Mine Sweeper

[Single player puzzle game]

PROJECT REPORT INT 213

BY

Reetika Tyagi , Kumar Dev and Abhinav kumar

Section – K19QK

Roll numbers- 55,65,66

Department of Intelligent Systems, School of Computer Science Engineering, Lovely Professional University, Jalandhar November, 2020

Student Declaration

This is to declare that this report has been written by me/us. No part of the report is copied from other sources. All information included from other sources have been duly acknowledged. I/We aver that if any part of the report is found to be copied, I/we are shall take full responsibility for it.

Signatures: Name: Reetika tyagi,

Abhinav,

Dev kumar

Roll Number: 55, 65, 66

Place: Online Submission using GitHub and UMS

Date: 31-10-20

Github link: https://github.com/dev691047/minisweeper-game

TABLE OF CONTENTS

BONAFIDE CERTIFICATE

Certified that this project report “MINESWEEPER” is the Bonafede work of “Reetika Tyagi, Abhinav, Dev Kumar” who carried out the project work under my supervision.

Dr. Dhanpratap Singh

Associate Professor,

ID- 25706

Department: Intelligence System

LPU, Phagwara

INTRODUCTION

Minesweeper is a single player puzzle video game. The objective of the game is to clear a rectangular board containing hidden "mines" or bombs without detonating any of them, with help from clues about the number of neighboring mines in each field. The game originates from the 1960s, and it has been written for many computing platforms in use today. It has many variations and offshoots.

In *Minesweeper*, mines are scattered throughout a board which is divided into cells. Cells have three states: uncovered, covered and flagged. A covered cell is blank and clickable, while an uncovered cell is exposed. Flagged cells are those marked by the player to indicate a potential mine location.

A player left-clicks a cell to uncover it. If a player uncovers a mined cell, the game ends. Otherwise, the uncovered cells displays either a number, indicating the quantity of mines adjacent to it, or a blank tile , and all adjacent non-mined cells will automatically be uncovered. Right-clicking on a cell will flag it, causing a flag to appear on it. Flagged cells are still covered, and a player can click on them to uncover them, although typically they must first be un flagged with an additional right-click.

The first click in any game will never be a mine.

To win the game, players must uncover all non-mine cells, at which point the timer is stopped. Flagging all the mined cells is not required.

Gameplay:

The goal of the game is to uncover every square without a mine under it. By uncovering a mine, you lose. Don't worry though; the first square you uncover will never be a mine! When you uncover a square, it will display a number that describes how many mines are in the surrounding eight squares. You can place a flag on a square to

signify that it has a mine under it. You do not need to place flags to win, but they sure are helpful! If you have marks selected, you can flag the square once more to mark it as a possible mine. You cannot uncover flagged squares, but you can uncover marked squares, so be careful. To un flag or unmark a square, flag or mark it again. If you get too frustrated, you can start a new game and try again! Sometimes you may run out of squares you know do not have mines under them. At this point, you'd have to make a random guess, so don't feel bad if luck wasn't on your side. There's always the next game! And if you decide you've played enough for the day, your settings are automatically backed up and synced to your account.

MAIN REPORT

BACKGROUND AND OBJECTIVES OF THE PROJECT

This is GUI based application designed using Tkinter, for making a complete minesweeper game in python language.

OBJECTIVE OF THIS PROJECT

The objective of the game 'Minesweeper' is to identify and mark hidden 'mines' without detonating any of them. Both luck and logic play a part, for while there are clues pointing to the presence of mines, the game also requires making educated guesses. The game is to clear a rectangular board containing hidden "mines" or bombs without detonating any of them, with help from clues about the number of neighboring mines in each field.

OUTCOME OF THE PROJECT ARE AS FOLLOW:

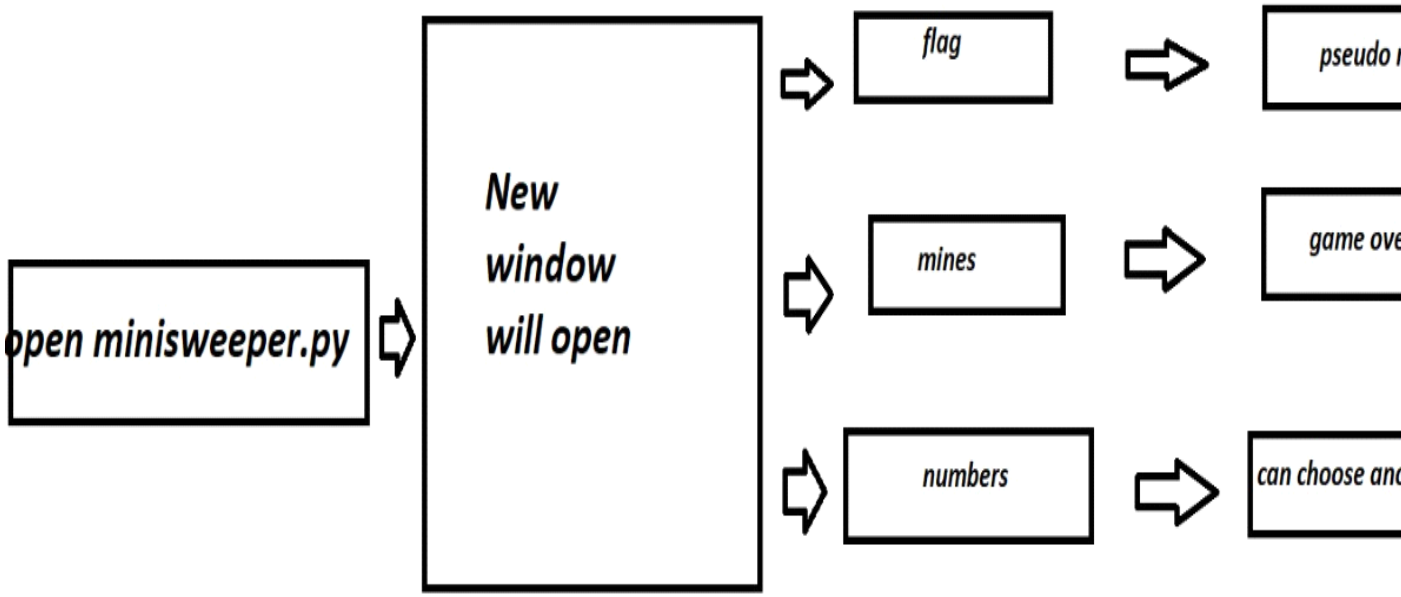
1.We learned modules like time and date.

2. Also understand the use of visual studio code.

3. The increasing number of tiles raises the difficulty bar.

So the complexity level increases as we proceed to next levels.

DESCRIPTION OF PROJECT



WORK DIVISION /ROLES AMONG STUDENTS:

Kumar dev:

coordinator

Major Part of coding using tkinter.

Major part of Testing

and debugging.

Reetika Tyagi:

part of UI design and code.

Making of presentation.

testing and debugging

Abhinav kumar:

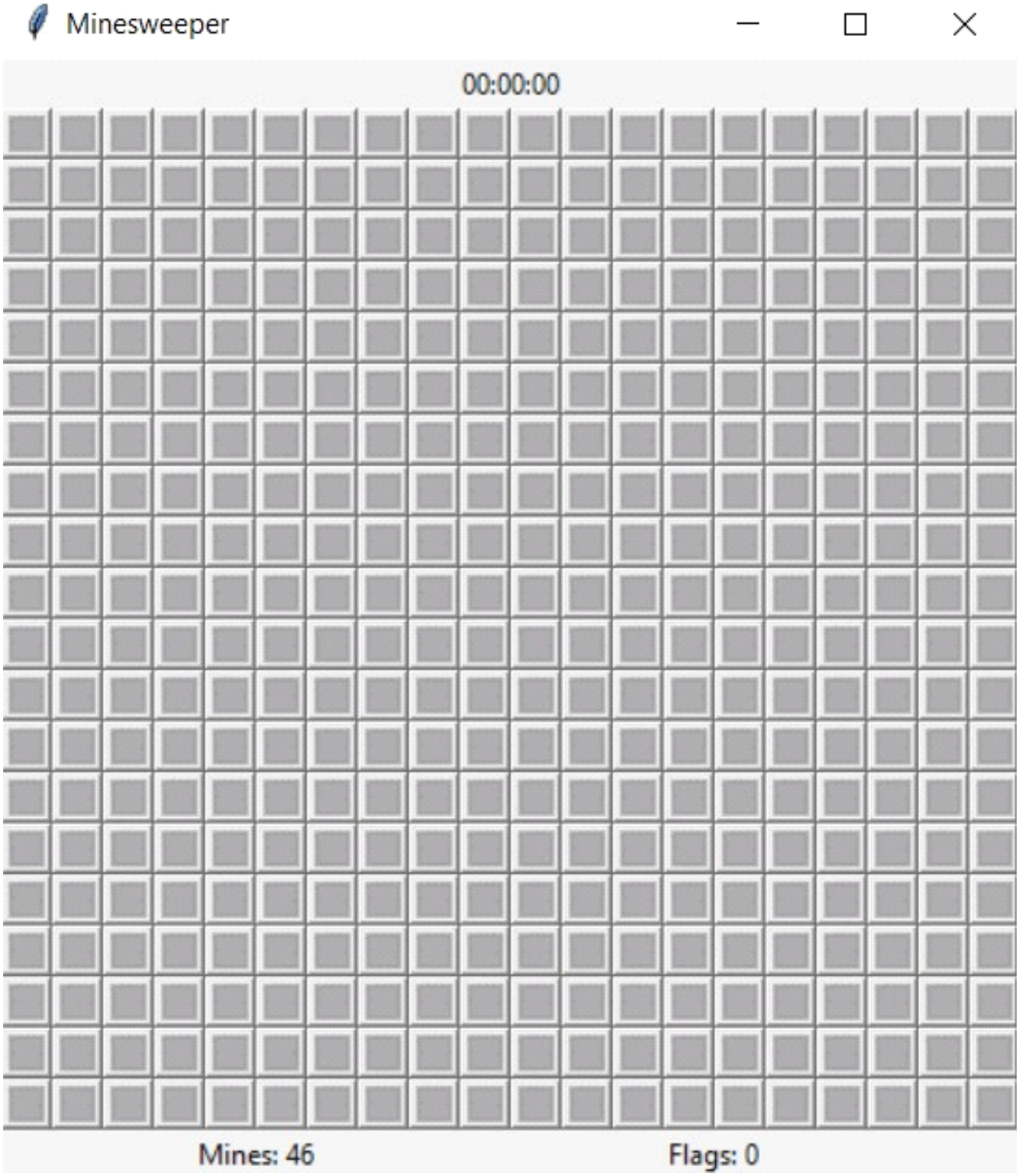
Making of report,

and helps in writing part of coding .

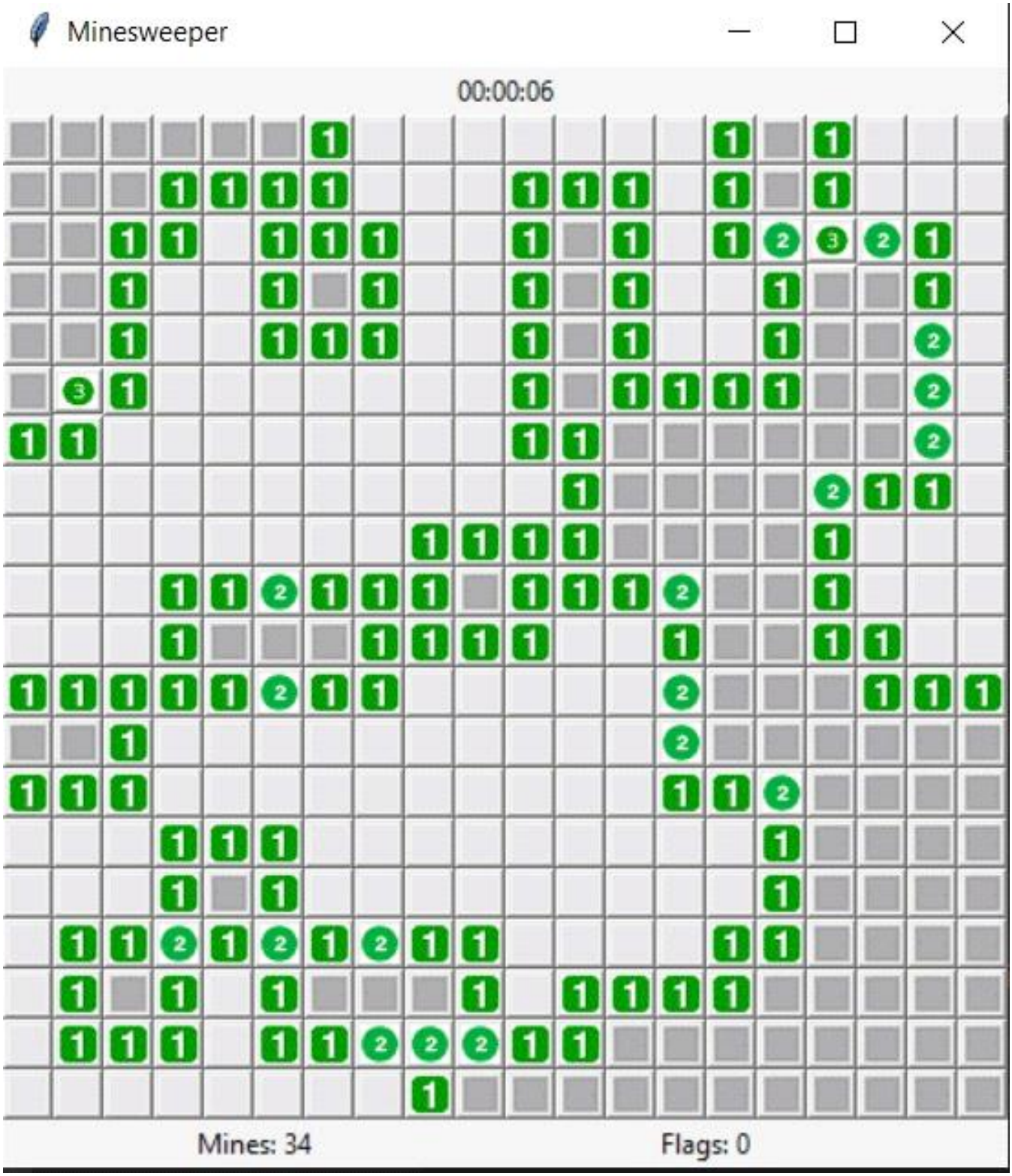
and testing.

EXPLANATION OF PROJECT:

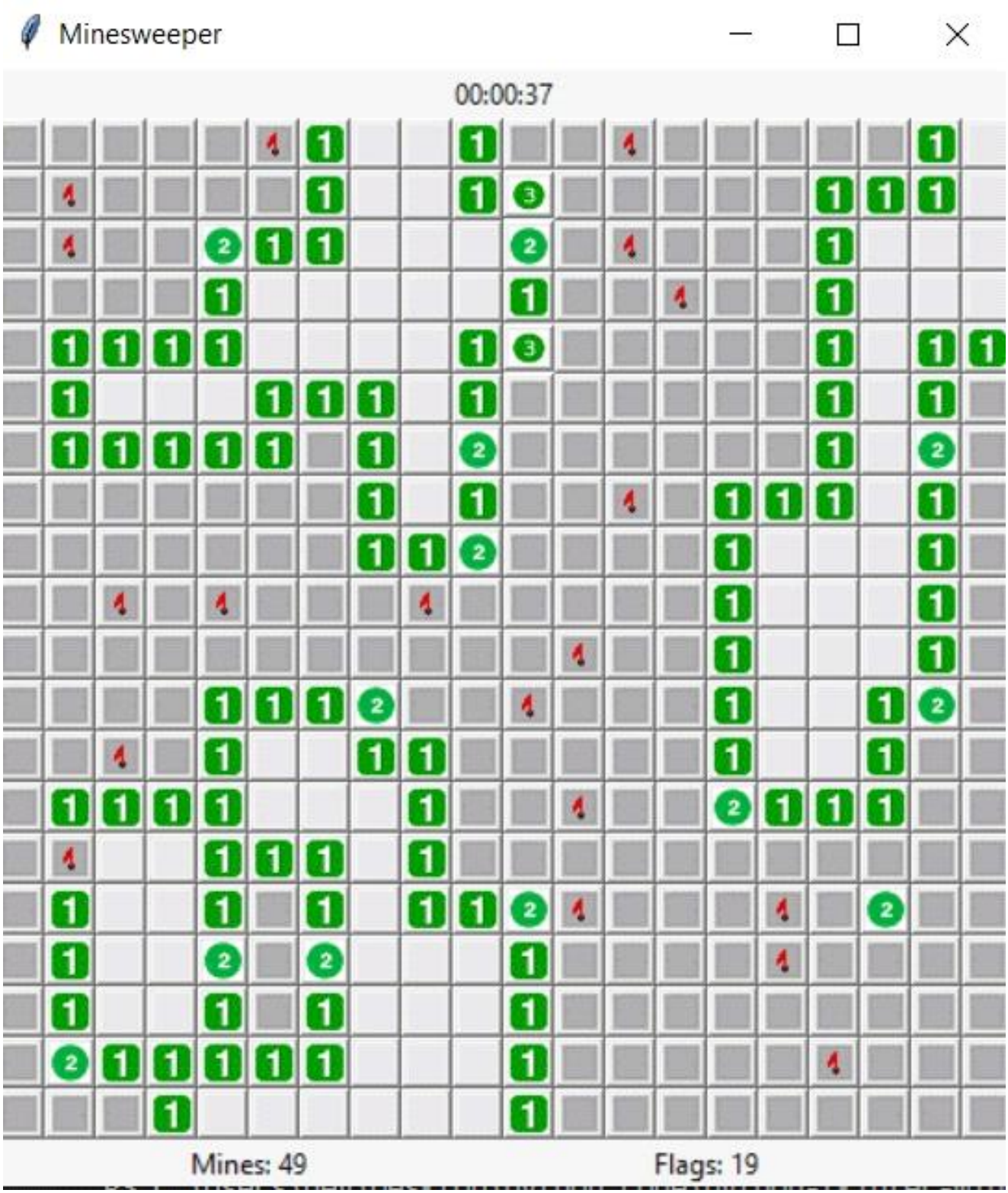
First window:



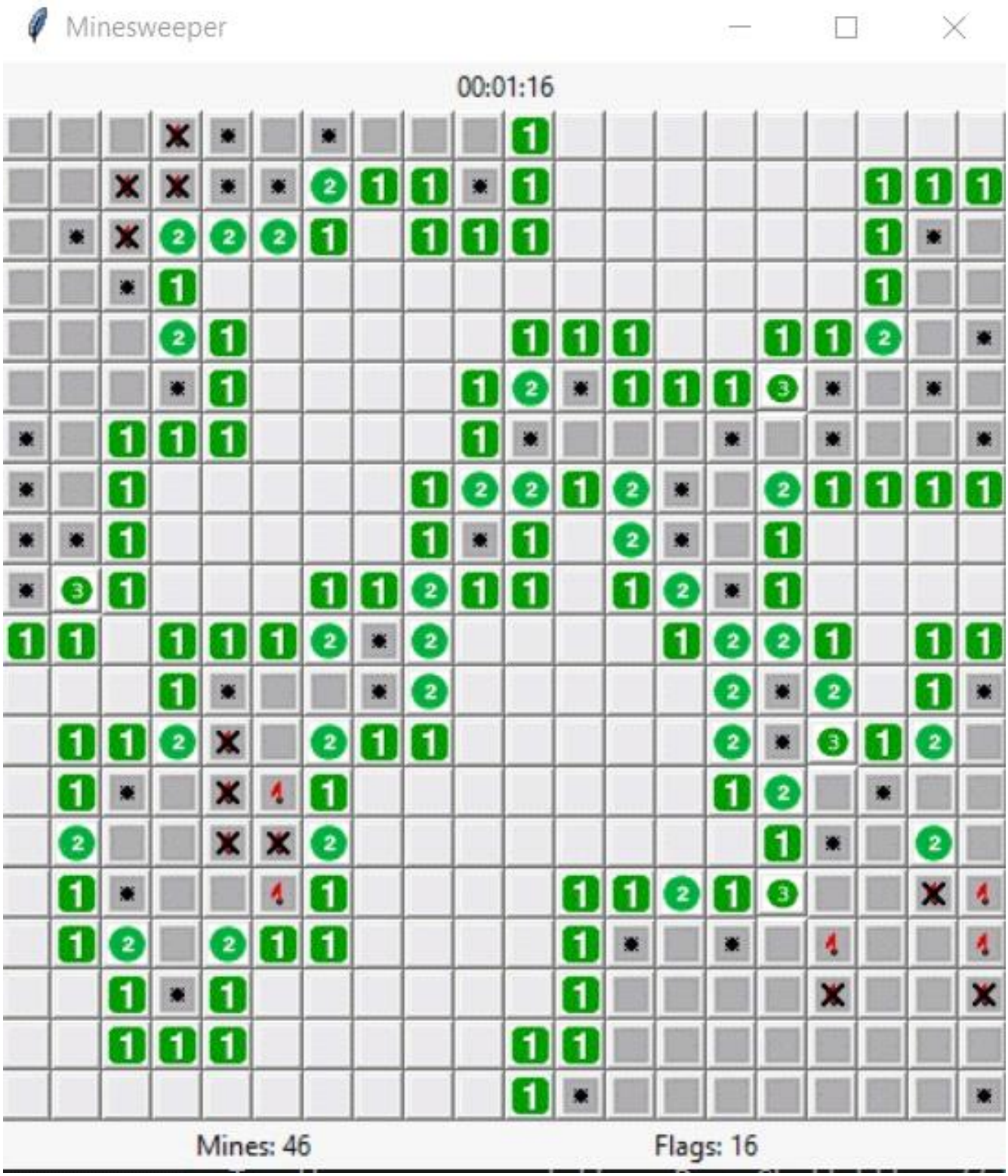
Second window:


Minesweeper
00:00:06
Mines: 34
Flags: 0

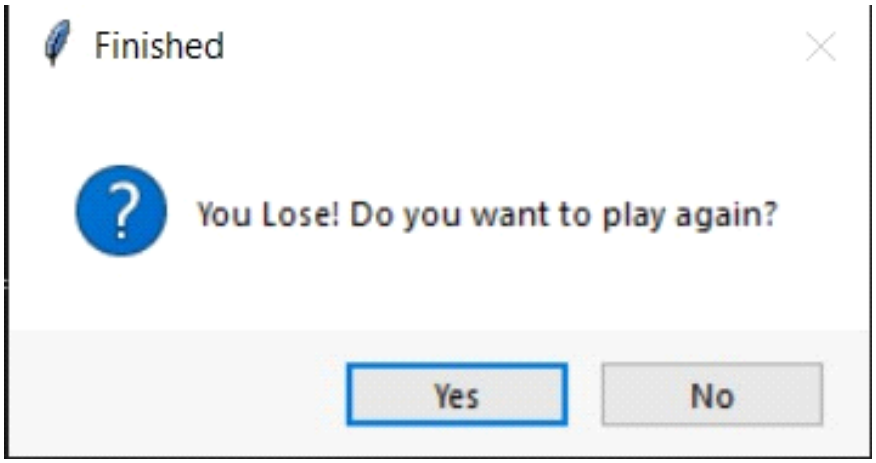
Third window:



Fourth window:



Fifth window:



CONCLUSION

During the process of creating the game, there were several obstacles that were encountered.

For example, there was difficulty in developing a manner in which all of the surrounding locations of a selected location could be checked for mines. This problem was solved by making an if-else selection structure that could check the values of the surrounding locations in the array that represented the game board. This was done by incrementing the indices of the array by different values depending on the location on the screen.

For example, the corner locations each had a different set of locations to check. The top right corner has to check the location below, below to the left, and the location to the left. Another difficulty that was encountered was developing a method to win. This was done by changing the values in the game board array that correspond to touched locations to equal two. then, after each touch, the components of the array are summed and compared to the value of the array if all the non-bomb locations had been touched, changed to a value of two, and summed with the value of the bomb locations.

Once the sum is reached, the user has cleared the minefield and won the game. This solution took planning and trial by error. The final difficulty that was encountered was difficulty in testing the game because the mines were randomly generated. To solve this problem, the randomly generated mines were commented out and bombs were manually placed in certain locations. This allowed for the testing of the game.

In terms of things that worked well, the for loops used to draw the boards worked well and made symmetric boards. Additionally, once the method for checking the surrounding locations of a selected location was developed it worked well in printing the number of surrounding bombs.

RECOMMENDATIONS:

To improve upon the game, there are several features that could be added.

For example, adjusting the program to ensure that a bomb could not be uncovered on the first tap. This could be done by generating the random row and column coordinates of the bomb locations in the game board array

after the first touch had been recorded on the touch screen. An if statement could be used in the code to prevent a bomb from being randomly placed in the location that corresponds to a location that was already touched by the user on the screen.

By generating the location of the bombs after the first touch has been made by the user, there is no way that the user could touch a bomb on the first touch. Then, the remaining logic of the code could still be applied.

Another reccomendation for the game is that flags could be added to the game. this would involve adding a button board that could be used to select locations on the board that the user suspects to have a bomb.

Then, the user could tap these locations and a shape could be printed to that desired location. An if statement could be used to allow for the user to tap the space again to remove the shape. These changes could be implemented with more time.

END OF THE PROJECT.