Task 12 :- Simulate Gaming concepts using Pygame

Aim:-
To Simulate Gaming concepts using pygame

Snake Game:-

problem 1:- write a python program to create a snake Game using pygame package.

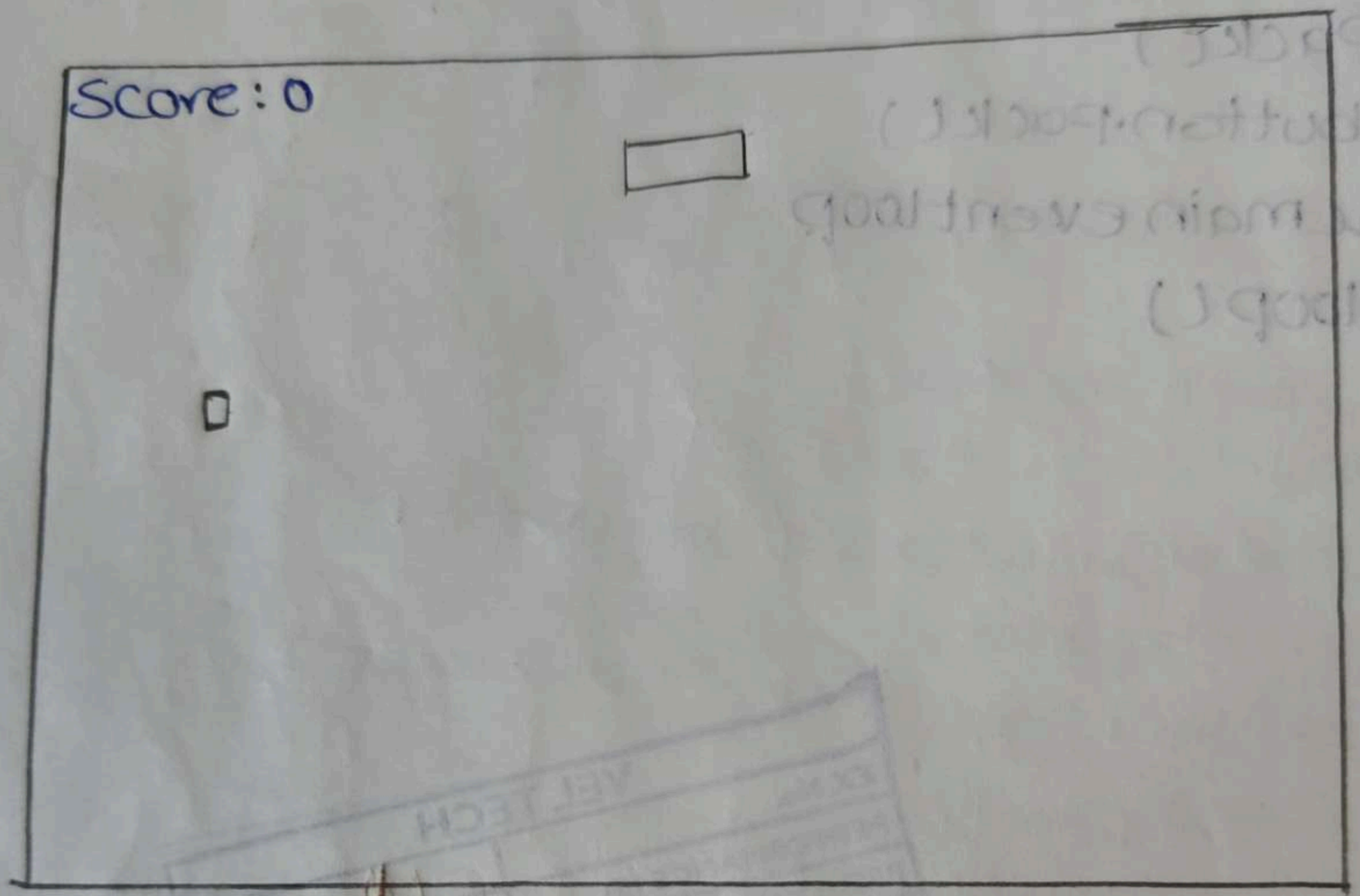
Algorithm:-

* Import pygame package and initialize it
* Define the window size and title
* Create a snake class which initializes the snake position, colour, and movement.
* create a fruit class which initializes the fruit position and colour
* Create a function to check if the snake collides with the fruit and increase the score.
* Create a function to Check if the snake collides with the window and end the game.
* create a function to update the snake position based on the user input
* Create a function to update the game display and draw the snake and fruit
* Create a game loop to continuously update the game display, snake position and check for collisions.
* end the game if the user quits or the snake collides with the window.

Program:-

```
#importing libraries
import pyname
import time
```

output:-



```
import random
snake_speed = 15
# window size
window_x = 720
window_y = 480
# defining colors
black = pygame.color(0,0,0)
white = pygame.color(255, 255,255)
red  = pygame.color(255,0,0)
green = pygame.color(0,255,0)
blue = pygame.color(0,0,255)
#Initialising pygame
pygame.init()
#initialise gamewindow
Pygame.display.set_caption('GeeKS for Geeks snakes')
game_window = pyname.display.set_mode((window_x, wind
                                       ow_y))
# FPS (frames per second) controller
fps = pygame.time.clock()
# defining snake default position
snake_position = [100, 50]
#defining first 4 blocks of snake body
snake_body = [[100,50], [90,50], [80,50], [70,50]]
#fruit position
fruit_position = [random.randrange(1, (window_x/10))*
                  random.randrange(1, (window_y/10))*10]

fruit_spawn = True
#setting default snake direction towards
#right
direction = 'RIGHT'
change_to = direction
```

```
# initial score
score =0
# displaying score function
def show_score(choice, color, font, size):
    # creating font object score_font
    score_font = pygame.font.sysfont(font, size)
    # create the display surface object
    #score_[illegible] Surface
    score_surface = score_font.render('score:' + str(score),
                                      True, color)
    # create a rectangular object for the next
    # surface object
    score_rect = score_surface.get_rect()
    #displaying text
    game_window.blit(score_surface, score_rect)
#game over function
def game_over():
    #creating font object my_font
    myfont = pygame.font.sysfont('times new roman', 50)
    #Creating a text surface on which text
    #will be drawn
    game_over_surface = my_font.render(
        'your score is:' + str(score), True, red)
    # create a rectangular object for the text
    # surface object
    game_over_rect = game_over_surface.get_rect()
    #setting position of the text
    game_over_rect.midtop = (window_x/2, window
                             -y/4)

    # blit will draw the text on screen
    game_window.blit(game_over_surface, game_
                                        over_rect)
    pygame.display.flip()
```

```
#after 2 seconds we will quit the program
time.sleep(2)
#deactivating pygame library
pygame.quit()
#quit the program
quit()
#main function
while True:
  #handling key events
  for event in pygame.event.get():
    if event.type == pygame.KEYDOWN:
      if event.key == pygame.K_UP:
        change_to = 'DOWN'
      if event.key == pygame.K_LEFT:
        change_to = 'LEFT'
      if event.key == pygame.K_RIGHT:
        change_to = 'RIGHT'
  #If two keys pressed simultaneously
  #we don't want snake to move into two
  #directions simultaneously
  if change_to == 'up' and direction != 'DOWN':
    direction = 'UP'
  if change_to == 'DOWN' and direction != 'up':
    direction = 'DOWN'
  if change_to == 'LEFT' and direction != 'RIGHT':
    direction = 'LEFT'
  if change_to == 'RIGHT and direction != 'LEFT':
    direction = 'RIGHT'
  #moving the snake
  if direction == 'UP':
    snake_position[1] -= 10
  if direction == 'DOWN':
```

```
    snake_position[1]+=10
if direction=='LEFT':
    snake_position[0]-=10
if direction=='RIGHT':
    snake_position[0]+=10
#snake body growing mechanism
#if fruits and snakes collide-then scores
# will be incremented by 10
snake_body.insert(0,list(snake_position))
if snake_position[0]==fruit_position[0]and snake_
Position[1]==Fruit_Position[1]:
    score+=10
    fruit_spawn=false
else:
    snake_body.pop()
if not fruit_spawn:
    fruit_position=[random.randrange(1,(window_x
                    //10))*10,random.randrange(1,
                    (window_y//10))*10]
Fruit_spawn=True
game_window.fill(black)
for pos in snake_body:
    pygame.draw.rect(game_window,green,
                     pygame.Rect(pos[0],pos[1],10,10))

pygame.draw.rect(game_window,white,pygame.Rect(
    #fruit_position[0],fruit_position[1],10,10))

#game over conditions
if snake_position[0]<0 or snake_position[0]>
                    window_x-10;
    game_over()
if snake_position[1]<0 or snake_position[1]>
                    window_y-10;
    game_over()
```

```
#Touching the snake body
for block in snake_body[1:]:
    if snake_position[0] == block[0] and snake_
                                Position[1] == block[1]:
        game_over()

#displaying score continuously
show■ = score(1, white, 'times new roman', 20)
#Refresh game screen
pygame.display.update()
# frame per second/RefreshRate
fps.tick(snake_speed)
```

Problem 2:- Write a python program to Develop a chess board using pygame.

Algorithm:-

Import pygame and initialize it
set screen size and title
Define colors for the board and pieces
Define a function to draw the board by looping over rows and columns and drawing squares of different colors.
Define a function to draw the pieces on the board by loading images for each piece and placing them on the corresponding square.
Define the initial state of the board as a list of lists containing the pieces.
Draw the board and pieces on the screen.
start the game loop.

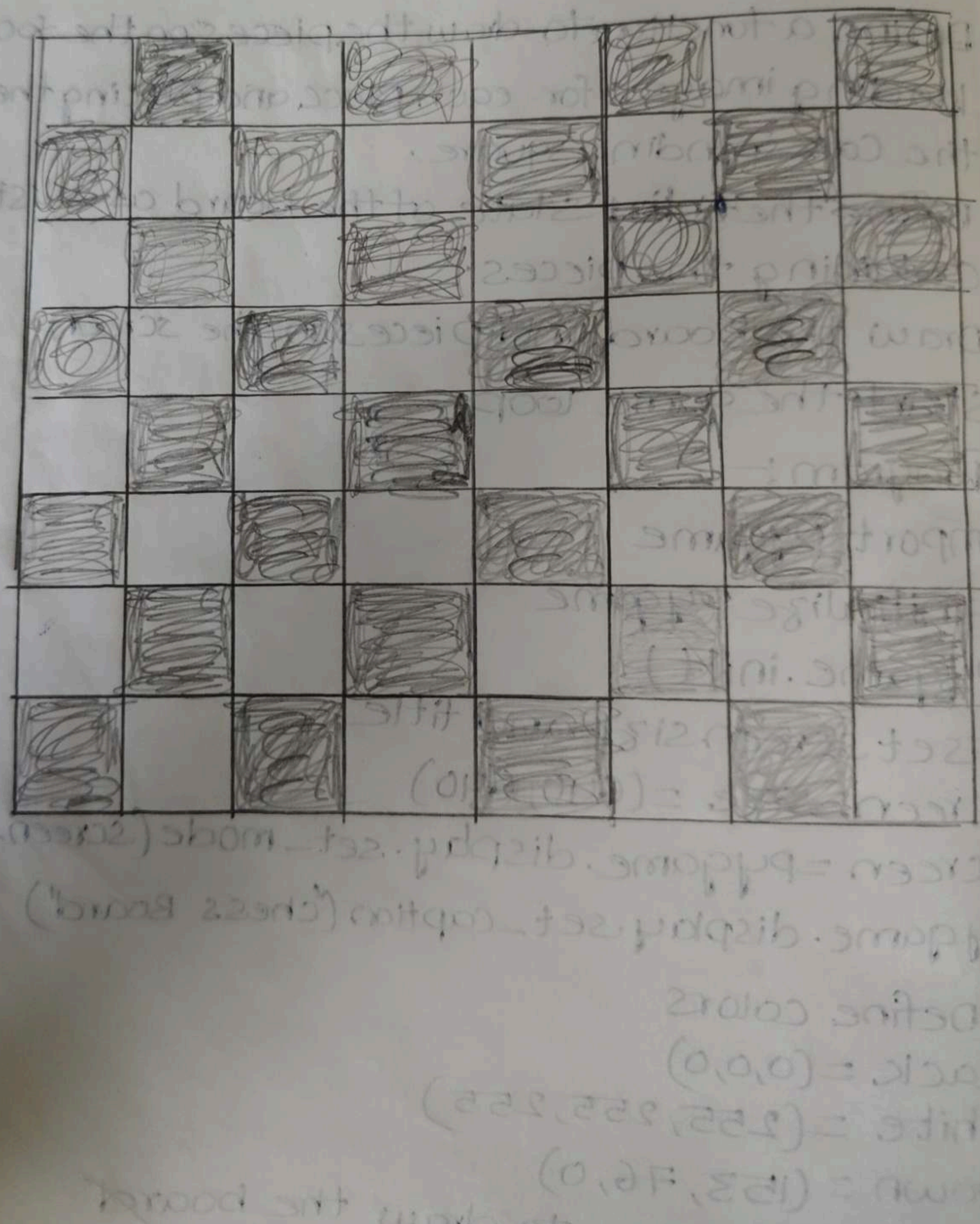
Program:-

```
import pygame
#initialize pygame
pygame.init()
#set screen size and title
screen_size =(640, 640)
screen =pygame.display.set_mode(screen_size)
Pygame.display.set_captian('Chess Board')

#Define colors
black =(0,0,0)
white =(255, 255,255)
brown =(153, 76, 0)
#Define function to draw the board
deF draw_board():
    for row in range (8):
        for col in range(8):
```

```
        square_color = white if (row+col) % 2 == 0 else brown
        square_rect = pygame.Rect(col*80, row*80, 80, 80)
        Pygame.draw.rect(screen, square_color, square
                                                _rect)
# Define function to draw the pieces
def draw_Pieces(board):
    Price_images = {
        'r': pygame.image.load('images|rook.png'),
        'n': pygame.image.load('images|kinght.png'),
        'b': pygame.image.load('images|bishop.png'),
        'q': pygame.image.load('images|queen.png'),
        'k': pygame.image.load('images|king.png'),
        'p': pygame.image.load('images|Pawn.png').
    }
    for row in range(8):
     For col in range(8):
        Piece = board[row][col]
        if Piece != '.':
            Piece_image = piece_images[Piece]
            Piece_rect = Pygame.Rect(col*80, row*80,
                                     80,80)
            screen.blit(piece_image, piece_rect)
#define initial state of the board.
board = [
    ['r', 'n', 'b', 'q', 'k', 'b', 'n', 'r'],
    ['p', 'p', 'p', 'p', 'p', 'p', 'p', 'p'],
    ['.', '.', '.', '.', '.', '.', '.', '.'],
    ['.', '.', '.', '.', '.', '.', '.', '.'],
    ['.', '.', '.', '.', '.', '.', '.', '.'],
    ['.', '.', '.', '.', '.', '.', '.', '.'],
```

output:



```
    ['P', 'P', 'P', 'P', 'P', 'P', 'P', 'P'],
    ['R','N', 'B', 'Q', 'K', 'B', 'N','R']
]
# Draw board and pieces
draw_board()
draw_pieces(board)
# start game loop
while True:
    for event in pygame.event.get():
        if event.type == pygame.QUIT:
            pygame.quit()
            quit()
    pygame.display.update()
```

| VEL TECH | |
|---|---|
| EX NO. | 12 |
| PERFORMANCE (5) | 5 |
| RESULT AND ANALYSIS (5) | 5 |
| VIVA VOCE (5) | 5 |
| RECORD (5) | 5 |
| TOTAL (10) | 25 |
| SIGN WITH DATE | 15/10 |

Result: Thus, the program for pygame is executed and verified successfully.