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

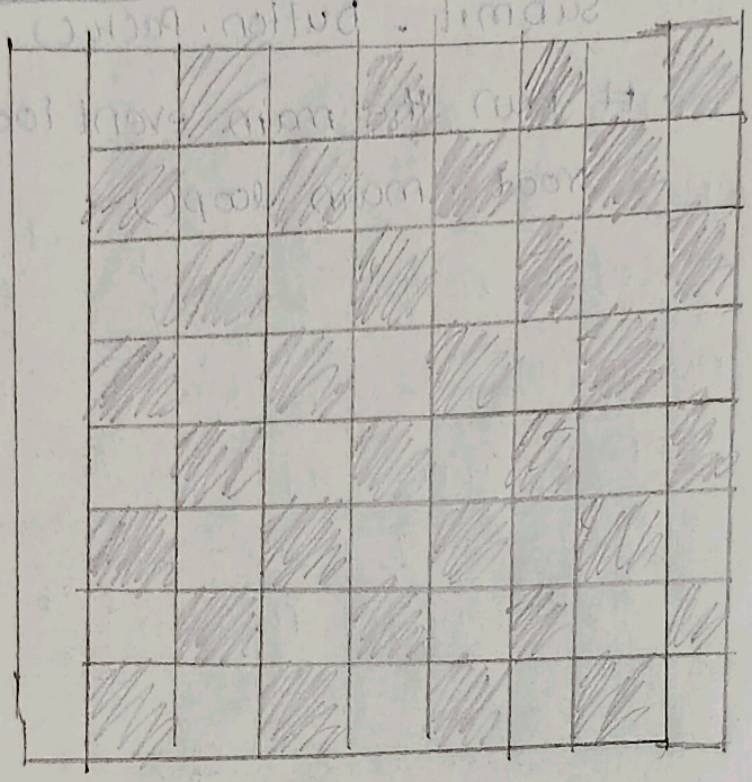
conditions:-

1) set the window size
2) create a snake
3) make the sanke to move in the directions when left, right, down and up key is pressed
4) when the snake hits the fruit increase the score by 10
5) if the snake hits the window Game over

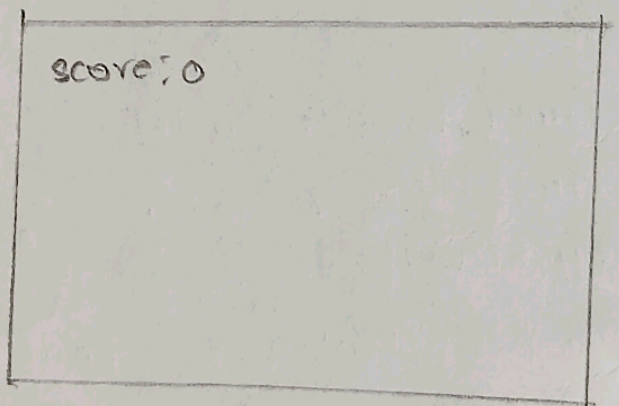
Algorithm:-

1) import pygame package and initialize it.
2) Define the window size and title
3) create a fruit class which initalize the fruit position and colour
4) Create a snake class which initialize the snake position, colour and moment.
5) Create a function to check if the snakee colides with the fruit and increse the score
6) Create a function to check if the snake colides with the window and end the game.
7) Create a function to update the snake position based on the usey input

Sample output



output



8) create a function to update the game display and draw the snake and fruit.

9) create a game loop to continuously update the game display snake position and check for collisions

10) End the game if the user exits or the snake collides with the window

Program:-

```
# importing libraries
import pygame
import time
import random
snake_speed = 15
# window size
window_x = 720
window_y = 480
# defining colours
black = pygame.colour(0,0,0)
white = pygame.colour(255, 255, 255)
red = Pygame.colour(22500)
green = pygame.colour(0,255,0)
blue = Pygame.color(0,0,255)
# initialising pygame
pg.pygame.init()
# initialise game window
Pygame.display.set_caption('Geeks for Geeks snakes')
```

```
window = pygame.display.set_mode
((window_x, window_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))*10,
    random.randrange(1,(window_y//10))*10]
fruit_spawn = True
# setting default snake direction towards
# right
direction='RIGHT'
    change_to = direction
# initial store
    score=0
# displaying store 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 - surface
  Score - surface = score - font .render(score,t
str (score), True colour)
# create a rectangular object for the text
# 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
  my - font = pygame. font sysFont('times
  new roman', 50)
# creating font 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 exit the
  program time.sleep(2)
# deactivating pygame library
  pygame.quit()

# main
# quit the program
  quit()
# main function
  while True:
# healding key events
    for event in pygame.event.get():
      if event.type == pygame.KEY DOWN:
        if event.key == pygame.K-UP:
          change-to = 'up'
        if event.key == pygame.K- DOWN:
          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
    # 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'
```

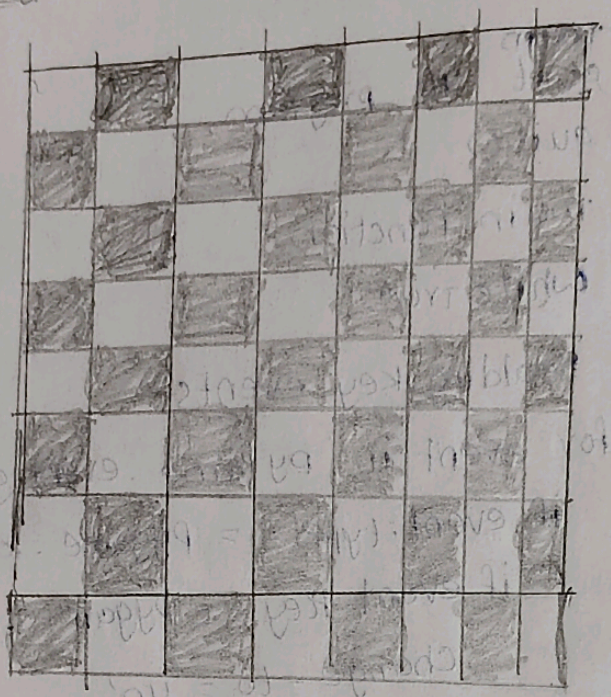
```
if change_to == 'RIGHT' and direction != 'LEFT':
    direction = 'RIGHT'

# 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 fruit 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+
        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:
        pg.game.draw.rect(game_window,green
```

output:



```
        pygame.Rect(pos[0], pos[1], 10, 10))
    pygame.draw.rect(game_window, white,
                                    Pygram.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 snak_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]: gameov
# displaying score continuously
    show_score(1, white, 'times new roman'
# refresh game screen
    pygame.display.update()
# Frome per second /Refresh Rate
    fps.tick(snake_speed)
```

Problem 2:- write a python program to develop a chess board using pygame samele

Algorithm:-

1) import pygame and initialize it
2) set screen size and title
3) Define colors for the board and pieces
4) Define a function to draw the board by looping over raws and columns and drawing squares of different colors
5) Define the initial state of the board as a list containing the pieces
6) Draw the board and pieces on the screen
7) start the game loop

Program:-

```
import pygame
# initialize pygame
   Pygame . init()
# set screen size and little
   screen - size = (640, 640)
   screen = pygame. display . set - mode (screen-size)
   Pygame - display. set - caption ('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):
```

```
            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):
    piece_images = {
        'r': pygame.image.load("images/rook.png"),
        'n': pygame.image.load("images/knight.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 rang(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'],
    ['', '', '', '', '', '', '', ''],
    ['', '', '', '', '', '', '', ''],
```

```
    ['.', '.', '.', '.', '.', '.', '.', '.']
    ['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()
```

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

VELTECH	
Ex.No.	
PERFORMANCE (5)	
RESULT AND ANALYSIS (5)	
VIVA VOCE (5)	
RECORD (5)	
TOTAL (20)	
SIGN WITH DATE	