TIC TAC TOE

A Project Report

Submitted by

SHREYAS DAS

In partial fulfillment of the

CBSE GRADE XI

IN

Computer Science

At

AECS MAGNOLIA MAARUTI PUBLIC SCHOOL

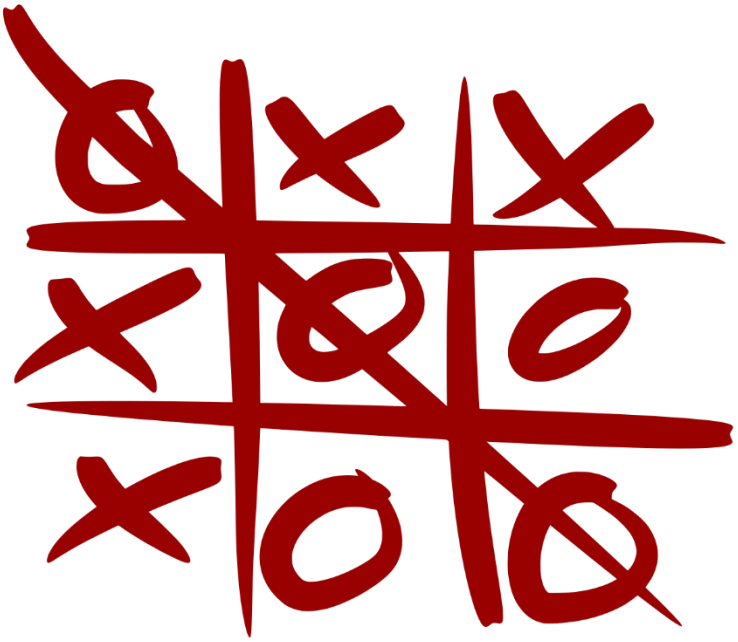
Arakere, Off Bannerghatta Road, Bangalore-560076

2019-20

CERTIFICATE

This is to certify that SHREYAS DAS of Grade XI, AECS MAGNOLIA MAARUTI PUBLIC SCHOOL, BANGALORE with Roll Number 29 has satisfactorily completed the project in Computer Science on “TIC TAC TOE” in partial fulfillment of the requirements of All Indian Secondary School Certificate Examination (AISSCE) as prescribed by CBSE in the year 2019-20.

TIC TAC TOE



ACKNOWLEDGEMENT

I warmly acknowledge the continuous encouragement and timely suggestions offered by our dear Principal Dr. Seema Goel. I extend my hearty thanks for giving me the opportunity to make use of the facilities available in the campus to carry out the project successfully.

I am highly indebted to Mrs. Jean Mathew for the constant supervision, providing necessary information and supporting in completing the project. I would like to express my gratitude towards them for their kind co-operation and encouragement.

Finally I extend my gratefulness to one and all who are directly or indirectly involved in the successful completion of this project work.

TABLE OF CONTENTS

INTRODUCTION

Tic-tac-toe (also known as noughts and crosses or Xs and Os) is a paper and a pencil for two players, X and O, who take turn marking the spaces in a 3×3 grid. The player who succeeds in placing three of their marks in a horizontal, vertical, or diagonal row wins the game.

In this project, we are using GUI (Graphical User Interface) to design this project. We use Python programming language for coding. This is a multiplayer game .

SYNOPSIS

In our program the moves taken by the human and the human are chosen randomly.

Winning Strategy – An Interesting Fact

If both the players play optimally then it is destined that you will never lose ("although the match can still be drawn"). It doesn't matter whether you play first or second. In another ways – "Two expert players will always draw".

PROCEDURE

- Importing the module – tkinter
- Create the main window (container)
- Add any number of widgets to the main window
- Apply the event Trigger on the widgets.
- Importing tkinter is same as importing any other module in Python.

REQUIREMENT ANALYSIS AND DESIGN

HARDWARE REQUIREMENTS:

- Processor : Intel Core i5 processor
- CPU Speed : 2.67 GZ
- RAM : 4.00 GB
- Hard Disk Memory : 80 GB

SOFTWARE REQUIREMENTS:

- Operating Systems : Windows 10, Linux
- Software : Python 3.6.X

MODULES AND FUNCTIONS

Widget	Usage	Syntax
C_disable()	Disable all buttons	-
C_enable()	Enable all buttons	-
Check_win()	Check if player has won	-
Click(player_ Symbol,number)	Updates the board when the user clicks	-
Player_chooser()	Chooses player at start and next	-
Disable(number)	Disables a specific button	-
Score_board()	Displays and updates the score	-
Clear()	Clears the value of board and resets the game, Except score	-
Button	To add a button to your application	W=(master, option=value)
Command	To call a function	-
Width	To set width of a button	-

Height	To set height of a button	-
Grid	It organizes the widgets in grid (table-like structure) before placing in the parent widget	Button.grid(X,Y)
Title	To set title of the widget	-
Label	The display box where you can put any text or image which can be updated anytime as per the code	W=Label (master,option =value)
Font	To set the font on the label button	-
Text	To edit a multi line text and format the way it has to be deployed	W=Text (master,option =value)
Image	To set the image on the widget	-

ABOUT PYTHON

Introduction

Python is a popular programming language. It was created by Guido van Rossum, and released in 1991.

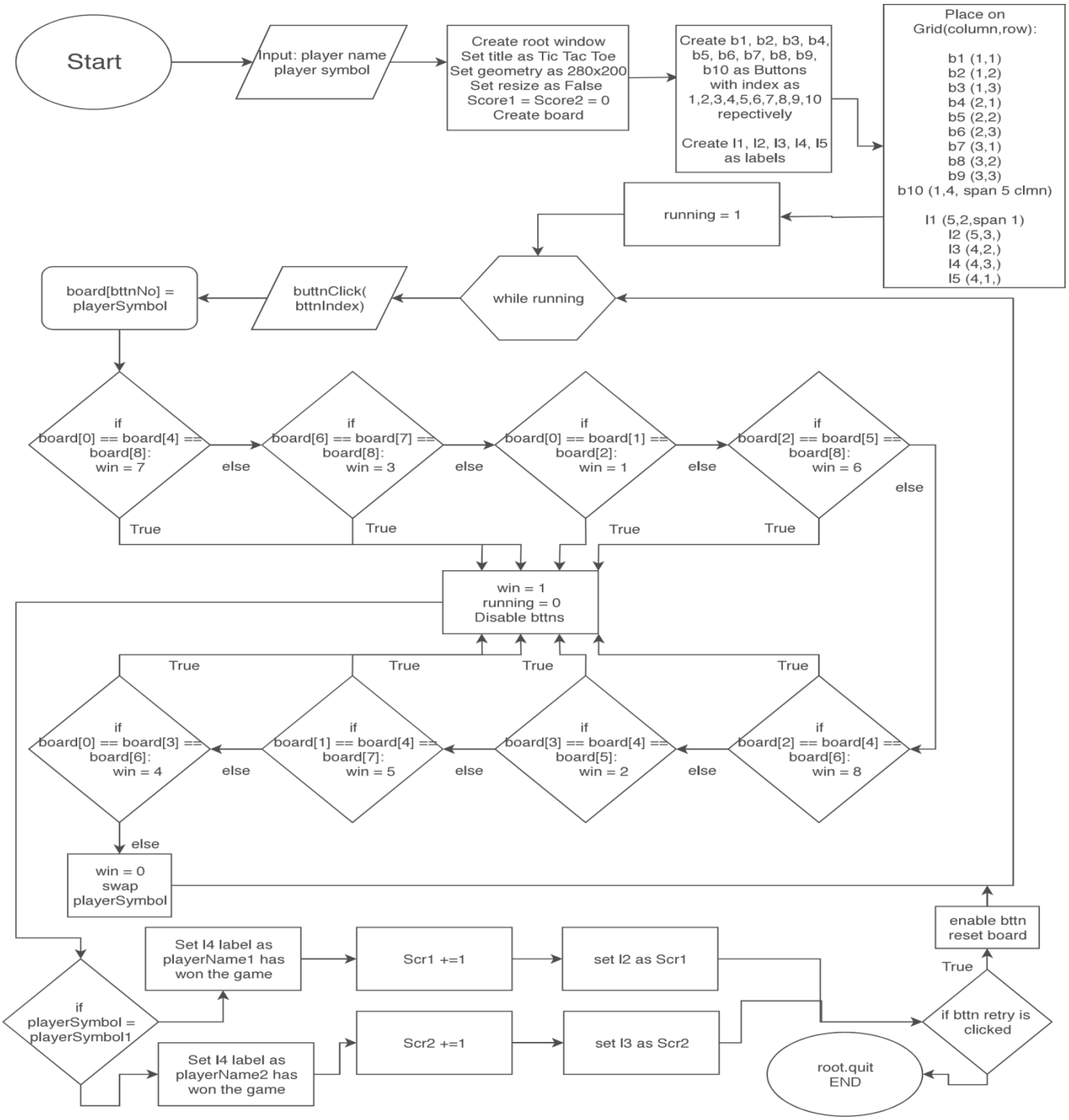
Advantages

- Python works on different platforms (Windows, Mac, Linux, Raspberry Pi, etc).
- Python has a simple syntax similar to the English language.
- Python runs on an interpreter system, meaning that code can be executed as soon as it is written. This means that prototyping can be very quick.
- Python can be treated in a procedural way, an objectorientated way or a functional way.

Applications

- Python can be used on a server to create web applications.
- Python can be used alongside software to create workflows.
- Python can connect to database systems. It can also read and modify files.
- Python can be used to handle big data and perform complex mathematics.
- Python can be used for rapid prototyping, or for production-ready software development.

FLOWCHART


Start
Input: player name
player symbol
Create root window
Set title as Tic Tac Toe
Set geometry as 280x200
Set resize as False
Score1 = Score2 = 0
Create board
Create b1, b2, b3, b4,
b5, b6, b7, b8, b9,
b10 as Buttons
with index as
1,2,3,4,5,6,7,8,9,10
repectively
Create l1, l2, l3, l4, l5
as labels
Place on
Grid(column,row):
b1 (1,1)
b2 (1,2)
b3 (1,3)
b4 (2,1)
b5 (2,2)
b6 (2,3)
b7 (3,1)
b8 (3,2)
b9 (3,3)
b10 (1,4, span 5 clmn)
l1 (5,2,span 1)
l2 (5,3,)
l3 (4,2,)
l4 (4,3,)
l5 (4,1,)
running = 1
while running
buttnClick(
bttnIndex)
board[bttnNo] =
playerSymbol
if
board[0] == board[4] ==
board[8]:
win = 7
else
if
board[6] == board[7] ==
board[8]:
win = 3
else
if
board[0] == board[1] ==
board[2]:
win = 1
else
if
board[2] == board[5] ==
board[8]:
win = 6
else
True
True
True
True
win = 1
running = 0
Disable bttns
True
True
True
True
if
board[0] == board[3] ==
board[6]:
win = 4
else
if
board[1] == board[4] ==
board[7]:
win = 5
else
if
board[3] == board[4] ==
board[5]:
win = 2
else
if
board[2] == board[4] ==
board[6]:
win = 8
else
win = 0
swap
playerSymbol
if
playerSymbol =
playerSymbol1
Set l4 label as
playerName1 has
won the game
Scr1 +=1
set l2 as Scr1
Set l4 label as
playerName2 has
won the game
Scr2 +=1
set l3 as Scr2
enable bttn
reset board
True
if bttn retry is
clicked
root.quit
END

PROGRAM SOURCE CODE

```
from tkinter import *

player1_name = input("Enter Player 1 name: ")
player2_name = input("Enter Player 2 name: ")
player1_symbol = input("Enter Player 1 symbol: ")
player2_symbol = input("Enter Player 2 symbol: ")
root = Tk()
root.title('Tic Tac Toe')
root.geometry('280x200')
root.resizable(False, False)
Score1 = 0
Score2 = 0
board = [' ', ' ', '  ', '   ', '    ', '     ', '      ', '       ', '        ', '']
swapper = 0
check_draw = 0
winner = '...'
quote_res = ' has won the game, '
quote = ' '
win = 0

def c_disable():
    b1.config(state=DISABLED)
    b2.config(state=DISABLED)
    b3.config(state=DISABLED)
    b4.config(state=DISABLED)
    b5.config(state=DISABLED)
    b6.config(state=DISABLED)
    b7.config(state=DISABLED)
    b8.config(state=DISABLED)
    b9.config(state=DISABLED)
```

```
def c_enable():
    b1.config(state=NORMAL)
    b2.config(state=NORMAL)
    b3.config(state=NORMAL)
    b4.config(state=NORMAL)
    b5.config(state=NORMAL)
    b6.config(state=NORMAL)
    b7.config(state=NORMAL)
    b8.config(state=NORMAL)
    b9.config(state=NORMAL)


def check_win():
    global Score1
    global Score2
    global swapper
    global player1_name
    global player2_name
    global quote
    global quote_res
    global win
    if board[0] == board[1] == board[2]:
        win = 1
elifboard[3] == board[4] == board[5]:
        win = 2
elifboard[6] == board[7] == board[8]:
        win = 3

elifboard[0] == board[3] == board[6]:
        win = 4
elifboard[1] == board[4] == board[7]:
        win = 5
elifboard[2] == board[5] == board[8]:
        win = 6
```

```
elifboard[0] == board[4] == board[8]:
        win = 7
elifboard[2] == board[4] == board[6]:
        win = 8

    if win in range(1,9):
c_disable()
    if swapper == 0 and win in range(1, 9):
        quote = quote_res
        Score1 += 1
        b10['text'] = str(player1_name) + quote + 'Retry'
elif swapper == 1 and win in range(1, 9):
        quote = quote_res
        Score2 += 1
        b10['text'] = str(player2_name) + quote + 'Retry'


def click(player_symbol, number):
    global check_draw
    global winner
    global quote
    global quote_res
    board[number] = player_symbol
    b1['text'] = board[0]
    b2['text'] = board[1]
    b3['text'] = board[2]

    b4['text'] = board[3]
    b5['text'] = board[4]
    b6['text'] = board[5]

    b7['text'] = board[6]
    b8['text'] = board[7]
    b9['text'] = board[8]
```

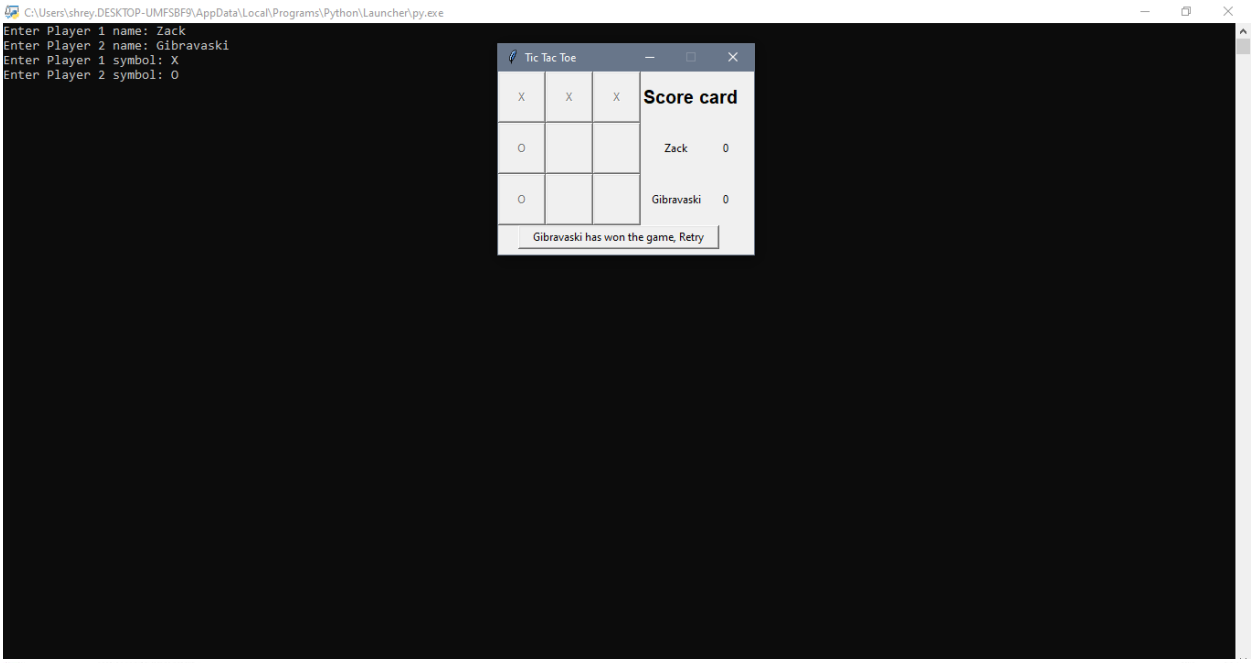
```
    disable(number)
check_draw += 1
check_win()
    if check_draw == 9:
        b10['text'] = "It's a draw, Retry?"


def player_chooser():
    global swapper
    global player1_symbol
    global player2_symbol
    if swapper == 0:
        swapper = 1
        return player1_symbol
    else:
        swapper = 0
        return player2_symbol


def disable(number):
    if number == 0:
        b1.config(state=DISABLED)
    if number == 1:
        b2.config(state=DISABLED)
    if number == 2:
        b3.config(state=DISABLED)
    if number == 3:
        b4.config(state=DISABLED)
    if number == 4:
        b5.config(state=DISABLED)
    if number == 5:
        b6.config(state=DISABLED)
    if number == 6:
        b7.config(state=DISABLED)
```

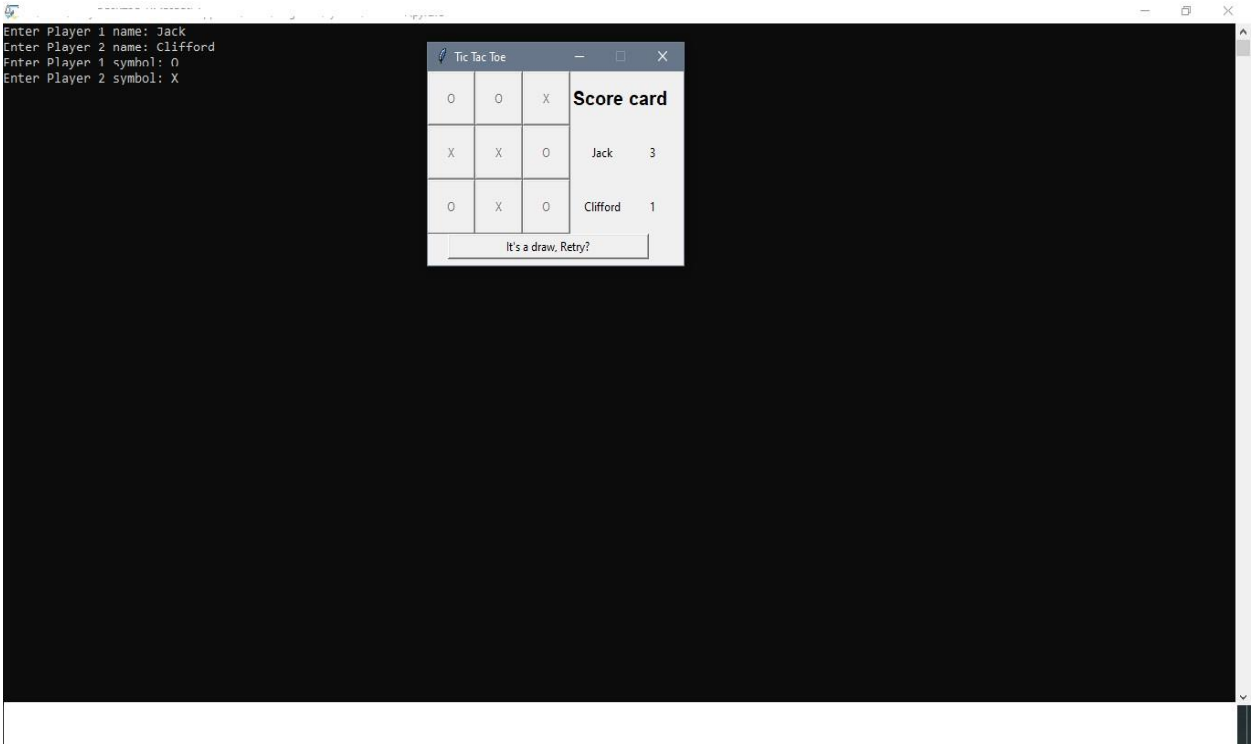
```
    if number == 7:

        b8.config(state=DISABLED)
    if number == 8:
        b9.config(state=DISABLED)


def score_board():
    global swapper
    if swapper == 0:
        l1['text'] += 1
elif swapper == 1:
        l2['text'] += 1


def clear():
    global Score1
    global Score2
    global board
    global check_draw
    global winner
    global quote_res
    global quote
    global win
    win = 0
check_draw = 0
    winner = '...'
quote_res = ' has won the game, '
    quote = ' '
    board = [' ', '  ', '   ', '    ', '     ', '      ', '       ', '        ', '']
    b1['text'] = board[0]
    b2['text'] = board[1]
    b3['text'] = board[2]
    b4['text'] = board[3]
    b5['text'] = board[4]
```

```
    b6['text'] = board[5]

    b7['text'] = board[6]
    b8['text'] = board[7]
    b9['text'] = board[8]
    b10['text'] = 'Reset'
    l1['text'] = str(Score1)
    l2['text'] = str(Score2)
c_enable()


b1 = Button(root, text=board[0], width=6, height=3, command=lambda:
click(player_chooser(), 0))
b2 = Button(root, text=board[1], width=6, height=3, command=lambda:
click(player_chooser(), 1))
b3 = Button(root, text=board[2], width=6, height=3, command=lambda:
click(player_chooser(), 2))
b4 = Button(root, text=board[3], width=6, height=3, command=lambda:
click(player_chooser(), 3))
b5 = Button(root, text=board[4], width=6, height=3, command=lambda:
click(player_chooser(), 4))
b6 = Button(root, text=board[5], width=6, height=3, command=lambda:
click(player_chooser(), 5))
b7 = Button(root, text=board[6], width=6, height=3, command=lambda:
click(player_chooser(), 6))
b8 = Button(root, text=board[7], width=6, height=3, command=lambda:
click(player_chooser(), 7))
b9 = Button(root, text=board[8], width=6, height=3, command=lambda:
click(player_chooser(), 8))
b10 = Button(root, text='Reset', width=30, height=1, command=lambda:
clear())
l1 = Label(root, text=str(Score2))
l2 = Label(root, text=str(Score1))
l3 = Label(root, text=str(player1_name))
l4 = Label(root, text=str(player2_name))
```

```
l5 = Label(root, text='Score card', anchor='n', font='NONE 15 bold')

b1.grid(column=1, row=1)
b2.grid(column=1, row=2)
b3.grid(column=1, row=3)
b4.grid(column=2, row=1)
b5.grid(column=2, row=2)
b6.grid(column=2, row=3)
b7.grid(column=3, row=1)
b8.grid(column=3, row=2)
b9.grid(column=3, row=3)
b10.grid(column=1, row=4, columnspan=5)
l1.grid(column=5, row=2, rowspan=1)
l2.grid(column=5, row=3, rowspan=1)
l3.grid(column=4, row=2, rowspan=1)
l4.grid(column=4, row=3, rowspan=1)
l5.grid(column=4, row=1, columnspan=2)
mainloop()
```

OUTPUT SCREEN





FUTURE ENHANCEMENTS

- The program can be easily modified so that both players play optimally (which will fall under the category of Artificial Intelligence).
- Also, the program can be modified such that the user himself gives the input (using scanf() or cin).

BIBLIOGRAPHY

https://www.youtube.com
https://google.com
https://www.geeksforgeeks.org
https://www.xda-developers.com