```
def __init__():
    pass

class Graph(object):
    #this list represents valid steps of goat and tiger
    WALKGRAPH = [
         [1, 5, 6],[0, 2, 6],[1, 3, 6, 7, 8],[2, 4, 8],[3, 8, 9],
         [0, 6, 10],[0, 1, 2, 5, 7, 10, 11, 12],[2, 6, 8, 12],[2, 3, 4, 7, 9,
12, 13, 14],[4, 8, 14],
         [5, 11, 15],[6, 10, 12, 16],[6, 7, 8, 11, 13, 16, 17, 18],[8, 12, 14,
18],[8, 9, 13, 18, 19],
         [10, 16, 20],[10, 11, 12, 15, 17, 20, 21, 22],[12, 16, 18, 22],[12,
13, 14, 17, 19, 22, 23, 24],[14, 18, 24],
         [15, 16, 21],[16, 20, 22],[16, 17, 18, 21, 23],[18, 22, 24],[18, 19,
23]
         ]
    #this list represents valid jump step for a tiger
    JUMPGRAPH = [
        [[10,12,2], [5,6,1]],
        [[11,3], [6,2]],
        [[0,10,12,14,4], [1,6,7,8,3]],
        [[1,13], [2,8]],
        [[2,12,14], [3,8,9]],
        [[15,7], [10,6]],
        [[16,18,8], [11,12,7]],
        [[5,17,9], [16,12,8]],
        [[6,16,18], [7,12,13]],
        [[7,19], [8,14]],
        [[0,20,22,12,2], [5,15,6,11,6]],
        [[1,21,13], [6,16,12]],
        [[2,0,10,20,22,24,14,4], [7,6,11,16,17,18,13,8]],
        [[3,11,23], [8,12,18]],
        [[4,2,12,22,24], [9,8,13,18,19]],
        [[5,17], [10,16]],
        [[6,18,8], [11,17,12]],
        [[7,15,19], [12,16,18]],
        [[8,6,16], [13,12,17]],
        [[9,17], [14,18]],
        [[10,22,12], [15,21,16]],
        [[11,23], [16,22]],
        [[12,10,20,24], [17,16,21,23]],
        [[13,21],[18,22]],
        [[14,12,22], [19,18,23]]
    ]
    @staticmethod
    def is_walk_valid(to, frm):
        key = 0
        for value in Graph.WALKGRAPH:
```

```
            if(key == frm):
                for k in value:
                    if k == to:
                        return True
            key = key + 1
        return False
    @staticmethod
    def is_jump_valid(to, frm):
        key = 0
        for value in Graph.JUMPGRAPH:
            #print("key",key)
            if (key == frm):
                k = 0
                for v in value[0]:
                    if (v == to):
                        return True, value[1][k]
                    k = k + 1
            key = key + 1
        return False, None

import turtle
from enum import Enum
from graph import Graph
import tkinter as tk
#from tkinter import *
import pygame
import pygame.mixer
pygame.init()
pygame.mixer.music.load("C:\\Users\\user\\Desktop\\WISE PROJECT(Bagh
Chal)\\02. Akatsukasute Kusu.mp3")
pygame.mixer.music.play(-1)
tigerSound = pygame.mixer.Sound("C:\\Users\\user\\Desktop\\WISE PROJECT(Bagh
Chal)\\roar-47757.mp3")
goatSound= pygame.mixer.Sound("C:\\Users\\user\\Desktop\\WISE PROJECT(Bagh
Chal)\\Goat-Baby-Bah-A-www.fesliyanstudios.com.mp3")

class states(Enum):
    ADD_GOAT = 1
    SELECT_GOAT = 2
    MOVE_GOAT = 3
    SELECT_TIGER = 4
    MOVE_TIGER = 5
PLAYERS = Enum('TYPES', 'EMPTY TIGER GOAT')

class Board(turtle.Turtle): # Class methods and attributes are defined here

    def __init__(self):  # Initialized attributes and created the game board
        self.pos_list = {}
```

```
        self.tigerStamp = "tiger.gif"
        self.goatStamp = "goat.gif"
        self.whiteStamp = "white.gif" #to cover old msg
        self.no_goats = 0
        self.X = -200
        self.Y = 200
        self.current_step = 12 #places cursor in middle of board
        self.died_goat = 0
        self.floating_pos = None
        self.game_state = states.ADD_GOAT
        super().__init__()
        self.myscreen = self.getscreen()
        self.myscreen.tracer(0,0)
        self.myscreen.bgpic("board.png") #400*400 px
        self.shape("turtle")
        self.color("red")
        self.penup()
        next_x = self.X
        next_y = self.Y
        for i in range(5):
            for j in range(5):
                self.setpos(next_x, next_y)
                key = 5 * i + j
                self.pos_list[key] = [next_x, next_y, PLAYERS.EMPTY, 0]
                next_x = next_x + 100
            next_x =self.X
            next_y = next_y - 100
            self.myscreen.register_shape(self.tigerStamp)
            self.myscreen.register_shape(self.goatStamp)
            self.myscreen.register_shape(self.whiteStamp)

 # Set up initial positions for tigers on the board
    def tiger_setup(self):
        keys = self.pos_list.keys()
        #tiger_ids = []
        self.shape(self.tigerStamp)
        for key in keys:
            if (key == 0 or key == 4 or key == 20 or key == 24):
                [x, y, _, _] = self.pos_list[key]
                self.setpos(x,y)
                s_id = self.stamp()
                self.pos_list[key] = [x,y,PLAYERS.TIGER, s_id]
        self.shape("turtle")
        [x, y, _, _] = self.pos_list[self.current_step]
        self.setpos(x,y)

#events Methods
    def move_up(self):
```

```
        if(self.current_step > 4):    # Move the cursor/turtle up
            self.current_step = self.current_step - 5
            [x, y, _, _] = self.pos_list[self.current_step]
            self.setpos(x,y)

    def move_down(self):  # Move the cursor/turtle down
        if(self.current_step < 20):
            self.current_step = self.current_step + 5
            [x, y, _, _] = self.pos_list[self.current_step]
            self.setpos(x,y)


    def move_left(self):  # Move the cursor/turtle left
        if((self.current_step > 0)):
            self.current_step = self.current_step - 1
            [x, y,_,_] = self.pos_list[self.current_step]
            self.setpos(x,y)


    def move_right(self):  # Move the cursor/turtle right
        if( (self.current_step < 24)):
            self.current_step = self.current_step + 1
            [x, y,_,_] = self.pos_list[self.current_step]
            self.setpos(x,y)


    def pressed_enter(self):
        if(self.died_goat>4):
            self.show_msg("Tiger won! Goat lost :( GAME OVER ")
            return
        if(self.game_state==states.ADD_GOAT):
            [x,y,t,_] = self.pos_list[self.current_step]
            if(t == PLAYERS.EMPTY):
                self.shape(self.goatStamp)
                s_id = self.stamp()
                self.pos_list[self.current_step] = [x,y,PLAYERS.GOAT, s_id]
                goatSound.play()
                self.shape("turtle")
                self.game_state = states.SELECT_TIGER
                if self.no_goats >= 20:
                    self.game_state = states.SELECT_GOAT
                if self.no_goats - self.died_goat > 5:
                    return


        elif(self.game_state == states.SELECT_GOAT):
            [x,y,t,i] = self.pos_list[self.current_step]
            if(t == PLAYERS.GOAT):
                self.clearstamp(i)
```

```
                [x,y,_,_] = self.pos_list[self.current_step]
                self.pos_list[self.current_step] = [x,y,PLAYERS.EMPTY,0]
                self.shape(self.goatStamp)
                self.floating_pos = self.current_step
                self.game_state = states.MOVE_GOAT

        elif(self.game_state == states.MOVE_GOAT):
            [x,y,t,_] = self.pos_list[self.current_step]
            if(t == PLAYERS.EMPTY and Graph.is_walk_valid(self.current_step,
self.floating_pos)):
                s_id = self.stamp()
                self.pos_list[self.current_step] = [x,y,PLAYERS.GOAT, s_id]
                self.shape("turtle")
                self.game_state = states.SELECT_TIGER

        elif(self.game_state == states.SELECT_TIGER):
            [x,y,t,i] = self.pos_list[self.current_step]
            if(t == PLAYERS.TIGER):
                self.clearstamp(i)
                [x,y,_,_] = self.pos_list[self.current_step]
                self.pos_list[self.current_step] = [x,y,PLAYERS.EMPTY,0]
                self.shape(self.tigerStamp)
                self.floating_pos = self.current_step
                self.game_state = states.MOVE_TIGER

        elif(self.game_state == states.MOVE_TIGER):
            [x,y,t,_] = self.pos_list[self.current_step]
            if (t != PLAYERS.EMPTY):
                return
            if(Graph.is_walk_valid(self.current_step, self.floating_pos) ):
                s_id = self.stamp()
                print("tiger@walk")
                self.pos_list[self.current_step] = [x,y,PLAYERS.TIGER, s_id]
                self.shape("turtle")
            else:
                jump, g_pos = Graph.is_jump_valid(self.current_step,
self.floating_pos )
                if(not jump and g_pos is  None):
                    print(g_pos)
                    return
                [xg,yg,gt,i] = self.pos_list[g_pos]
                if(gt != PLAYERS.GOAT):
                    print("not goat")
                    return
                self.clearstamp(i)  #killing goat
                tigerSound.play()
                self.pos_list[g_pos] = [xg,yg,PLAYERS.EMPTY,i]
                self.died_goat = self.died_goat + 1
```

```
                s_id = self.stamp()
                self.pos_list[self.current_step] = [x,y,PLAYERS.TIGER, s_id]
                self.shape("turtle")
            print(self.no_goats)
            if(self.no_goats >= 20):
                self.game_state = states.SELECT_GOAT
            else:
                self.game_state = states.ADD_GOAT

        self.show_msg(self.game_state)

    def setup_key_controls(self):
                self.myscreen.onkey(self.move_up, "Up")
                self.myscreen.onkey(self.move_down, "Down")
                self.myscreen.onkey(self.move_left, "Left")
                self.myscreen.onkey(self.move_right, "Right")

                # Player 2 controls with 'w', 's', 'a', 'd' keys
                self.myscreen.onkey(self.move_up, "w")
                self.myscreen.onkey(self.move_down, "s")
                self.myscreen.onkey(self.move_left, "a")
                self.myscreen.onkey(self.move_right, "d")

        # Player actions are activated by 'Enter' key for both players
                self.myscreen.onkey(self.pressed_enter, "Return")
                self.myscreen.listen()


    def show_msg(self, msg,font_size=16):   # Display a message on the screen
        temp_pos = self.pos()
        temp_stamp = self.shape()
        self.hideturtle()
        self.setpos(0, 270)
        self.shape(self.whiteStamp)
        self.stamp()
        self.shape(temp_stamp)
        self.write(msg, True, align="center",font=("Arial",font_size,"bold"))
        self.setpos(temp_pos)
        self.showturtle()



def start_button_clicked():
    def main():     # Main function to run the game
        board = Board()
        board.tiger_setup()
        board.setup_key_controls()
        board.myscreen.tracer(1, 1)
        board.show_msg("Game Started, Place your goat")
```

```
        board.myscreen.title("Bagh Chal")
        board.myscreen.mainloop()
    if __name__ == "__main__":
        main()
    start_button.config(text="Started", state=tk.DISABLED)

# Create the main window
root = tk.Tk()
root.title("Start Button GUI")

# Create a label
label = tk.Label(root, text="Welcome to BAGH CHAL! Click on start to play game
:)")
label.pack(pady=10)

# Create a start button
start_button = tk.Button(root, text="Start", command=start_button_clicked)
start_button.pack()

# Run the GUI event loop
root.mainloop()
```