

“TIC TAC TOE”

A MINI- PROJECT REPORT ON

Submitted in partial fulfillment of the requirements

For the degree of

Bachelor of Engineering

In

Information Technology

by

Name of the Student-1:MAYURESH R. PANDARE Roll No.-19IT5011

Name of the Student-2:SHIVANI D.WAKDE Roll No.-18IT1058

Name of the Student-3: SHRUSHTI S. POLEKAR Roll No.-18IT2021

Supervisor

Nilima Dongre



Department of Information Technology

Dr. D. Y. Patil Group's

Ramrao Adik Institute of Technology

Dr. D. Y. Patil Vidyanagar, Sector 7, Nerul, Navi Mumbai 400706.
(Affiliated to University of Mumbai)

(2020)



Ramrao Adik Institute of Technology

(Affiliated to the University of Mumbai)

Dr. D. Y. Patil Vidyanagar, Sector 7, Nerul, Navi Mumbai 400706.

CERTIFICATE

This is to certify that, Mini Project entitled

“TIC TAC TOE USING BASH SCRIPTING”

is a bonafide work done by

Student Names

1. MAYURESH PANDARE

2. SHIVANI WAKDE

3. SHRUSHTI POLEKAR

and is submitted in the partial fulfillment of the requirement for the
degree of

Bachelor of Engineering

in

Information Technology

to the

University of Mumbai

Supervisor

Prof. Nilima M. Dongre

Project Guide
Nilima Dongre

Head of the department
Dr. Ashish Jadhav

Certificate of Approval by Examiners

This Mini Project report entitled “ Tic Tac Toe ” is a bonafide work done by Student Names under the supervision of Prof.Nilima Dongre approved for the award of Bacheor Degree in Information Technology, University of Mumbai.

Examiners :

1.....

2.....

Supervisors :

1.....

2.....

Principal :

.....

Date :

Place :

DECLARATION

We declare that this written submission represents our ideas in our own words and where others' ideas or words have been included, we have adequately cited and referenced the original sources. We also declare that we have adhered to all principles of academic honesty and integrity and have not misrepresented or fabricated or falsified any idea/data/fact/source in our submission. We understand that any violation of the above will be cause for disciplinary action by the Institute and can also evoke penal action from the sources which have thus not been properly cited or from whom proper permission has not been taken when needed.

Name and Roll No. of Students

Signature

1. MAYURESH R. PANDARE
2. SHIVANI D.WAKDE
3. SHRUSHTI S. POLEKAR

Date:

Place:

ACKNOWLEDGEMENT

The project “Tic Tac Toe” is creative work of many minds. A proper synchronization between individual is must for any project to be completed successfully. One cannot imagine the power of the force that guides us all and neither can we succeed without acknowledging it.

We take this opportunity to express my profound gratitude and deep regards to our Guide **Nilima Dongre**, Department of the Information Technology Engineering for her or her exemplary guidance, monitoring and constant encouragement throughout the completion of this mini project.

We would like to express our gratitude to **Dr. Ashish Jadhav**, Head of the department, Information Technology Engineering for encouraging and inspiring us to carry out the project in the department lab. We take this privilege to express my sincere thanksare thankful to **Dr. Mukesh D. Patil, Principal RAIT**, for his constant support and motivation.

We also would like to thank all the staff members Department of the Information Technology Engineering for providing us with the required facilities and support towards the completion of the project.

Last but not the least we are thankful to our parents and friends for their constant Inspiration, encouragement and well wishes by which we have made a challenging project.

STUDENT- MAYURESH R. PANDARE (19IT5011)

Signature

PREFACE

We take great opportunity to present this Mini Project report on “**TIC TAC TOE**” and put before readers some useful information regarding our project.

We have made sincere attempts and taken every care to present this matter in precise and compact form, the language being as simple as possible. We are sure that the information contained in this volume certainly prove useful for better insight in the scope and dimension of this project in its true perspective.

The task of the completion of the project though being difficult was made quite simple, interesting and successful due to deep involvement and complete dedication of our group members.

TABLE OF CONTENTS

| | |
|--------------------------------|------------|
| Declaration | I |
| Acknowledgement | II |
| Preface | III |
| Table of Contents | IV |
| Table of figures | V |
| Abstract..... | VI |

TABLE OF CONTENTS

| Sr. No. | Topic | Page No. |
|---------|--|----------|
| 1. | INTRODUCTION..... | 8 |
| 1.1 | INTRODUCTION TO SCRIPTING LANGUAGES..... | 10 |
| 1.2 | WHY PARTICULAR SCRIPTING LANGUAGE..... | 11 |
| 1.3 | PROBLEM STATEMENT..... | 12 |
| 1.4 | OBJECTIVES..... | 13 |
| 2. | LITERATURE SURVEY..... | 14 |
| 2.1 | MOTIVATION..... | 15 |
| 3. | PROPOSED SYSTEM..... | 16 |
| 3.1 | INTRODUCTION OF PROPOSED SYSTEM AND ARCHITECTURE..... | 16 |
| 3.2 | HARDWARE AND SOFTWARE REQUIREMENTS..... | 17 |
| 4. | IMPLEMENTATION..... | 21 |
| 4.1 | SYSTEM BLOCK DIAGRAM..... | 21 |
| 4.2 | MODULE DESCRIPTION..... | 22 |
| 4.3 | CODE..... | 26 |
| 5. | RESULT..... | 36 |
| 5.1.1 | OUTPUT SNAPSHOTS | |
| 5.1.2 | TESTING AND VALIDATION | |

| | |
|--|-----------|
| 6. CONCLUSION AND FUTURE SCOPE..... | 37 |
| 6.1 CONCLUSION..... | 37 |
| 6.2 FUTURE SCOPE..... | 37 |
| 6.3 REFERENCES..... | 37 |

ABSTRACT

Our project name is Tic-Tac-Toe game. But the question that comes to our mind is what is exactly Tic Tac Toe?

- WHAT IS TIC TAC TOE?

This game is very popular and is fairly simple by itself. It is actually a two player game. In this game, there is a board with $n \times n$ squares. In our game, it is 3×3 squares. The goal of Tic-Tac-Toe is to be one of the players to get three same symbols in a row - horizontally, vertically or diagonally - on a 3×3 grid .Tic-tac-toe is not a very challenging game for human beings. If you're an enthusiast ,you've probably moved from the basic game to some variant like three dimensional tic-tac-toe on a larger grid. If you sit down right now to play ordinary three-by-three tic-tac-toe with a friend, what will probably happen is that every game will come out a tie. Both you and your friend can probably play perfectly, never making a mistake that would allow your opponent to win. But can you describe how you know where to move each turn? Most of the time ,you probably aren't even aware of alternative possibilities; you just look at the board and instantly know where you want to move. That kind of instant knowledge is great for human beings, because it makes you a fast player. But it isn't much help in writing a computer program. For that, you have to know very explicitly what your strategy is.

CHAPTER -1

INTRODUCTION

INTRODUCTION

1.1 INTRODUCTION TO SCRIPTING LANGUAGES

Usually shells are interactive that mean, they accept command as input from users and execute them. However some time we want to execute a bunch of commands routinely, so we have type in all commands each time in terminal. Shell scripts are similar to the batch file in MS-DOS. Each shell script is saved with .sh file extension eg .myscript.sh A shell script have syntax just like any other programming language. If you have any prior experience with any programming language like Python, C/C++ etc. it would be very easy to get started with it.

1.2 WHY PARTICULAR SCRIPTING LANGUAGE

- There are many reasons to write shell scripts –
- To avoid repetitive work and automation
- System admins use shell scripting for routine backups
- System monitoring
- Adding new functionality to the shell etc.

1.3 PROBLEM STATEMENT

- Prone to costly errors, a single mistake can change the command which might be harmful
- Slow execution speed
- Design flaws within the language syntax or implementation
- Not well suited for large and complex task

1.4 OBJECTIVES

- The command and syntax are exactly the same as those directly entered in command line, so programmer do not need to switch to entirely different syntax: Writing shell scripts are much quicker
- Quick start

CHAPTER -2
LITERATURE SURVEY

2.1 MOTIVATION

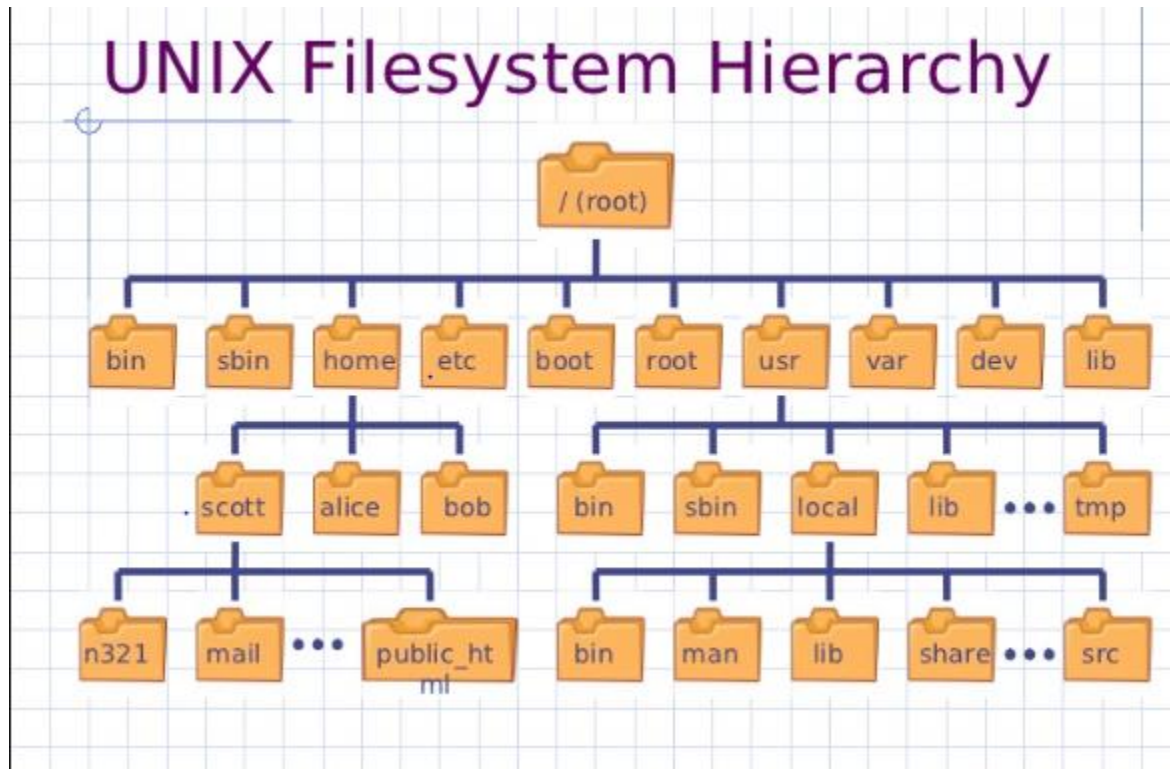
A player can play a perfect game of tic-tac-toe (to win or, at least, draw) if each time it is his turn to play he chooses the first available move from the following list, as used in Newell and Simon's 1972 tic-tac-toe program. [15]

1. Win: If the player has two in a row, they can place a third to get three in a row.
2. Block: If the opponent has two in a row, the player must play the third themselves to block the opponent.
3. Fork: Create an opportunity where the player has two threats to win (two non-blocked lines of 2).
4. Blocking an opponent's fork: If there is only one possible fork for the opponent, the player should block it. Otherwise, the player should block any forks in any way that simultaneously allows them to create two in a row. Otherwise, the player should create a two in a row to force the opponent into defending, as long as it doesn't result in them creating a fork.
5. Center: A player marks the center. (If it is the first move of the game, playing on a corner gives the second player more opportunities to make a mistake and may therefore be the better choice; however, it makes no difference between perfect players.)
6. Opposite corner: If the opponent is in the corner, the player plays the opposite corner.
7. Empty corner: The player plays in a corner square.
8. Empty side: The player plays in a middle square on any of the 4 sides.

CHAPTER -3
PROPOSED SYSTEM

3.1 INTRODUCTION OF PROPOSED SYSTEM AND ARCHITECTURE

The File Operations are performed over the ubuntu operating system and there are various types of directories present in this system.



The main components here are:

1. /boot : Contains the boot loader
2. /home : Contains the home directories of users.
3. /bin : All the executable binaries and commands used by all the users on the system are located here.
4. /sbin : This contains the system executable binaries typically used by system administrators.
5. /lib : Contains the system libraries that support the binaries in /bin and /sbin.
6. /etc : Contains the configuration files for network, boot-time, etc.
7. /dev : This has the device files i.e. usb, terminal device or any other device attached to system are shown here.

8. /proc : Contains information about the process running.

9. /tmp : This is the temporary directory where many processes create the temporary files required.

This is purged each time the machine is booted.

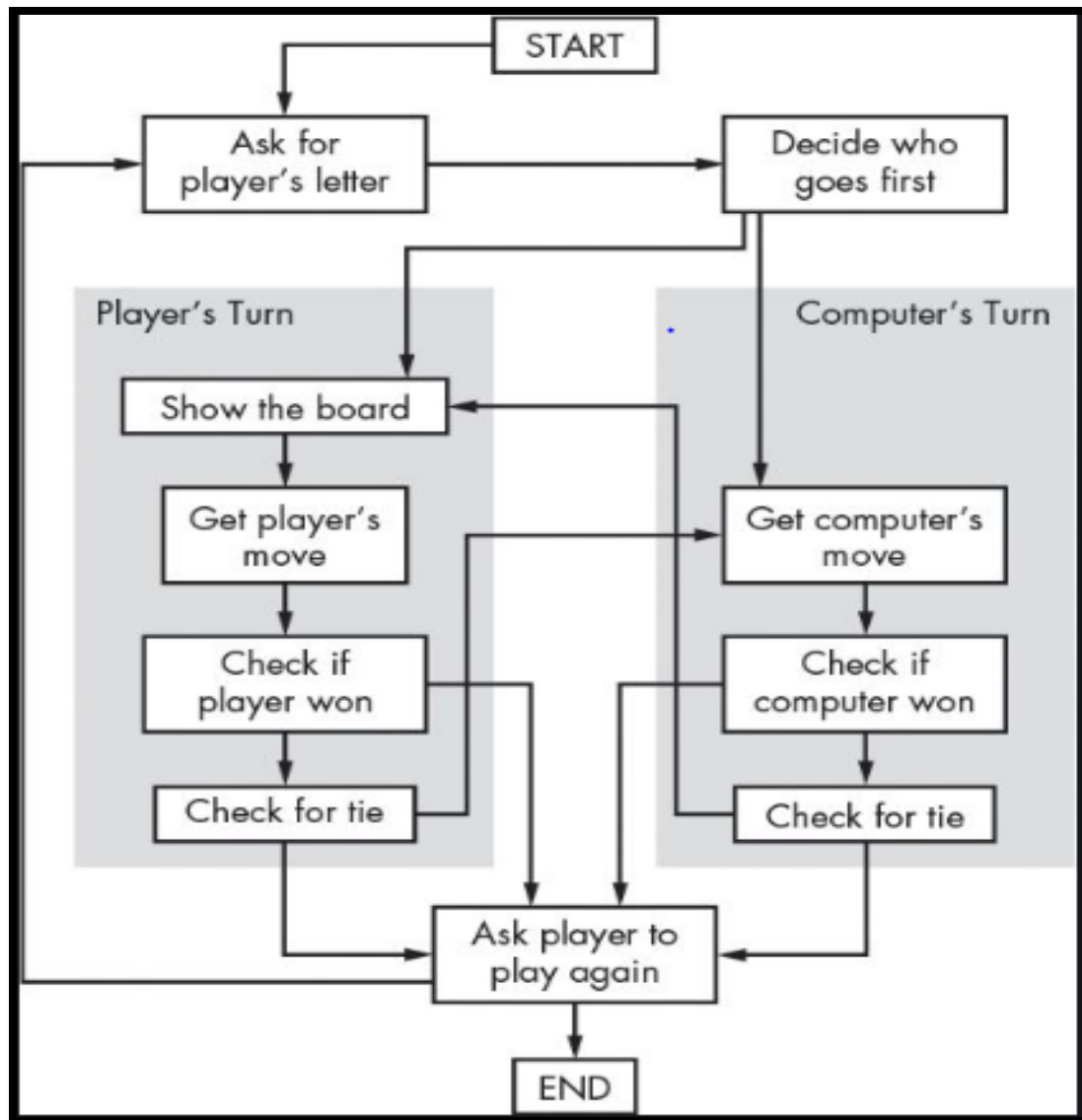
3.2 HARDWARE AND SOFTWARE REQUIREMENTS

- Shell can be accessed by user using a command line interface. A special program called Terminal in linux/macOS or Command Prompt in Windows OS is provided to type in the human readable commands such as “cat”, “ls” etc. and then it is being execute. The result is then displayed on the terminal to the user. It will list all the files in current working directory in long listing format. Working with command line shell is bit difficult for the beginners because it’s hard to memorize so many commands. It is very powerful, it allows user to store commands in a file and execute them together. This way any repetitive task can be easily automated. These files are usually called batch files in Windows and Shell Scripts in Linux/macOS systems
- Graphical shells provide means for manipulating programs based on graphical user interface (GUI), by allowing for operations such as opening, closing, moving and resizing windows, as well as switching focus between windows. Window OS or Ubuntu OS can be considered as good example which provide GUI to user for interacting with program. User do not need to type in command for every actions. There are several shells are available for Linux systems like – BASH (Bourne Again SHell) – It is most widely used shell in Linux systems. It is used as default login shell in Linux systems and in macOS. It can also be installed on Windows OS. CSH (C SHell) – The C shell’s syntax and usage are very similar to the C programming language. KSH (Korn SHell) – The Korn Shell also was the base for the POSIX Shell standard specifications etc. Each shell does the same job but understand different commands and provide different built in functions.

CHAPTER –4

IMPLEMENTATION

4.1 SYSTEM BLOCK DIAGRAM



4.2 MODULE DESCRIPTION

If any player is able to draw three X s or three O s in the following combinations then that player wins. The combinations are:

a) 0,1,2

b) 3,4,5

c) 6,7,8

d) 0,4,8

e) 2,4,6

f) 0,3,6

g) 1,4,7

h) 2,5,8

A player can play a perfect game of tic-tac-toe (to win or, at least, draw) if each time it is his turn to play he chooses the first available move from the following list, as used in Newell and Simon's 1972 tic-tac-toe program.

1. Win: If the player has two in a row, they can place a third to get three in a row.
2. Block: If the opponent has two in a row, the player must play the third themselves to block the opponent.
3. Fork: Create an opportunity where the player has two threats to win (two non-blocked lines of 2).
4. Blocking an opponent fork: If there is only one possible fork for the opponent, the player should block it. Otherwise, the player should block any forks in any way that simultaneously allows them to create two in a row. Otherwise, the player should create a two in a row to force the opponent into defending, as long as it doesn't result in them creating a fork.
5. Center: A player marks the center. (If it is the first move of the game, playing on a corner gives the second player more opportunities to make mistake and may therefore be the better choice; however, it makes no difference between perfect players.)
6. Opposite corner: If the opponent is in the corner, the player plays the opposite corner.
7. Empty corner: The player plays in a corner square.
8. Empty side: The player plays in a middle square on any of the 4 sides.

4.3 CODE

```
#!/bin/bash

reset(){

    Arr=('.' '.' '.' '.' '.' '.' '.' '.' '.')
    player=1
    gamestatus=1
    echo "-----WELCOME-----"
    echo "Let's start the game"
}

set(){
    idx=$(( $1 * 3 + $2 ))
    if [ ${Arr[$idx]} == "." ]; then
        Arr[$idx]=$3
        player=$((player%2+1))
    else
        echo "You can't place there!"
    fi
}

print(){
    echo "r\c 0 1 2"
    echo "0   ${Arr[0]} ${Arr[1]} ${Arr[2]}"
    echo "1   ${Arr[3]} ${Arr[4]} ${Arr[5]}"
    echo "2   ${Arr[6]} ${Arr[7]} ${Arr[8]}"
}

checkmatch(){
    if [ ${Arr[$1]} != "." ] && [ ${Arr[$1]} == ${Arr[$2]} ] && [ ${Arr[$2]}
== ${Arr[$3]} ]; then
        gamestatus=0
    fi
}

checkgame(){
    checkmatch 0 1 2
    checkmatch 3 4 5
    checkmatch 6 7 8
    checkmatch 0 3 6
    checkmatch 1 4 7
    checkmatch 2 5 8
    checkmatch 0 4 8
    checkmatch 2 4 6
}

reset
# infinite game loop
var=0
while [ 1 == 1 ] && [ $var -lt 9 ]; do
```

```

echo ""
if [ $player == 1 ]; then sym=O
else sym=X;
fi
echo "Player $player's turn: ($sym)"
print
echo ""
echo "  Command:"
echo "      1. set [row] [column]"
echo "      2. restart"
while [ 1 == 1 ]; do
    read -r cmd a b
    if [ $cmd == "set" ]; then
        set $a $b $sym
        var=$(( $var+1 ))
        break
    elif [ $cmd == "restart" ]; then
        var=0
        reset
        break
    else
        echo "Wrong input"
    fi
done
checkgame
if [ $gamestatus != 1 ]; then
    echo "Gameover"
    player=$((player%2+1))
    echo "Player $player ($sym) win!!"
    while [ 1 == 1 ]; do
        read -r cmd n
        if [ $cmd == "restart" ]; then
            reset
            break
        fi
    done
fi
done
fi
if [ $var -ge 9 ]; then
echo "Sorry, your match is draw"
fi

```

CHAPTER –5

RESULT

5.1.1 OUTPUT SNAPSHOTS

1.

```
GNU bash, version 4.4.12(1)-release (x86_64-pc-linux-gnu)
❯ bash main.sh
-----WELCOME-----
Let's start the game

Player 1's turn: (O)
r\c 0 1 2
0   . . .
1   . . .
2   . . .

Command:
  1. set [row] [column]
  2. restart
set 0 0

Player 2's turn: (X)
r\c 0 1 2
0   O . .
1   . . .
2   . . .

Command:
  1. set [row] [column]
  2. restart
set 1 1

Player 1's turn: (O)
r\c 0 1 2
0   O . .
1   . X .
2   . . .
```


2.

```
Player 1's turn: (O)
r\c 0 1 2
0   O . .
1   . X .
2   . . .

Command:
  1. set [row] [column]
  2. restart
set 2 0
```

```
Player 2's turn: (X)
r\c 0 1 2
0   O . .
1   . X .
2   O . .

Command:
  1. set [row] [column]
  2. restart
set 2 1
```

```
Player 1's turn: (O)
r\c 0 1 2
0   O . .
1   . X .
2   O X .

Command:
  1. set [row] [column]
  2. restart
set 1 0
Gameover
Player 1 (O) win!!
```

3.

```
-----WELCOME-----
Let's start the game

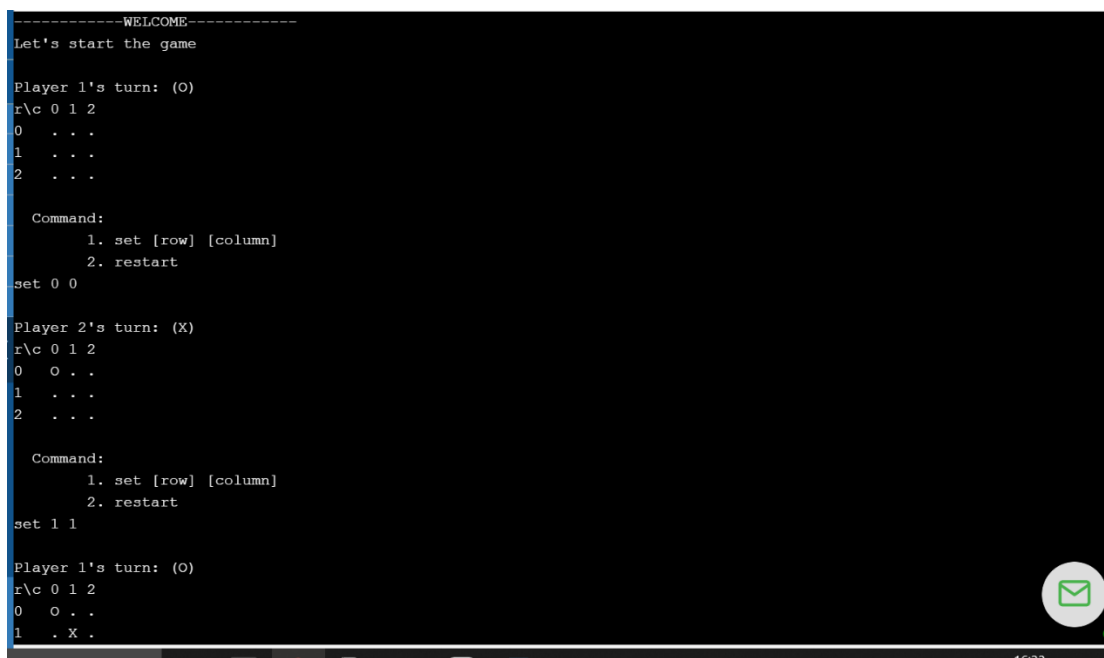
Player 1's turn: (O)
r\c 0 1 2
0  . . .
1  . . .
2  . . .

Command:
    1. set [row] [column]
    2. restart
set 0 0

Player 2's turn: (X)
r\c 0 1 2
0  O . .
1  . . .
2  . . .

Command:
    1. set [row] [column]
    2. restart
set 1 1

Player 1's turn: (O)
r\c 0 1 2
0  O . .
1  . X .
```



4.

```
input
Command:
  1. set [row] [column]
  2. restart
set 0 2

Player 2's turn: (X)
\c 0 1 2
  0 . 0
  . X .
  . 0 X

Command:
  1. set [row] [column]
  2. restart
set 2 0

Player 1's turn: (O)
\c 0 1 2
  0 . 0
  . X .
  X 0 X

Command:
  1. set [row] [column]
  2. restart
set 1 0

Player 2's turn: (X)
\c 0 1 2
```

5.

```
2. restart
set 1 0

Player 2's turn: (X)
r\c 0 1 2
0  o . o
1  o X .
2  X o X

Command:
1. set [row] [column]
2. restart
set 0 1

Player 1's turn: (O)
r\c 0 1 2
0  o X o
1  o X .
2  X o X

Command:
1. set [row] [column]
2. restart
set 1 2
Sorry, your match is draw

...Program finished with exit code 0
Press ENTER to exit console.
```

5.2 Testing and validation

Testing Details

Testing is a verification process for quality assessment and improvement. Testing is basically done to find errors, faults in the system. The basic goal of software development process is to produce the software that has very few or no errors. In an effort to detect errors soon after they are introduced each phase ends with verification activity such as reviews. However most of these verification activities in the early phase of the software development are based on human evaluation and cannot detect all the errors.

Testing plays an important role in quality assurance for the software. It is a dynamic method for the verification and validation, where the system to be tested is executed and the behavior of the system is observed.

14.1.1 Black Box Testing

Black Box Testing is also known as Behavioral Testing, is a software testing method in which the internal structure/ design/ implementation of the item being tested is not known to the tester. These tests can be functional or non-functional, though usually functional. This method is named so because the software program, in the eyes of the tester, is like a black box; inside which one cannot see.

14.1.2 White Box Testing

White Box Testing (also known as Clear Box Testing, Open Box Testing, Glass Box Testing,

Transparent Box Testing, Code-Based Testing or Structural Testing) is a software testing method in which the internal structure/design/implementation of the item being tested is known to the tester. The tester chooses inputs to exercise paths through the code and determines the appropriate outputs. Programming know-how and the implementation knowledge is essential. White box testing is testing beyond the user interface and into the nitty-gritty of a system.

14.1.3 Unit Testing

Unit Testing is a level of the software testing process where individual units/components of a

software/system are tested. The purpose is to validate that each unit of the software performs as designed. A unit is the smallest testable part of software. It usually has one or a few inputs and usually a single output. In procedural programming a unit may be an individual program, function,

procedure, etc. In object-oriented programming, the smallest unit is a method, which may belong to a base/super class, abstract class or derived/child class.

14.1.4 Integration Testing

Integration is the process by which components are aggregated to create larger components. Testing the data flow or interface between two features is known as integration testing. Testing that occurs at lowest level is called unit/ module testing. As the units are tested and low level bugs are found and fixed, they are integrated and integration testing is performed against groups of modules. Integration Testing is also called as Structural Testing.

14.1.5 System Testing

In system testing the behavior of whole system/product is tested as defined by the scope of the development project or product. It may include tests based on risks and/or requirement specifications, business process, use cases, or other high level descriptions of system behavior, interactions with the operating systems, and system resources. System testing is most often the final test to verify that the system to be delivered meets the specification and its purpose. System testing is carried out by specialist's testers or independent testers. System testing should investigate both functional and non- functional requirements of the testing.

14.1.6 Alpha Testing

Alpha testing is one of the most common software testing strategy used in software development. It's specially used by product development organizations. This test takes place at the developer's site. Developers observe the users and note problems. Alpha testing is typically performed by a group that is independent of the design team, but still within the company, e.g. in-house software test engineers, or software QA engineers. Alpha testing is final testing before the software is released to the general public.

It has two phases:

1. In the first phase of alpha testing, the software is tested by in-house developers. They use either debugger software, or hardware-assisted debuggers. The goal is to catch bugs quickly.
2. In the second phase of alpha testing, the software is handed over to the software QA staff, for additional testing in an environment that is similar to the intended use.

CHAPTER –6

CONCLUSION AND FUTURE SCOPE

6.1 CONCLUSION

In the end, we would like to conclude that our aim was to make tic tactoe game using bash script. Our code is written for two player version i.e Human vs human. The final outcome (win, lose, draw) was the only information available regarding the quality of the play.

6.2 FUTURE SCOPE

The game is to be played between two people .One of the player chooses 'O' and the other 'X' to mark their respective cells. The game starts with one of the players and the game ends when one of the players has one whole row/ column/ diagonal filled with his/her respective character ('O' or 'X').If no one wins, then the game is said to be draw.

6.3 REFERENCES

- www.google.com
- www.wikipedia.org
- www.tutorialpoint.com
- www.w3schools.com
- www.youtube.com
- <https://www.tutorialspoint.com/unix/unix-file-operators.html>
- <http://heim.ifi.uio.no/gisle/staging2/drupalprimer/unix/unix03.html>

