APPENDIX 1

SNAKE AND LADDERS

END TERM REPORT

by

Dheeraj Kumar rai, Manohar Kumar thakur, Nehpal Singh

Section: K19PG

Roll Numbers: 62,63,65

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

November, 2020

APPENDIX 2

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.

Signature: (Due to Covid 19, signature is exempted)
Name: Dheeraj Kumar rai , Manohar Kumar thakur, Nehpal Singh

Roll Number: 62,63,65

Place: Phagwara
Date: 25/10/2020

APPENDIX 3

TABLE OF CONTENTS

APPENDIX 4

BONAFIDE CERTIFICATE

Certified that this project report “SNAKE AND LADDERS” is the bonafide work of “Dheeraj Kumar rai, Manohar Kumar thakur, Nehpal Singh” who carried out the project work under my supervision.

Signature of the Supervisor(Due to Covid 19, signature is exempted)

Dr Dhanpratap singh

Associate professor

Intelligence system

Background and objectives of the project assigned

ABOUT THE GAME:

Snakes and Ladders, known originally as Moksha Patam, is an ancient Indian board game for two or more players regarded today as a worldwide classic. It is played on a game board with numbered, gridded squares. A number of "ladders" and "snakes" are pictured on the board, each connecting two specific board squares. The object of the game is to navigate one's game piece, according to die rolls, from the start (bottom square) to the finish (top square), helped by climbing ladders but hindered by falling down snakes.

The game is a simple race based on sheer luck, and it is popular with young children. The historic version had its roots in morality lessons, on which a player's progression up the board represented a life journey complicated by virtues (ladders) and vices (snakes). The game is also sold under other names such as Chutes and Ladders, Bible Ups and Downs, etc., some with a morality motif a morality Chutes and Ladders was published by Milton Bradley starting from 1943.

When the game was brought to England, the Indian virtues and vices were replaced by English ones in hopes of better reflecting Victorian doctrines of morality. Squares of Fulfillment, Grace and Success were accessible by ladders of Thrift, Penitence and Industry and snakes of Indulgence, Disobedience and Indolence caused one to end up in Illness, Disgrace and Poverty. While the Indian version of the game had snakes outnumbering ladders, the English counterpart was more forgiving as it contained each in the same amount. This concept of equality signifies the cultural ideal that for every sin one commits, there exists another chance at redemption.

Snake and ladder is a basic game which can be played by anyone old or young. In this project we have to program this game using python tkinter library using GUI concepts. In this there is a board I which there are 30 blocks and on that blocks there are some hurdles or shortcuts like snake which can take the player from its mouth to its tail if player touches its mouth and we have ladders which can take players to their higher no blocks.

There is one starting point, one ending point. In this we have to create a window having this board in this format and we have to put one button to select players between 2 to 6 , one start button and we have to create a dice rolling simulator also. After selecting players we can start the game by taping on the start button and then both players start from block having 1 and can move with the help of roll button on the window which give random number between 1 and 6 every time and the first player who reaches the 30^{th} block wins the game.

We have to create all this with the help of object-oriented programming and GUI concepts. We have taken some inspiration from other similar games and created this game so that anyone can easily play this game

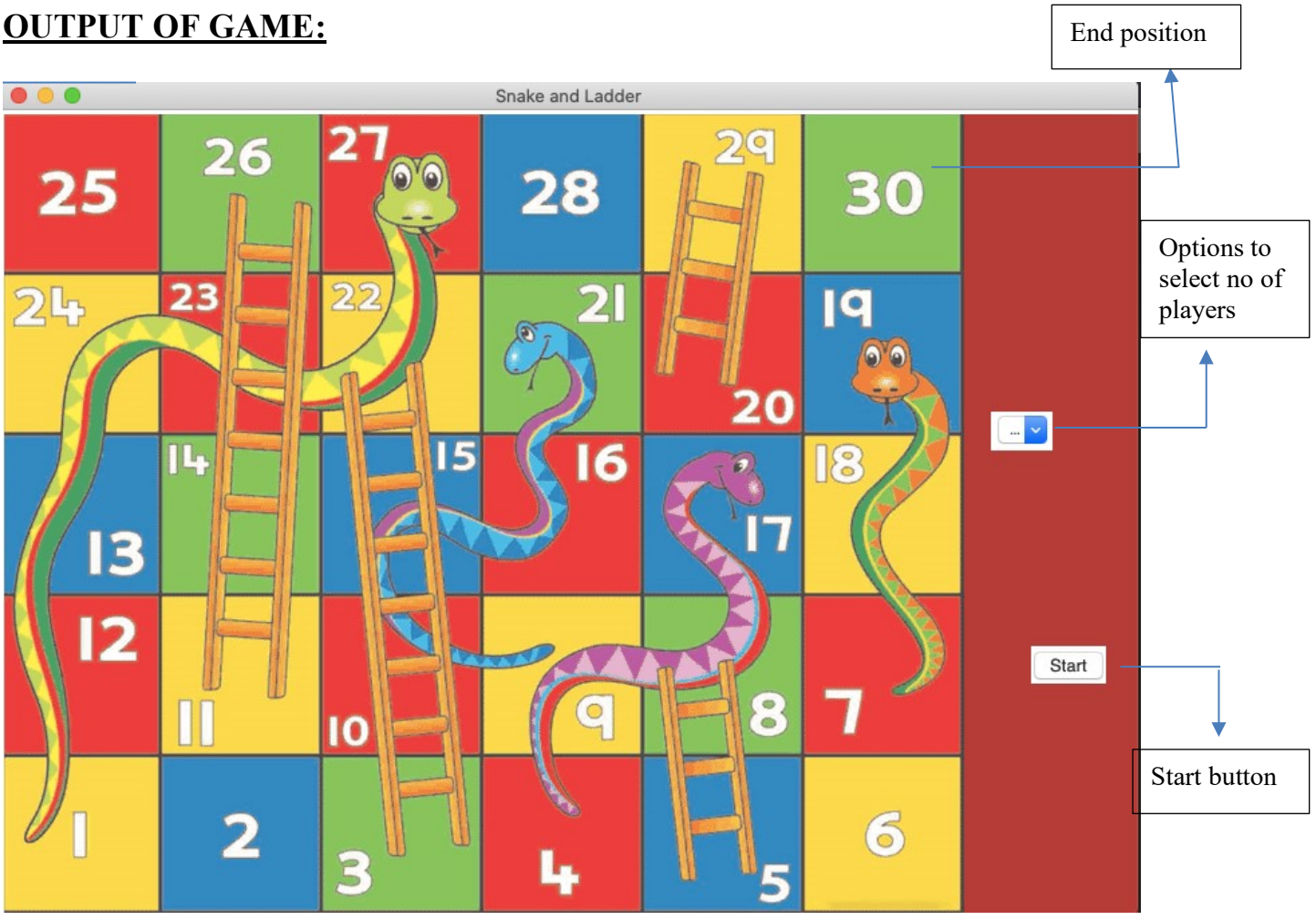
MATHS USED IN THIS GAME:

Any version of Snakes and Ladders can be represented exactly as an absorbing Markov chain, since from any square the odds of moving to any other square are fixed and independent of any previous game history.

The Milton Bradley version of Chutes and Ladders has 100 squares, with 19 chutes and ladders. A player will need an average of 39.2 spins to move from the starting point, which is off the board, to square 100. A two-player game is expected to end in 47.76 moves with a 50.9% chance of winning for the first player. These calculations are based on a variant where throwing a six does not lead to an additional roll; and where the player must roll the exact number to reach square 100 and if they overshoot it their counter does not move.

This is the logic of typical snake and ladders game but we have used different type of logic in our game because of board limitations which includes that starting and ending positions and animation for how player ascends in game and how one can go to end point which is 25th block in our game

OUTPUT OF GAME:

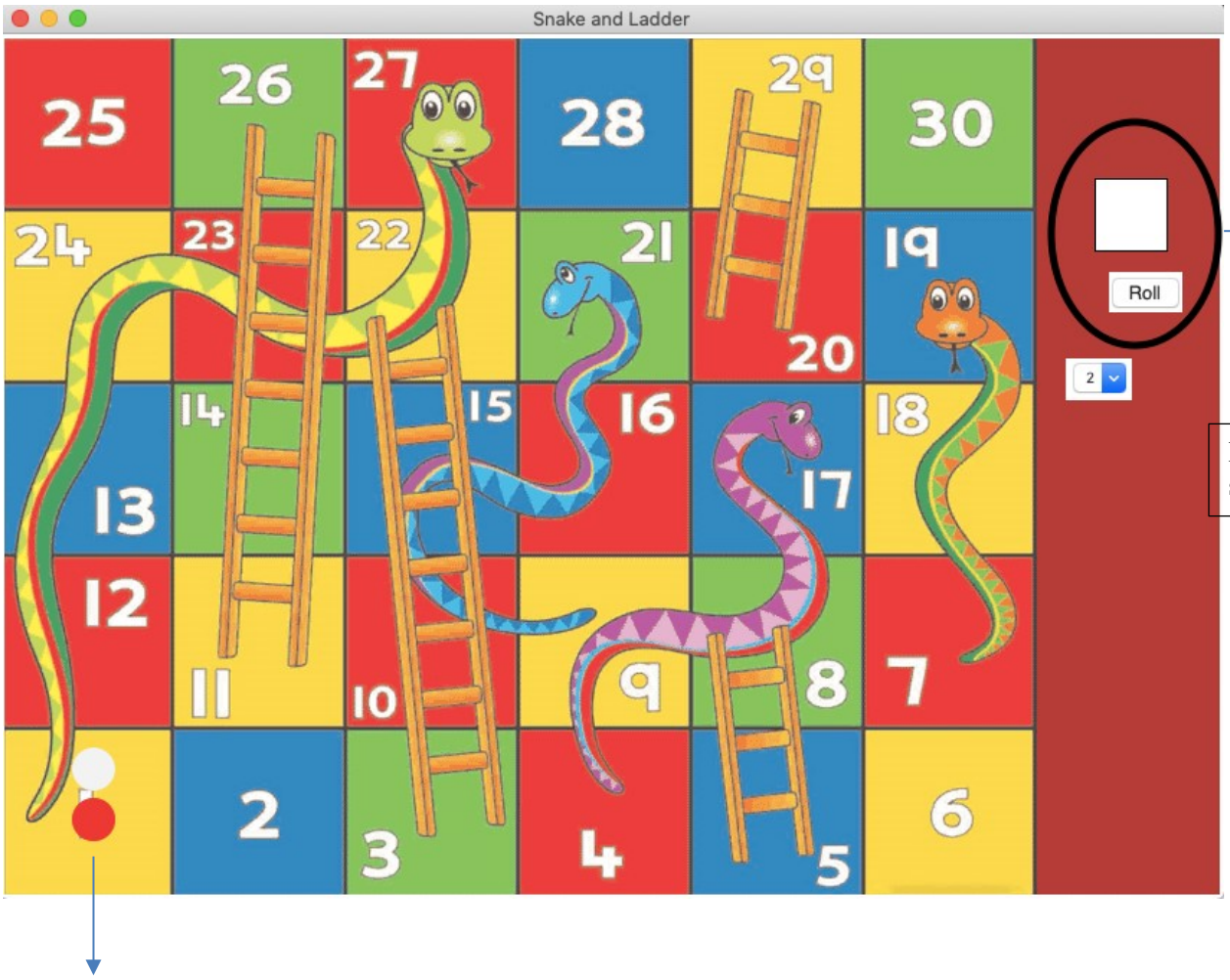


This is the output of our game in the window we are having a board in the format of typical snake and ladder, we are having a drop-down menu for selecting no of players. In this game we can select maximum 6 players and minimum 2 players then we have one start button. After pressing the start button we can see a dice rolling simulator having one empty box in which we are going to get random numbers from 1 t

Each player starts with a token on the starting square (usually the "1" grid square in the bottom left corner, or simply, off the board next to the "1" grid square). Players take turns rolling a single die to move their token by the number of squares indicated by the die roll. Tokens follow a fixed route marked on the gameboard which usually follows a boustrophedon (ox-plow) track from the bottom to the top of the playing area, passing once through every square. If, on completion of a move, a player's token lands on the lower-numbered end of a "ladder", the player moves the token up to the ladder's higher-numbered square. If the player lands on the highernumbered square of a "snake" (or chute), the token must be moved down to the snake's lowernumbered square.

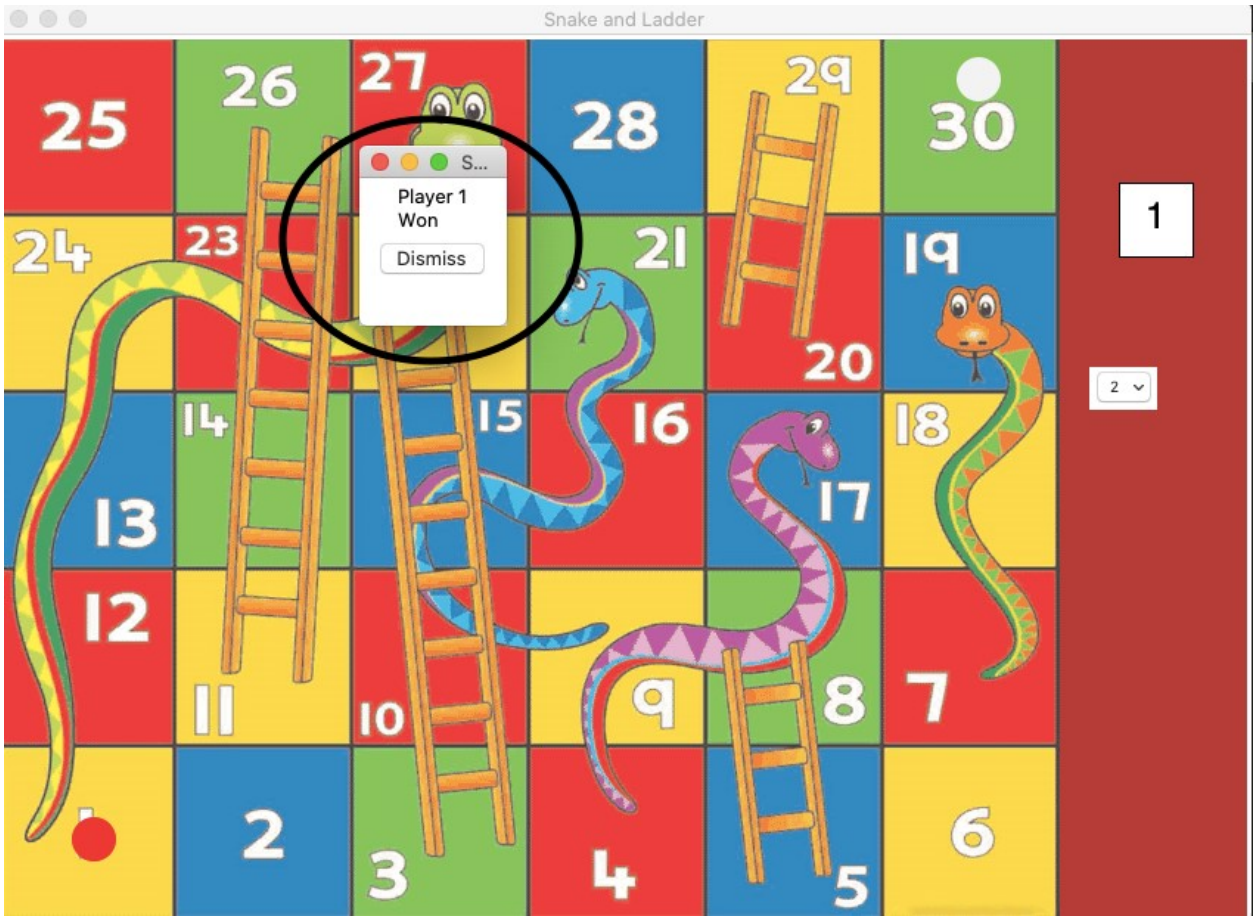
If a 6 is rolled the player, after moving, immediately rolls again for another turn; otherwise play passes to the next player in turn. The player who is first to bring their token to the last square of the track is the winner.

We can see the dice rolling simulator in the given image of output after pressing the start game button this simulator comes and start button does not shows anymore and by pressing the roll button we can get random numbers between 1 to 6 inside white box and we can see 2 players are ready at start position which is 1st block and according to the number we get inside the white box both players can proceed the game and the first players which can make their way to block number 25 after going through all the shortcut and snake bites wins the game



At starting position the no of players we select are standing at when one player presses the roll button then random numbers between 1 to 6 displayed on white box and after that second player chance comes

And the first players which can touch the block 30 wins the game and after winning the game there comes a pop up window in which it shows which player is the winner by mentioning the name of the player



as we can see player 1 which is white player reaches the 30th block first so wins the game and after that pop up window shows up mentioning the name of player

Description of Project

In this we created the game using python tkinter library, dicemove library, time and GUI concepts with some help of object-oriented programing of python. We have used object-oriented functions like

def gamePlay(self) def
create_peice(self) def
peices(self, move, turn) def
diceMove(self, position, turn)
def get_choice(self, value) def
startGame(self)

we have created a class for every aspect of game like

Matchposition() - for match position we have created this class and
Display() - for display we have created this class

we have created a method named find_snake_or_ladder and used self concept. In this method we have assigned the position of player after a shortcut or being cut by a snake by using “elif” then inside class Display we have created the board for snake and ladders using “self.canvas” having width and height 850 and 600 respectively and we have background color brown.

Then we have created a drop-down menu to select players between 2 to 6 named “OPTIONS” for this we have used StringVar and the default value for options is 0

After that we have created start game button with the use of self.startGame placed at x=770, y=400 then we created roll button in rectangle shape having fill = white and outline = black named “ROLL” we have used

self.canvas.create_rectangle
self.diceRoll

and then we have created dice rolling simulator in method def diceMove after starting the game there comes roll button after rolling the button some random numbers between 1 to 6 and time difference between one roll and second roll is 0.25 we have used time.sleep for that and according to that number players ascends same amount of blocks

Then on the logic part we have implemented the logic according to our game which include the starting point at x=120 and y=120 so the starting point should be at same place

Then to reach the end point x has to be 5*x and y has to be 4*y. x should be added to value and y should be subtracted to the value and we have included a variable m which tells which side we have to move i.e. left to right or right to left

And at last we have implemented the code for animation of piece using if-else conditions and we hace used

self.block[turn] self.m[turn]
self.player[turn]
self.canvas.delete(self.player[turn])
self.player.append
self.position.append self.m.append
self.block.append

where m is the variable which denoted where to go left to right or right to left by indicating -1 or 1

we did all this in a single python file and then we have created another python file which include all the information of the file we have created and the image which we wanted to upload as a board of snake and ladders and in this file we have named the game as “Snake and ladder” having geometry 850*600. We have used PhotoImage for displaying the image

WORKFLOW OF PROJECT:

Logic for positions in match having shortcuts or snakes

(class MatchPosition)

Created board for snake and ladder

(class Display(object))

Created drop down menu for selecting players

(OPTIONS)

Created start game button

(self.startGame)

Created dice rolling simulator

Defined the movement of pieces

(def diceMove)

Animation of piece

Importing everything to main python file

```
main.py
1  from userInterface import *
2
3  def main():
4      master = Tk()
5      master.title("Snake and Ladder")
6      master.geometry("850x600")
7      img = PhotoImage( file = "board.gif")
8      x = Display(master,img)
9      master.mainloop()
10
11 main()
12
```

Giving title to game

WORKFLOW OF GAME:

After compiling the main file a window popup comes

In that window we are having an option to select no of players

After selecting no of players we have to press start button

After starting the game we can see a roll button and a white box
(Dice rolling simulator)

We have to press roll button to get numbers on that white box

First player reaching 30th block wins the game

DESCRIPTION OF WORK DIVISION

1. DHEERAJ KUMAR RAI:

I have programmed the position inside match having shortcuts(ladders) or snakes. What happens when someone goes to that position using “elif” inside class MatchPosition from starting which is upto 31st line of code

```
class MatchPosition():
    def find_snake_or_ladder(self, block, turn, position):
        x = 35*(turn>=3)
        y = (turn%3)*35
        if(block == 3):
            return 305+x, 150+y, 22
        elif(block == 5):
            return 545+x, 390+y, 8
        elif(block == 11):
            return 185+x, 30+y, 26
        elif(block == 20):
            return 545+x, 30+y, 29
        elif(block == 17):
            return 425+x, 510+y, 4
        elif(block == 19):
            return 665+x, 390+y, 7
        elif(block == 21):
            return 425+x, 390+y, 9
        elif(block == 27):
            return 65+x, 510+y, 1
        else:
            return position[0], position[1], block
```

Conditions

In this we have commands for what happens at all that place which have either snake or ladder

Then I have created the board for snake and ladder inside class display(object) having width = 850 and height = 600 and background color brown

```
class Display(object):
    def __init__(self,master,img):

        #Create board of snake and ladder
        canvas_width = 850
        canvas_height = 600
        self.color = ["#FFF", "#F00", "#0F0", "#00F", "#FF0", "#0FF"]
        self.canvas = Canvas(master, width = canvas_width, height = canvas_height, bg = "brown")
        self.canvas.grid(padx=0, pady=0)
        self.canvas.create_image(360,300,anchor=CENTER, image = img)
```

Then I have created the drop-down menu for selection of no of players named “options” between 2 to 6 having default value 0 with the help of variable.set and width = 5

```
OPTIONS = ["Players", "2", "3", "4", "5", "6"]
variable = StringVar(master)
variable.set(OPTIONS[0]) # default value
w = OptionMenu(self.canvas, variable, *OPTIONS, command=self.get_choice)
w.pack()
w.place(x=740, y=225)
w.config(font=('calibri',(10)),bg='white',width=5)
```

Then I have created start button denoted by “START” placed at x=770 and y=400 having background white After that I have created the screen for dice and roll button. Screen with outline = black and fill = white And button denoted by “ROLL” having background white using button

```
self.canvas.create_rectangle(810, 150, 760, 100, fill='white', outline='black')
self.canvas.pack(fill=BOTH, expand=1)
self.diceRoll = Button(self.canvas, text="Roll",background='white',
                       command = self.gamePlay, font=("Helvetica"))
self.num_player = int(self.num_player)
self.diceRoll.place(x=770, y=165)
self.create_peice()
self.startGame.place(x=-30, y=-30)
```

After this all the work are done by other members of the team and after the completion of this I have created the main file which imports everything from this file and in that I have named the game as “Snake and ladder” and I imported the image of snake and ladder board using PhotoImage

main.py

```
1  from userInterface import *
2
3  def main():
4      master = Tk()
5      master.title("Snake and Ladder")
6      master.geometry("850x600")
7      img = PhotoImage( file = "board.gif")
8      x = Display(master,img)
9      master.mainloop()
10
11 main()
12
```

2. **MANOHAR KUMAR THAKUR:(THIS PART HAS BEEN WRITTEN INDIVIDUALLY BY TEAM MEMBERS)**

I have created the part after which Dheeraj left. I have created the code for printing the dice value to screen using dice_value then I have coded the part in which movement of pieces are there using elif and then I have coded some part of animation for piece

3. **NEHPAL SINGH:(THIS PART HAS BEEN WRITTEN INDIVIDUALLY BY TEAM MEMBERS)**

I have coded the rest part of animation of piece in that I used

self.player[turn]
if-else
for loop

and having starting value of x and y as 20 and 120 to reach to the last block x should be 5*x and y should be 4*y.
X should be added to value and Y should be subtracted and I have included a variable named m which tells where to go i.e. left to right or right to left
And I have coded the end part of code in which there is instructions for what happens when someone reaches the end point

After any player reaches to end point a pop-up window shows up shwing the name of winner

TECHNOLOGIES AND FRAMEWORK TO BE USED

We have used python programming to built this game with the help of tkinter library and GUI Python is an interpreted, high-level and general-purpose programming language. Created by Guido van Rossum and first released in 1991, Python's design philosophy emphasizes code readability with its notable use of significant whitespace. Its language constructs and objectoriented approach aim to help programmers write clear, logical code for small and large-scale projects.

Python is dynamically typed and garbage collected. It supports multiple programming paradigms, including structured (particularly, procedural), object-oriented, and functional programming. Python is often described as a "batteries included" language due to its comprehensive standard library.

Python was created in the late 1980s as a successor to the ABC language. Python 2.0, released in 2000, introduced features like list comprehensions and a garbage collection system with reference counting.

Python 3.0, released in 2008, was a major revision of the language that is not completely backward compatible, and much Python 2 code does not run unmodified on Python 3.

The Python 2 language was officially discontinued in 2020 (first planned for 2015), and "Python 2.7.18 is the last Python 2.7 release and therefore the last Python 2 release." No more security patches or other improvements will be released for it. With Python 2's end-of-life, only Python 3.6.x and later are supported.

Python interpreters are available for many operating systems. A global community of programmers develops and maintains CPython, a free and open-source reference implementation. A non-profit organization, the Python Software Foundation, manages and directs resources for Python and CPython development.

We have used python tkinter library in this Tkinter is a Python binding to the Tk GUI toolkit. It is the standard Python interface to the Tk GUI toolkit and is Python's de facto standard GUI. Tkinter is included with standard Linux, Microsoft Windows and Mac OS X installs of Python. The name Tkinter comes from Tk interface. Tkinter was written by Fredrik Lundh.Tkinter is free software released under a Python license.

As with most other modern Tk bindings, Tkinter is implemented as a Python wrapper around a complete Tcl interpreter embedded in the Python interpreter. Tkinter calls are translated into Tcl commands which are fed to this embedded interpreter, thus making it possible to mix Python and Tcl in a single application.

SWOT ANALYSIS

Strengths:

1. Technologies innovation
2. Simple to use
3. Light user interface
4. Interactive system with the help of GUI
5. Any age group can use
6. Easy commands

Weakness:

1. Support of all type of image is not there
2. We cant add audio in this with GUI
3. It uses more computer memory
4. It becomes more complicated when needed to communicate with computer
5. Consumes more power when compares to others

Opportunities:

1. Can make more user interactive things
2. User friendly interface

Threats:

1. Hacking can be easy

GITHUB LINK

https://github.com/dkrkv1107/Snake-and-ladder-game

…….END OF REPORT……..