

ASSIGNMENT OF PYTHON BCA 4th END SEMESTER EXAM

NAME - GITARTHA TALUKDAR

STUDENT ID - CS23BCAGN007

1. WAP Using Python to implement Arithmetic & Quadratic Operations

CODE -

```
1  import math
2  a = 30
3  b = 5
4  print("Addition:", a + b)
5  print("Subtraction:", a - b)
6  print("Multiplication:", a * b)
7  print("Division:", a / b)
8  a = 1
9  b = -3
10 c = 2
11 discriminant = b**2 - 4*a*c
12 root1 = (-b + math.sqrt(discriminant)) / (2*a)
13 root2 = (-b - math.sqrt(discriminant)) / (2*a)
14 print("Roots:", root1, root2)
15 a = 10
16 b = 6
```

2. WAP Using Python to implement linear equation

CODE -

```
1  def solve_linear_equation(a, b):
2      # ax + b = 0
3      if a == 0:
4          if b == 0:
5              return "Infinite solutions (any value of x satisfies the equation)."
6          else:
7              return "No solution (inconsistent equation)."
8      else:
9          x = -b / a
10         return f"The solution is x = {x}"
```

3. WAP Using mathematical function or equation to give graphical representation like star & graph.

CODE-

```
1 import matplotlib.pyplot as plt
2 import numpy as np
3 x = np.linspace(-10, 10, 400)
4 y1 = x**2 - 2*x + 1
5 y2 = np.sin(x)
6 plt.figure(figsize=(10, 5))
7 plt.plot(x, y1, label='y = x2 - 2x + 1', color='blue')
8 plt.plot(x, y2, label='y = sin(x)', color='red')
9 plt.title("Graph of Quadratic and Sine Functions")
10 plt.xlabel("x-axis")
11 plt.ylabel("y-axis")
12 plt.legend()
13 plt.grid(True)
14 plt.tight_layout()
15 plt.show()
```

4. WAP to implement Functions

Defines a Python function to check for prime numbers.

CODE-

```
1 def is_prime(n):
2     if n <= 1:
3         return False
4     for i in range(2, int(n**0.5)+1):
5         if n % i == 0:
6             return False
7     return True
8
9 print(is_prime(7))
```

5. WAP using Tkinter make any formatted application according to your ideas.

CODE-

```
Tkinter

import tkinter as tk
import random

WIDTH = 600
HEIGHT = 400
CELL_SIZE = 20
UPDATE_DELAY = 100

direction = 'Right'
snake = [(100, 100), (80, 100), (60, 100)]
food = None

window = tk.Tk()
window.title("Snake Game - Enhanced Version")
canvas = tk.Canvas(window, width=WIDTH, height=HEIGHT, bg="black")
canvas.pack()

def draw_snake():
    canvas.delete("snake")
    for segment in snake:
        x, y = segment
        canvas.create_rectangle(x, y, x + CELL_SIZE, y + CELL_SIZE, fill="lime",
                                tag="snake")

def place_food():
    global food
    x = random.randint(0, (WIDTH - CELL_SIZE) // CELL_SIZE) * CELL_SIZE
    y = random.randint(0, (HEIGHT - CELL_SIZE) // CELL_SIZE) * CELL_SIZE
    food = (x, y)
    canvas.create_oval(x, y, x + CELL_SIZE, y + CELL_SIZE, fill="red", tag="food")

def change_direction(event):
    global direction
    opposites = {'Up': 'Down', 'Down': 'Up', 'Left': 'Right', 'Right': 'Left'}
    if event.keysym in opposites and direction != opposites[event.keysym]:
        direction = event.keysym

def move_snake():
    global snake, food

    head_x, head_y = snake[0]
    if direction == 'Up': head_y -= CELL_SIZE
    elif direction == 'Down': head_y += CELL_SIZE
    elif direction == 'Left': head_x -= CELL_SIZE
    elif direction == 'Right': head_x += CELL_SIZE

    new_head = (head_x, head_y)

    if new_head in snake or head_x < 0 or head_y < 0 or head_x >= WIDTH or head_y >=
HEIGHT:
        canvas.create_text(WIDTH // 2, HEIGHT // 2, text="GAME OVER", fill="white",
                             font=('Arial', 32))
        return

    snake.insert(0, new_head)

    if new_head == food:
        place_food()
    else:
        snake.pop()

    draw_snake()
    canvas.after(UPDATE_DELAY, move_snake)

draw_snake()
place_food()
window.bind("<KeyPress>", change_direction)
move_snake()
window.mainloop()
```