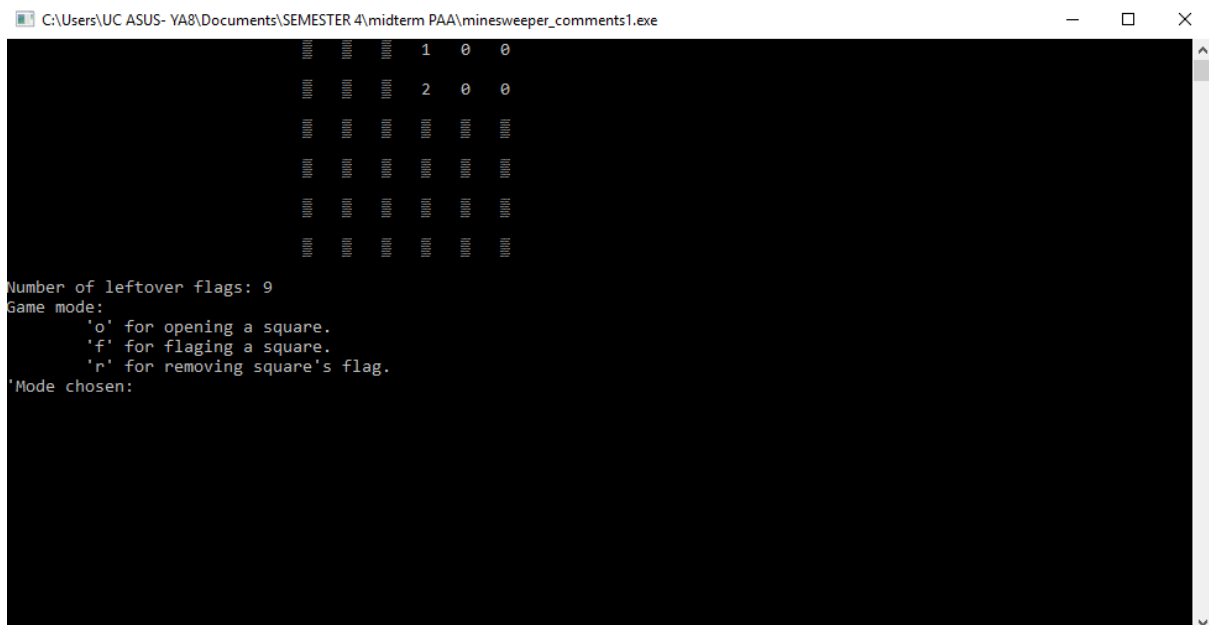


## Initial Display when The game start



*We need to input the coordinate (x,y) that we want to open first*

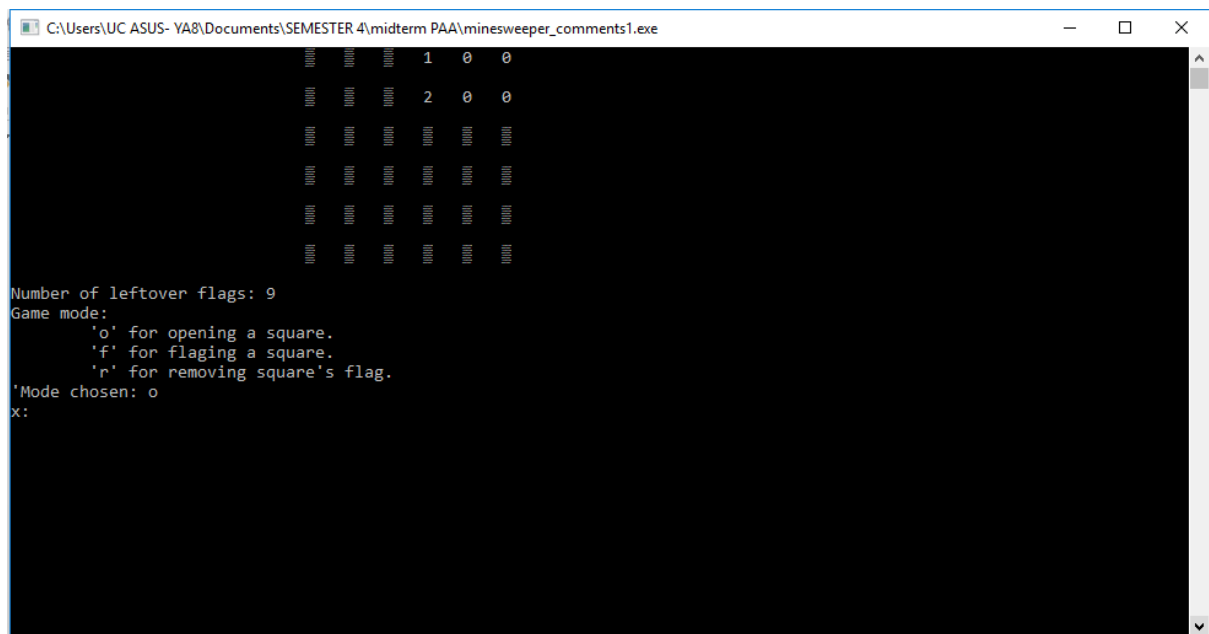
## After we put first coordinate



*It ask what mode we will choose for the next step*



🌈 If we choose mode 'o'

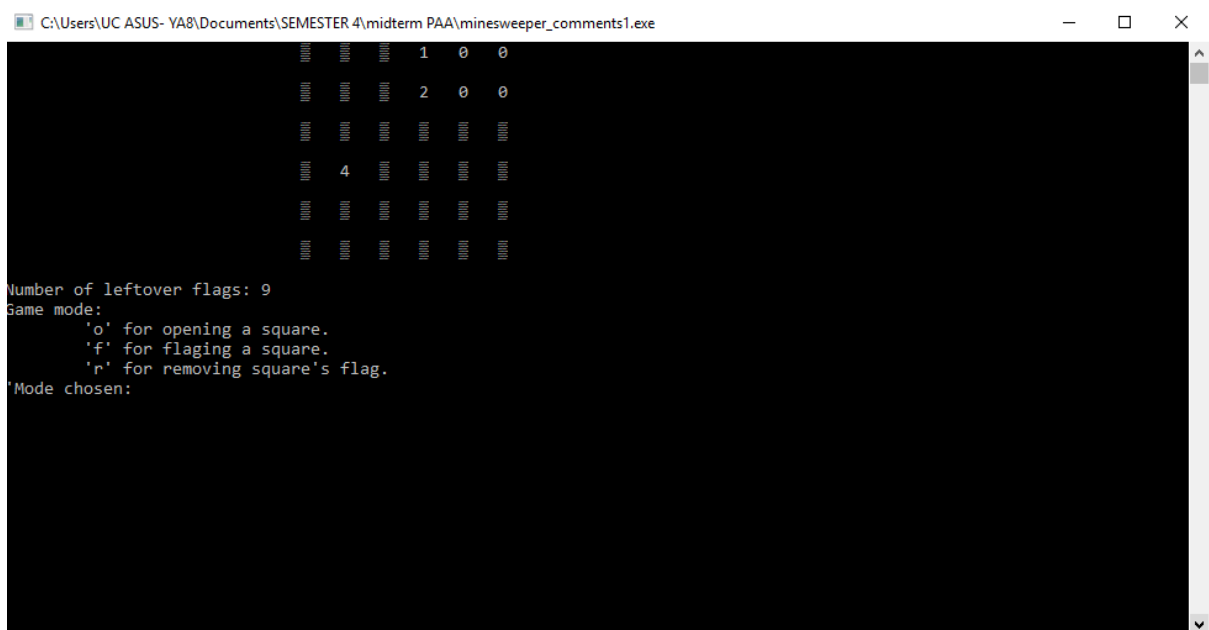


```
C:\Users\UC ASUS- YA8\Documents\SEMESTER 4\midterm PAA\minesweeper_comments1.exe

  ||| ||| ||| 1 0 0
  ||| ||| ||| 2 0 0
  ||| ||| ||| ||| |||
  ||| ||| ||| ||| |||
  ||| ||| ||| ||| |||
  ||| ||| ||| ||| |||

Number of leftover flags: 9
Game mode:
'o' for opening a square.
'f' for flaging a square.
'r' for removing square's flag.
'Mode chosen: o
x:
```

*It will ask for another coordinate to open*




```
C:\Users\UC ASUS- YA8\Documents\SEMESTER 4\midterm PAA\minesweeper_comments1.exe

  ||| ||| ||| 1 0 0
  ||| ||| ||| 2 0 0
  ||| ||| ||| ||| |||
  ||| 4 ||| ||| |||
  ||| ||| ||| ||| |||
  ||| ||| ||| ||| |||

Number of leftover flags: 9
Game mode:
'o' for opening a square.
'f' for flaging a square.
'r' for removing square's flag.
'Mode chosen:
```

*After we put coordinate, it will open another field*

 If we choose mode 'f'

```
C:\Users\UC ASUS- YA8\Documents\SEMESTER 4\midterm PAA\minesweeper_comments1.exe

  1 0 0
  2 0 0
  4
  0 0 0
  0 0 0
  0 0 0

Number of leftover flags: 9
Game mode:
'o' for opening a square.
'f' for flaging a square.
'r' for removing square's flag.
'Mode chosen: f
x:
```

*It will ask for coordinate we want to mark*

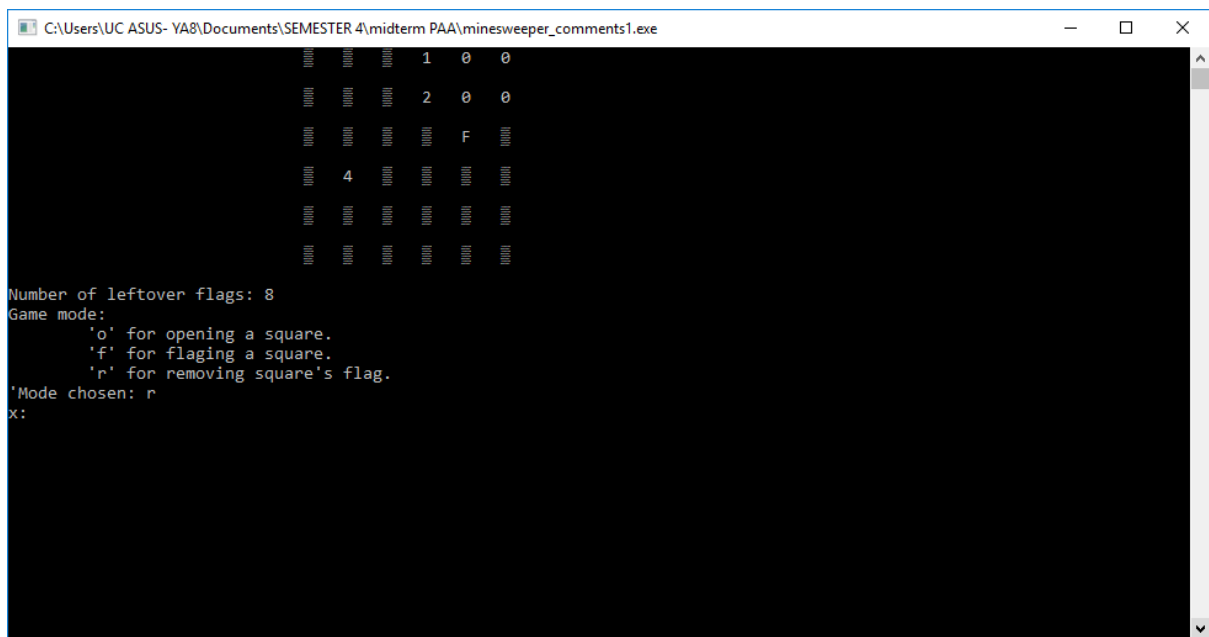
```
C:\Users\UC ASUS- YA8\Documents\SEMESTER 4\midterm PAA\minesweeper_comments1.exe

  1 0 0
  2 0 0
  4
  0 0 0
  0 0 0
  0 0 0

Number of leftover flags: 8
Game mode:
'o' for opening a square.
'f' for flaging a square.
'r' for removing square's flag.
'Mode chosen:
```

*After we choose coordinate to mark*

🌈 If we choose mode 'r'

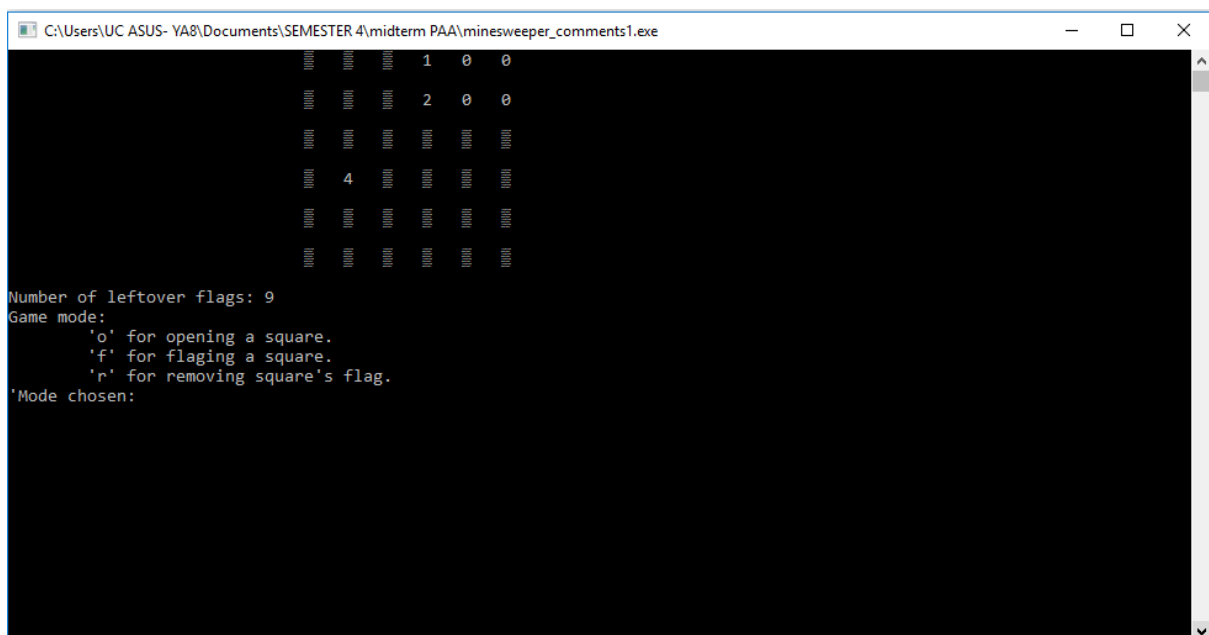


```
C:\Users\UC ASUS- YA8\Documents\SEMESTER 4\midterm PAA\minesweeper_comments1.exe

  1 0 0
  2 0 0
  F
  4
  5
  6

Number of leftover flags: 8
Game mode:
  'o' for opening a square.
  'f' for flagging a square.
  'r' for removing square's flag.
'Mode chosen: r
x:
```

*It will ask coordinate which mark we want to remove*



```
C:\Users\UC ASUS- YA8\Documents\SEMESTER 4\midterm PAA\minesweeper_comments1.exe

  1 0 0
  2 0 0
  F
  4
  5
  6

Number of leftover flags: 9
Game mode:
  'o' for opening a square.
  'f' for flagging a square.
  'r' for removing square's flag.
'Mode chosen:
```

*It will look like that after we remove the flag*

🌈 If we accidentally choose coordinates with bomb

```
C:\Users\UC ASUS- YA8\Documents\SEMESTER 4\midterm PAA\minesweeper_comments1.exe

  0  1  2  3  4  5
  0  2  *  2  0  0
  1  3  *  2  1  1
  *  4  1  2  1  1
  *  *  1  *  2  1
  *  1  *  1  1  1

Number of leftover flags: 9
Game mode:
'o' for opening a square.
'f' for flaging a square.
'r' for removing square's flag.

You opened a bomb!
GAME OVER

Play again?
1. Yes
2. No
```

🌈 If we not choose any bomb until the end

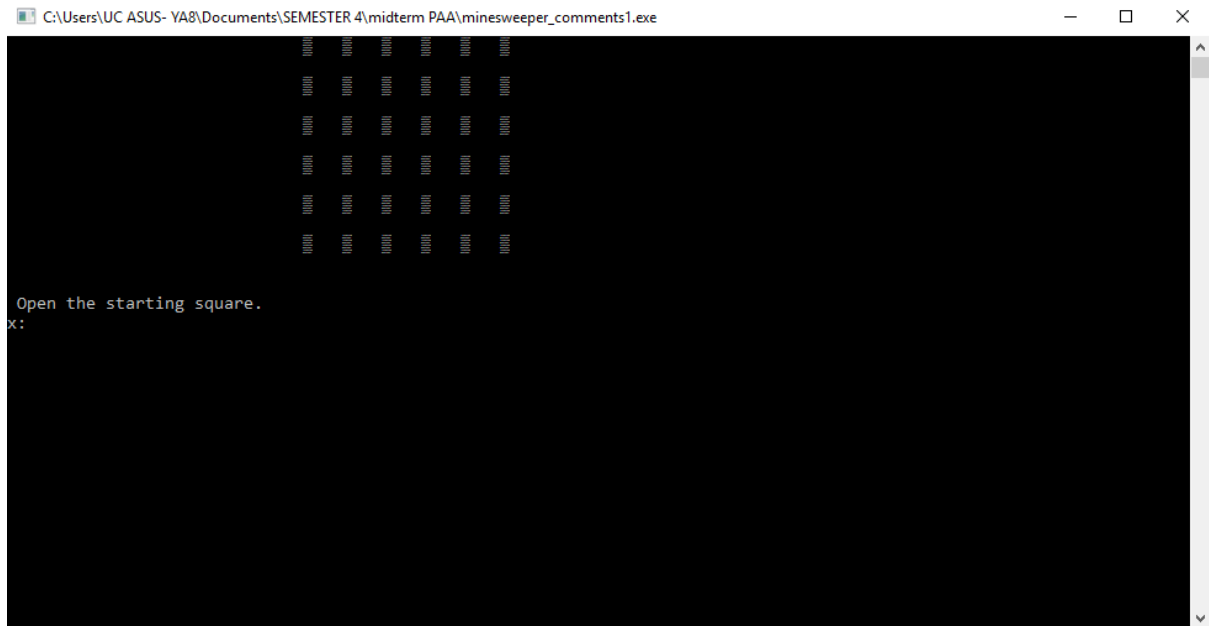
```
  2  F  F  1  1  1
  4  F  4  1  1  F
  F  F  3  0  1  1
  F  F  2  0  0  0
  2  2  1  0  1  1
  0  0  0  0  1  F

Number of leftover flags: 0
Game mode:
'o' for opening a square.
'f' for flaging a square.
'r' for removing square's flag.

You win
Play again?
1. Yes
2. No
```

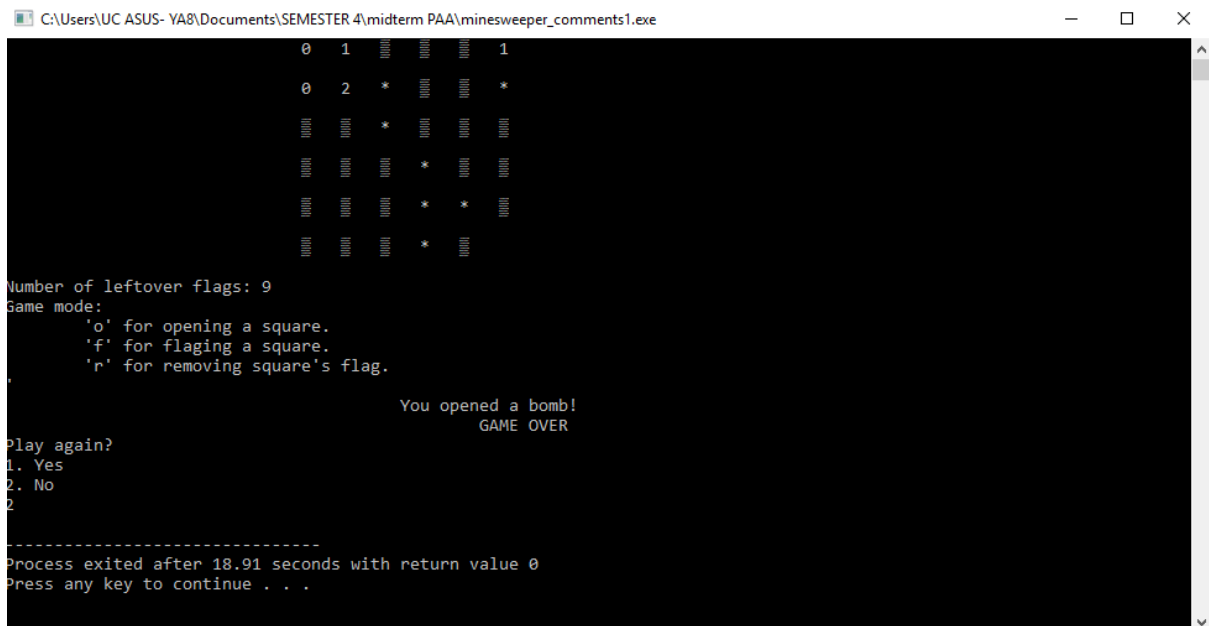
*We win.*

🌈 If we choose yes for play again



*It will back to Initial Display when The game start*

🌈 If we choose no for play again



*The program will stop running*

## Source Code Explanation

### 🚧 Entering the game phase

```
while(1){
    char grid[6][6],
    field[6][6];
    char m[2],
    tmpstr[10];
    int gridn[6][6] =
    {0};
    int i, j, k, l;
    int x, y, xs, ys;
    int mine = 0;
    int nof;
    int a = 0;
    char menu;
```

Resetting the value of the boxes

Initialization

### 🚧 Passing to menu function for the interface

```
char menu;
for(i = 0; i < 6; i++) {
    for(j = 0; j < 6; j++) {
        grid[i][j] = 0;
    }
}
printf("\t\t\t\t\t\t\tMinesweeper 6x6\n\n");
printf("\t\t\t\t\t\t\t1.Play Minesweeper\n\t\t\t\t\t\t\t2.Rules\n");
scanf("%c",&menu);
if(menu == 50){
    system("cls");
    printf("This is the classic game of minesweeper written in
C.\n\nRules for minesweeper:\n");
    printf("1. You are in a mine field and you have to successfully
flag all the places which have a mine(%c). If you flag all the mines, You
win!\n",42);
    printf("2.In your first turn, you have to choose a starting
square. You can open a square by entering its row number(x) and column
number(y)\n(Note: row and column number starts from 1)\n");
    printf("3.There are three modes of operation:\n");
    printf("\t(i) open mode. type 'o' in mode option. This mode lets
the user to open a square\n");
    printf("\t(ii) flag mode. type 'f' in mode option. This mode
lets the user to flag a particular square. Flagged square is denoted by an
'F'\n");
    printf("\t(iii) remove flag mode. type 'r' in mode option. This
mode lets the user to remove a particular flag from a flagged square\n");
    printf("4.If you open a square with a mine, you lose\n");
    printf("5.If you open a square with a number written on it. The
number shows that how many mines are there in the adjacent 8 squares\n");
    printf("\nFor eg:\n%c %c %c\n\n%c 4 %c\n\n%c %c
%c",177,177,177,177,177,177,177,177);
    printf("\nHere 4 denoted that there are 4 mines in the remaining
uncovered squares");
    printf("\n\nPress enter to continue.....");
    fflush(stdin);
    gets(tmpstr);
}
```

Delete the content of the input buffer

- ✚ Constructing the minesweeper's game interface of 6x6 into some symbol in ascii code

```
system("cls");
for(i = 0; i < 6; i++){
    for(j = 0; j < 6; j++){
        field[i][j] = 177;
    }
}
```

Clean the screen on the program to be run so the program can determine when to delete the data that has been run without having to close the program and reopen it

- ✚ Making it beautiful by giving it some space so that it doesn't look too awkward to see

```
for(i = 0; i < 6; i++){
    for(j = 0; j < 30; j++){
        printf(" ");
    }
    for(j = 0; j < 6; j++){
        printf("%c", field[i][j]);
    }
    printf("\n\n");
}
```

Making it a little bit centered on the terminal

Giving the space between each box as we constructing the game interface

- ✚ Giving details on how to use the mode option and starting the game by opening the first block of boxes

```
for(;;){
    printf("\n Open the starting square.\n");
    printf("x: ");
    scanf("%d",&xs);
    printf("y: ");
    scanf("%d",&ys);
    if(xs>6 || ys>6 || xs<1 || ys<1){
        printf("Row or column not defined. Try again\n"); continue;
    }
    else break;
}
```

Same function as While(true)

Scanning the coordinate of (x,y) the value of (x,y) can't be more than 6 and less than 1 storing the (x,y) into (xs,ys)

- ✚ Creating the mines after we trying to open the starting boxes using the rand() function to get many cases of the game limiting the mines to 9 only

```
xs = xs-1;
ys = ys-1;
srand(time(NULL));
while(1){
    i=rand()%6;
    j=rand()%6;
    if(grid[i][j]!=42 && i!=xs && j!=ys) grid[i][j]=42;
    else continue;
    mine++;
    if(mine == 9) break;
}
```

Makes use of the computer's internal clock to control the choice of the seed.

✚ Creating the hints or boxes value to indicate where the bombs are

```
for(i = 0; i < 6; i++){
    for(j = 0; j < 6; j++){
        if(grid[i][j] != 42){
            for(k = i-1; k <= i+1; k++){
                for(l = j-1; l <= j+1; l++){
                    if(grid[k][l] == 42 && k
>= 0 && l >= 0 && k <= 5 && l <= 5)
                        gridn[i][j]++;
                }
            }
            grid[i][j] = gridn[i][j] + 48;
        }
    }
}
```

Make sure that the k and l values fixed in 0-5 counting the sum of mines nearby

Since gridn's data type is int, so we need to add 48 to make it numeric in ascii code

✚ Storing the hints value as the bombs stored with \* symbol in ascii

```
for(i = xs-1; i <= xs+1; i++)
{
    for(j = ys-1; j <= ys+1; j++)
    {
        if(grid[i][j] != 42){
            if (yf == 1 && j==1) continue;
            else if (yf == 6 && j==6) continue;
            field[i][j] = grid[i][j];
            if(field[i][j] == 48){
                field[i-1][j] = grid[i-1][j];
                field[i+1][j] = grid[i+1][j];
                field[i][j+1] = grid[i][j+1];
                field[i][j-1] = grid[i][j-1];
                field[i-1][j+1] = grid[i-1][j+1];
                field[i-1][j-1] = grid[i-1][j-1];
                field[i+1][j+1] = grid[i+1][j+1];
                field[i+1][j-1] = grid[i+1][j-1];
            }
            printf("i: %d j: %d value: %c\n", i, j,
field[i][j]);
        }
    }
}
```



```

while(1){
    system("cls");
    for(i = 0; i < 6; i++)
    {
        for(j = 0; j < 30; j++)
            printf(" ");
        for(j = 0; j < 6; j++)
            printf("%c  ", field[i][j]);
        printf("\n\n");
    }

    printf("Number of leftover flags: %d\n",
nof);
    printf("Game mode:\n");
    printf("\t'o' for opening a square.\n\t'f'
for flagging a square.\n\t'r' for removing square's
flag.\n");

    if(grid[x][y] == 42 && strcmp(m, "o") == 0)
    {
        printf("\n\t\t\t\t\tYou opened a
bomb!");
        printf("\n\t\t\t\t\tGAME OVER\n");
        break;
    }

    for(i = 0; i < 6; i++)
    {
        for(j = 0; j < 6; j++)
        {
            if(grid[i][j] == 42 && field[i][j]
== 70)
                a++;
        }
    }

    if(a == mine)
    {
        printf("You win\n");
        break;
    }
}

```

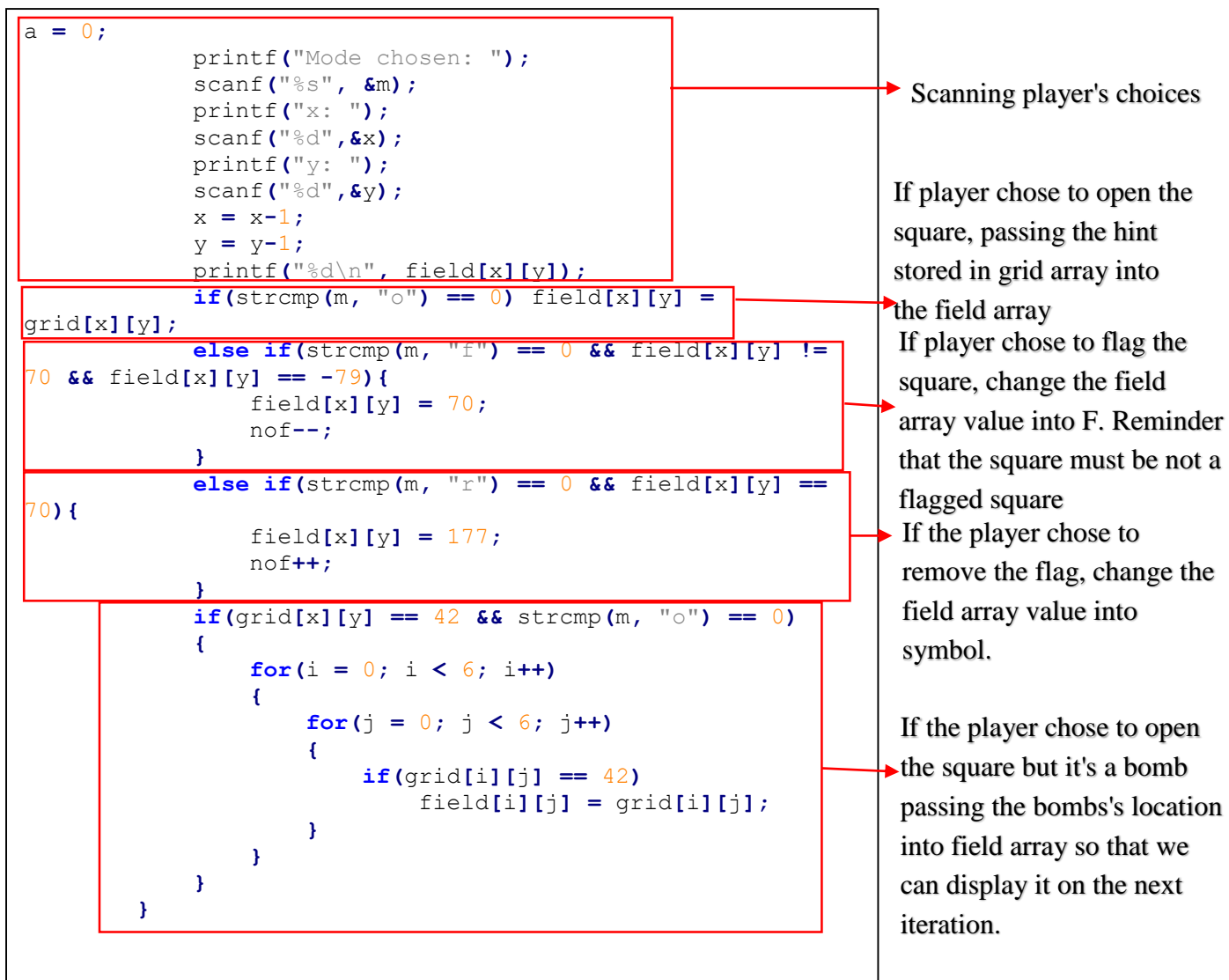
Revealing the opened starting square on the first iteration. On the next iteration, revealing the player's choices, whether to open or flag the squares. If a bomb is opened in the previous iteration, revealing all the bombs location.

- ▶ Giving hints for the leftover flags and providing details on game modes

➔ If the square is a bomb

→ Checking if the player flagging the right squares with bombs

→ Win if all the bombs are flagged



🎮 Option to play the game again or exit

```

int opt;
printf("Play
again?\n1. Yes\n2. No\n");
scanf("%d", &opt);
if(opt==2) break;
}

```

## Explanation of Algorithm

So, in this minesweeper we using **Depth First Search** algorithm.

In minesweeper, if player clicks an empty tile (tile that do not have number or bomb on it) all other empty tiles adjacent to it will also be opened.



The main objective of the algorithm is to visit all empty tiles and open it. If it encounters a numbered tile, it opens the tile but not looks further. Since the main objective is to visit all tiles (or, in graph theory term, nodes).

✚ Suppose there are these functions and variables in the game code:

- Tiles (variable)  
A matrix that resembles minesweeper game board. It may has three kind of value : bomb, number and empty. Tiles[i,j] means tiles at column i row j.
- Mark (procedure) : marks Tiles[i,j] as visited / unvisited
- Open (procedure) : opens Tiles[i,j]

✚ DFS implementation is defined as follows:

- Mark all tiles as unvisited
- Mark as visited
- While not empty
  - Delete top element
  - Open o If tile[k,l] is empty tile then Check for every valid index neighboring . If it is unvisited, visit it and mark it as visited.

In minesweeper, there are 8 neighbors of = { , , , , { , , , }, in simple, all 8 tiles surrounding it.

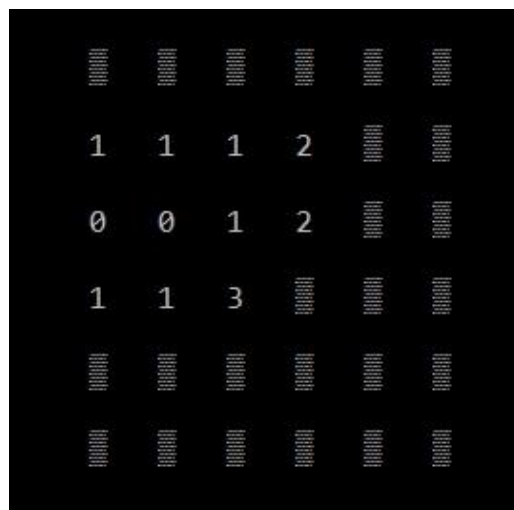
A valid neighbor is defined as a neighbor that its index is within the index bound of the matrix, i.e. if matrix's index is [1...100, 1...100], then is valid if and only if  $1 < x < 100$  and  $1 < y < 100$ .

## Implementation of DFS in our Program

### Source Code

```
for(i = xs-1; i <= xs+1; i++)
{
    for(j = ys-1; j <= ys+1; j++)
    {
        if(grid[i][j] != 42){
            if (yf == 1 && j==1) continue;
            else if (yf == 6 && j==6) continue;
            field[i][j] = grid[i][j];
            if(field[i][j] == 48){
                field[i-1][j] = grid[i-1][j];
                field[i+1][j] = grid[i+1][j];
                field[i][j+1] = grid[i][j+1];
                field[i][j-1] = grid[i][j-1];
                field[i-1][j+1] = grid[i-1][j+1];
                field[i-1][j-1] = grid[i-1][j-1];
                field[i+1][j+1] = grid[i+1][j+1];
                field[i+1][j-1] = grid[i+1][j-1];
            }
            printf("i: %d j: %d value: %c\n", i, j, field[i][j]);
        }
    }
}
```

### Result



*We choose coordinate (3,3).*

### Note

We only use one-level DFS because our minesweeper's dimension is too small (6x6). If we put another level of DFS it will be too easy.

## References

<https://www.gamedev.net/forums/topic/644428-simple-console-based-minesweeper-game/>

<https://www.dreamincode.net/forums/topic/138018-simple-textual-minesweeper/>

<https://programmersheaven.com/discussion/382963/somebody-have-simple-minesweeper-source-code-in-c>

<http://informatika.stei.itb.ac.id/~rinaldi.munir/Matdis/2012-2013/Makalah2012/Makalah-IF2091-2012-069.pdf>

<http://massivealgorithms.blogspot.com/2017/08/leetcode-529-minesweeper.html>

<https://www.geeksforgeeks.org/cpp-implementation-minesweeper-game/>

"By the name of Allah (God) Almighty, herewith we pledge and truly declare that we have solved quiz 2 by ourselves, didn't do any cheating by any means, didn't do any plagiarism, and didn't accept anybody's help by any means. We are going to accept all of the consequences by any means if it has proven that we have been done any cheating and/or plagiarism."

Surabaya & Malang, 30 March 2019



Sherly Rosa Anggraeni  
05111340000018



Adzra Zaky Haura  
05111340000037



Indira Nursyamsina Hazimi  
05111340000082