AMRITA
VISHWA VIDYAPEETHAM

Sub. Code : 19AIE112

Sub Name : Elements of computing systems - 2

Name of the Project : Minesweeper using Jack Language

Team members

1.Shyam Ganesh (21149)

2.Sidesh Sundar (21150)

3.Sabarinath (21141)

4.Jayakrishna (21154)

5.Sai Teja (21124)

6.Bharadwaj (21165)

ACKNOWLEDGEMENT

We would like to express our gratitude to
our professor V. Damodaran sir and Dr.Sivasundar Manisankar sir who gave us this golden opportunity to do the project on the topic " **Minesweeper using Jack Language".**

We would like to extend our gratitude to the AIE department of Amrita Vishwa Vidhyapeetham, Chennai who assigned this project to us and helped us to improve our knowledge.

TABLE OF CONTENTS

INTRODUCTION:

Minesweeper:

- 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.Players may elect to replay a board, in which the game is played by revealing squares of the grid by clicking or otherwise indicating each square. If a square containing a mine is revealed, the player loses the game. If no mine is revealed, a digit is instead displayed in the square, indicating how many adjacent squares contain mines; if no mines are adjacent, the square becomes blank, and all adjacent squares will be recursively revealed. The player uses this information to deduce the contents of other squares and may either safely reveal each square or mark the square as containing a mine.

Jack Language:

- Jack is a simple object-based language that can be used to write high-level programs. It has the basic features and flavor of modern object-oriented languages like Java, with a much simpler syntax and no support for inheritance. The introduction of Jack marks the beginning of the end of our journey

VM Emulator:

- In computing, a virtual machine (VM) is the virtualization/emulation of a computer system. Virtual machines are based on computer architectures and provide functionality of a physical computer. Their implementations may involve specialized hardware, software, or a combination. Virtual machines differ and are organized by their function, shown here:
  - System virtual machines (also termed full virtualization VMs) provide a substitute for a real machine. They provide functionality needed to execute entire operating systems. A hypervisor uses native execution to share and manage hardware, allowing for multiple environments which are isolated from one another, yet exist on the same physical machine.Modern hypervisors use hardware-assisted virtualization, virtualization-specific hardware, primarily from the host CPUs.

This is my final project for the nand2tetris course. It's a Minesweeper game written in the Jack language.The course challenges us to build a full application (in this case, Minesweeper) using a high level language (the Jack language) compiled using a compiler we wrote ourselves, with an operating system we write ourselves, running on a hardware platform (the Hack machine) we designed ourselves, powered by a CPU we wired together ourselves.

LITERATURE REVIEW:

Minesweeper is a well known game, having been on windows computers for many years.Minesweeper as we know it was created in 1990 and has been bundled with every new version of Windows since 1992.

The mobile Xbox Live version, developed by Babaroga and published by Microsoft, is one of the first two free ad-supported games for Xbox Live and the first to be available in regions outside of North America.

This game shares much in common with its fellow free game Sudoku, though the actual gameplay is completely different, with a much higher learning curve.

Get past that and you'll find an enjoyable logic puzzle game.Minesweeper consists of a grid or 'minefield' that contains a predetermined number of mines.

The player's goal is to uncover all of the squares that don't contain mines by digging out those squares.

Digging on a mine blows it up and ends the game, but this can usually be avoided by logical deduction.

After tapping one or more spaces on the board at random, several blank and numbered squares will be revealed.

The number on a square denotes how many mines are touching it – for instance, a square with a '1' is surrounded by one adjacent mine and seven blank spaces.

When players have deduced the location of a mine, they can plant a flag to mark it. So it goes until either all non-mine squares have been dug out (victory!) or the player hits a mine

We have also used an algorithm named BFS fill flood .

The idea is to use BFS traversal to replace the color with the new color.

- Create an empty queue lets say Q.
- Push the starting location of the pixel as given in the input and apply replacement color to it.
- Iterate until Q is not empty and pop the front node (pixel position).
- Check the pixels adjacent to the current pixel and push into the queue if valid (had not been colored with replacement color and have the same color as the old color).

METHODOLOGY:

In the project we have Jack language to write the code and converted it into VM-file so that we can run it in VM Emulator.

The project consists of 14 Jack files which are:

- Cursor
- Main
- Grid
- Maths
- Minesweeper
- Node
- Queue
- QueueTest
- Random
- RandomTest
- Sprites
- Strings
- Tutorial
- Utils

Cursor:

In this class we created the cursor by giving it a position in X and Y axis so that we can move the cursor along the grids.

At first we set the initial position as X=0 and Y=0 , locked = false so that the cursor can be moved and deallocated it's memory as "**this**".

Now the loop wait for the user to give the input so that action gets a value.

Initially the value of action is 0 after the user gives the input the cursor moves.

If the value of the pressed key is 130(left-arrow) the cursor must move left, similarly the value of key pressed is 131(up-arrow), 132(right-arrow), 133(down-arrow) the cursor moves up, right, down respectively .

If the value of key pressed is 70(f) the flag will be planted similarly the value of key pressed is 32(space) it reveals a cell, 82(r) it restarts the game , 81(q) it quits the game.

Main:

In this class we have uncommmented and recompiled the files to run the tests.

This initializes the game string and generates random numbers.

It checks whether QueueTest and RandomTest works properly or not and gives the output if it runs properly.

Grid:

In this class we created the grid by giving the values for column width, row height, flag count ,mine probability, mine count.

We have the give the percentage for randomness of placement of the mine.

Then we initialize the grid to 0 and place the mines in random places.

We now deallocated it's memory as "**this**".

If the value of cell = 0 it will display the closed cells but if the cell = 1 it will finish the game by showing all the mine position.

If the vale of cell = 2 it will place the flag ans if it is 3 it will remove the placed flag and if the value of cell is 4 it will show all empty cells.

Here we use BFS flood fill algorithm as we can't use a recursive algorithm and we must use a queue because the HACK computer's call stack isn't big enough to handle the large number of recursive calls we want to make.

Now we enqueue all neighboring cells into the provided queue.

Maths:

In this class we gets the value of X and Y and return the value of X-(Y-(X/Y)).

We also convert character to integer.

Minesweeper:

In this class we constructed the minesweeper game.

The first step is to set the grids and place the mines.

The base game variables values are grid-width=24, grid-height=12, grid-percentmines=12.

Now we initialize the object, drawing and we enter the main game loop.

The main loop starts with a while loop whose condition is action not equal to 4(which means the user did not press the quit button).

Next the if loop with condition action = 3 (on pressing the restart button) it will clear everything and redraw the new game and reinitialize the objects.

The next if loop has the condition action = 2 (on pressing the space) if the particular grid has the mine the game will return gamelost.

The next if loop has condition action = 1(on pressing the f) it will place the flag in particular grid and if we found all the mine it will return game won.

The second last if loop has the condition gameLost it will lock the cursor and it will display GameLost().

The last if loop has the condition grid.gameWon() it will lock the cursor and it will display GameWon().

Node:

In this class we have created a doubly linked list which has 3 values:

value

Prev

Next

Now we create the new node with prev and next as null and value as _value.

Now we get pointer to the next or prev node as per the commands given by the user and get the value of the node.

We have to set to next or prev pointer to the provided node.

Queue:

It is a simple FIFO queue implemented as a doubly-linked list so the values stored can only be integers.

Now we are creating a queue with head and tail value as null.

As the queue is empty if both head and tail don’t point to a node so it checks whether there is any value in queue and returns head = null and tail = null.

Now we are going to add value to the front of the queue so we are declaring a variable node whose value is the Node.new(value).

If there was a next node we have to link it’s previous back to the new node .

The first element will point the tail and also the head.

Now we the remove the last element of the queue and it returns the last element.

If the value which we get is the last node we have to dereference the head.

It will display the linked list is empty after the last value in the queue is poped out

QueueTest:

In this class we are going to add values to the queue which we created previously.

So we created 3 queues q1, q2, q3.

Now we enqueued the values 1, 2, 3 into the q1.

The q2 is just a empty queue.

Similarly now we enqueued the values 1, 2, 3 into the q1.

We dequeued the q1 and checked whether the queue is empty or not and continued the process till we found that q1 is empty, once the queue is emptied we have to dispose the queue.

Now carry the same process with q2 and q3.

Random:

In this class Random we have to use the keyboard presses as noise to seed the random number generator.

We created a function named generateseed().

First we defined variables namely key(char), done(boolean), randomnessCounter(int), seed(int).

The while loop starts with the condition not done which has a nested while loop with the condition key = 0(which means no key is pressed).

We are giving range to the randomnessCounter as minimum of randomnessCounter+3 to 100.

Now add the key to the seed (seed=seed + Maths.charToint(key))

Now initialize the last integer with the seed and the next random integer will be created by LCG.

Return a random integer between the lower and upper ranges.

RandomTest:

In this class we have created a function named test.

We declared variables a, b, c, d, e, f, i, result which are in integer.

Now we are initializing Random function.

The main loop has a while loop which has the condition i<6000 then we are giving value to result as random.randInt(1,7) which means it can have values 1, 2, 3, 4, 5, 6.

The first if loop has the condition if result = 1, a = a+1.

The next if loop has the condition if result = 2, b = b+1.

The next if loop has the condition if result = 3, c = c+1.

The next if loop has the condition if result = 4, d = d+1.

The next if loop has the condition if result = 5, e = e+1.

The last if loop has the condition if result = 6, f = f+1.

After this we will print the output.

Sprites:

In this class we drew the border of the grid using drawborder function .

Also we have created the bitmap code for the following:

- Cursor
- Closed cell
- Mine
- Flag
- Numbered cell
- Zero cell
- One cell
- Two cell
- Three cell
- Four cell
- Five cell
- Six cell
- Seven cell
- Eight cell

Strings:

We used static variables and initialized them so that they don’t take up heap space while reconstucting the string multiple times.

The title is named as minesweeper by team 11 aie-b.

Mine count = mines remaining and it will display the number of mines.

The movement details is also mentioned below in the display area of vm emulator.

Tutorial:

This is displays the information of the game and rules and how to play the game.

On pressing the enter button this text message will disappear and the game will start.

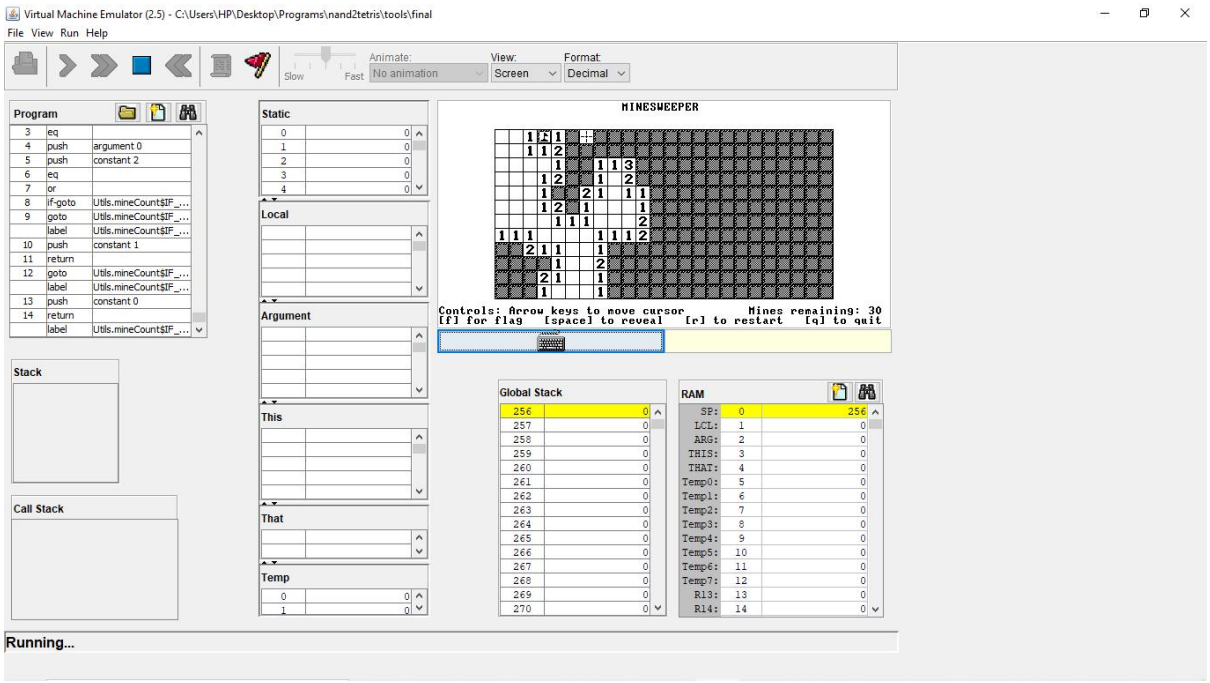
Utils:

This class takes the combination of x, y cell coordinates and converts it into a memory location for the 16x16 draw function.

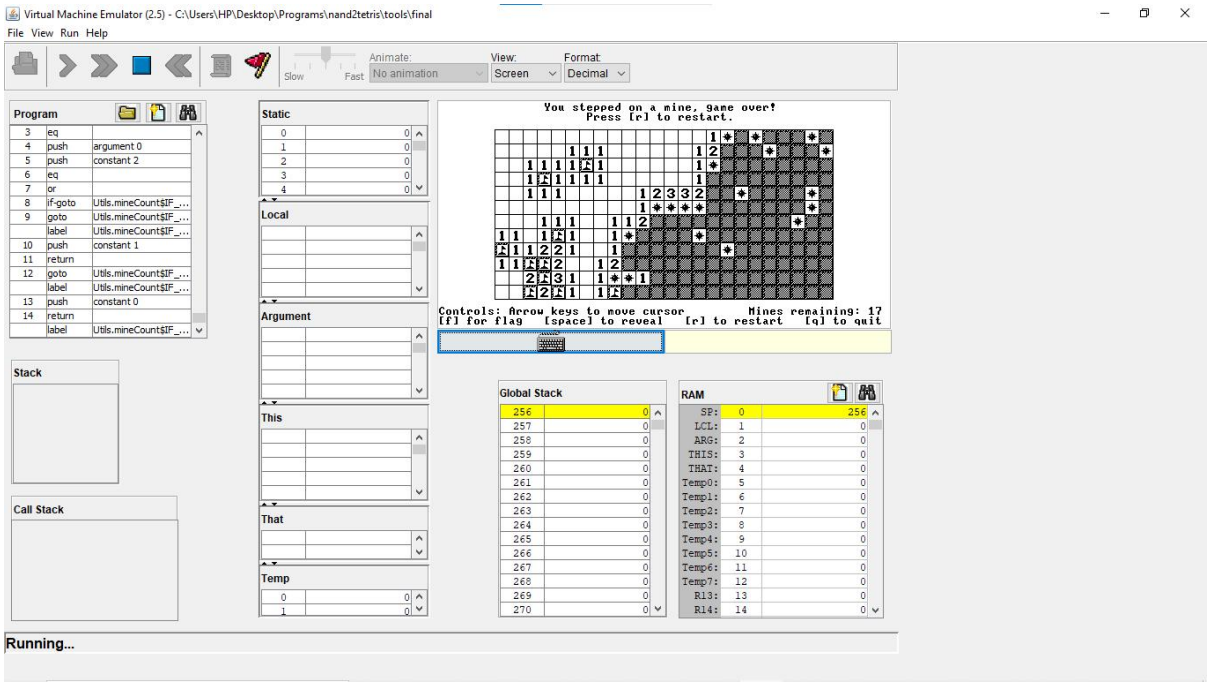
It returns 1 if the given cell value is a mine otherwisse 0.

OUTPUT:

While playing:



Game lost:



CONCLUSION:

This is my final project for the nand2tetris course. It's a Minesweeper game written in the Jack language.The course challenges us to build a full application (in this case, Minesweeper) using a high level language (the Jack language) compiled using a compiler we wrote ourselves, with an operating system we write ourselves, running on a hardware platform (the Hack machine) we designed ourselves, powered by a CPU we wired together ourselves.

FUTURE SCOPE:

We are planning to add few more difficulty level by increasing the number of rows and columns and the number of mines.

Further we planned to add picture which displays after the player wins or loses the game .

REFERENCE:

https://www.youtube.com/watch?v=SpAAYjXAtJ0

https://github.com/idelvall/nand-mines

https://www.geeksforgeeks.org/flood-fill-algorithm/

https://drive.google.com/file/d/1rbHGZV8AK4UalmdJyivgt0fpPiD1Q6Vk/view

The elements of computing systems

Computer architecture A quantitative approach(5th edition)