INT213-PROJECT

Member-1

Name-farhan ahmad

Roll no-38

Reg no-11916199

Section-K19QK B38

Member-2

Name-Amaralinga reddy

Roll no-32

Reg no-11903205

Section-K19QK A32

SUDOKU GAME

Sudoku is a logic-based, combinatorial number-placement puzzle. The objective is to fill a 9×9 grid with digits so that each column, each row, and each of the nine 3×3 subgrids that compose the grid contain all of the digits from 1 to 9.
We will be buiding the Sudoku Game in python using pygame library and automate the game using backtracking algorithm.

Features Implemented

- **Game Interface to Play**
- **Auto solving**
- **Visualization of auto solving i.e. Backtracking Algorithm visualization**
- **Options: Reset, Clear game**

Prerequisite :

- **Paygame library must be preinstalled**
- **Knowledge on Backtracking Algorithm**

Implementation Steps :

1. **Fill the pygame window with Sudoku Board i.e., Construct a 9×9 grid.**
   2. Fill the board with default numbers.
   3. Assign a specific key for each operations and listen it.
   4. Integrate the backtracking algorithm into it.
   5. Use set of colors to visualize auto solving.

Instruction:

- **Press 'Enter' To Auto Solve and Visualize.**

- **To play the game manually,**
  Place the cursor in any cell you want and enter the number.

- **At any point, press enter to solve automatically.**

Below is the Implementation code:

```
# import pygame library

Import pygame


# initialise the pygame font

pygame.font.init()


# Total window

screen = pygame.display.set_mode((500, 600))


# Title and Icon

pygame.display.set_caption("SUDOKU SOLVER USING
BACKTRACKING")

img = pygame.image.load('icon.png')

pygame.display.set_icon(img)


x = 0

y = 0

dif = 500 / 9

val = 0

# Default Sudoku Board.

grid =[
```

```
            [7, 8, 0, 4, 0, 0, 1, 2, 0],
            [6, 0, 0, 0, 7, 5, 0, 0, 9],
            [0, 0, 0, 6, 0, 1, 0, 7, 8],
            [0, 0, 7, 0, 4, 0, 2, 6, 0],
            [0, 0, 1, 0, 5, 0, 9, 3, 0],
            [9, 0, 4, 0, 6, 0, 0, 0, 5],
            [0, 7, 0, 3, 0, 0, 0, 1, 2],
            [1, 2, 0, 0, 0, 7, 4, 0, 0],
            [0, 4, 9, 2, 0, 6, 0, 0, 7]
        ]

# Load test fonts for future use
font1 = pygame.font.SysFont("comicsans", 40)
font2 = pygame.font.SysFont("comicsans", 20)
def get_cord(pos):
        global x
        x = pos[0]//dif
        global y
        y = pos[1]//dif

# Highlight the cell selected
def draw_box():
        for i in range(2):
```

```
            pygame.draw.line(screen, (255, 0, 0), (x * dif-3, (y + i)*dif), (x
* dif + dif + 3, (y + i)*dif), 7)

            pygame.draw.line(screen, (255, 0, 0), ( (x + i)* dif, y * dif ), ((x
+ i) * dif, y * dif + dif), 7)


# Function to draw required lines for making Sudoku grid

def draw():

    # Draw the lines


    for i in range (9):

        for j in range (9):

            if grid[i][j]!= 0:


                # Fill blue color in already numbered grid

                pygame.draw.rect(screen, (0, 153, 153), (i * dif, j *
dif, dif + 1, dif + 1))


                # Fill gird with default numbers specified

                text1 = font1.render(str(grid[i][j]), 1, (0, 0, 0))

                screen.blit(text1, (i * dif + 15, j * dif + 15))

    # Draw lines horizontally and verticallyto form grid

    for i in range(10):

        if i % 3 == 0 :

            thick = 7
```

```
            else:
                thick = 1
            pygame.draw.line(screen, (0, 0, 0), (0, i * dif), (500, i * dif),
thick)
            pygame.draw.line(screen, (0, 0, 0), (i * dif, 0), (i * dif, 500),
thick)

# Fill value entered in cell
def draw_val(val):
    text1 = font1.render(str(val), 1, (0, 0, 0))
    screen.blit(text1, (x * dif + 15, y * dif + 15))

# Raise error when wrong value entered
def raise_error1():
    text1 = font1.render("WRONG !!!", 1, (0, 0, 0))
    screen.blit(text1, (20, 570))
def raise_error2():
    text1 = font1.render("Wrong !!! Not a valid Key", 1, (0, 0, 0))
    screen.blit(text1, (20, 570))

# Check if the value entered in board is valid
def valid(m, i, j, val):
    for it in range(9):
```

```
            if m[i][it]== val:
                return False
            if m[it][j]== val:
                return False
        it = i//3
        jt = j//3
        for i in range(it * 3, it * 3 + 3):
            for j in range (jt * 3, jt * 3 + 3):
                if m[i][j]== val:
                    return False
        return True

# Solves the sudoku board using Backtracking Algorithm
def solve(grid, i, j):

        while grid[i][j]!= 0:
            if i<8:
                i+= 1
            elif i == 8 and j<8:
                i = 0
                j+= 1
            elif i == 8 and j == 8:
                return True
```

```
pygame.event.pump()
for it in range(1, 10):
        if valid(grid, i, j, it)== True:
                grid[i][j]= it
                global x, y
                x = i
                y = j
                # white color background\
                screen.fill((255, 255, 255))
                draw()
                draw_box()
                pygame.display.update()
                pygame.time.delay(20)
                if solve(grid, i, j)== 1:
                        return True
                else:
                        grid[i][j]= 0
                # white color background\
                screen.fill((255, 255, 255))

                draw()
                draw_box()
                pygame.display.update()
```

```
                pygame.time.delay(50)
        return False


# Display instruction for the game
def instruction():
        text1 = font2.render("PRESS D TO RESET TO DEFAULT / R TO EMPTY", 1, (0, 0, 0))
        text2 = font2.render("ENTER VALUES AND PRESS ENTER TO VISUALIZE", 1, (0, 0, 0))
        screen.blit(text1, (20, 520))
        screen.blit(text2, (20, 540))


# Display options when solved
def result():
        text1 = font1.render("FINISHED PRESS R or D", 1, (0, 0, 0))
        screen.blit(text1, (20, 570))
run = True
flag1 = 0
flag2 = 0
rs = 0
error = 0
# The loop thats keep the window running
while run:
```

```
# White color background
screen.fill((255, 255, 255))
# Loop through the events stored in event.get()
for event in pygame.event.get():
        # Quit the game window
        if event.type == pygame.QUIT:
                run = False
        # Get the mouse postion to insert number
        if event.type == pygame.MOUSEBUTTONDOWN:
                flag1 = 1
                pos = pygame.mouse.get_pos()
                get_cord(pos)
        # Get the number to be inserted if key pressed
        if event.type == pygame.KEYDOWN:
                if event.key == pygame.K_LEFT:
                        x-= 1
                        flag1 = 1
                if event.key == pygame.K_RIGHT:
                        x+= 1
                        flag1 = 1
                if event.key == pygame.K_UP:
                        y-= 1
```

```
            flag1 = 1
    if event.key == pygame.K_DOWN:
            y+= 1
            flag1 = 1
    if event.key == pygame.K_1:
            val = 1
    if event.key == pygame.K_2:
            val = 2
    if event.key == pygame.K_3:
            val = 3
    if event.key == pygame.K_4:
            val = 4
    if event.key == pygame.K_5:
            val = 5
    if event.key == pygame.K_6:
            val = 6
    if event.key == pygame.K_7:
            val = 7
    if event.key == pygame.K_8:
            val = 8
    if event.key == pygame.K_9:
            val = 9
    if event.key == pygame.K_RETURN:
```

```
            flag2 = 1
    # If R pressed clear the sudoku board
    if event.key == pygame.K_r:
            rs = 0
            error = 0
            flag2 = 0
            grid =[
            [0, 0, 0, 0, 0, 0, 0, 0, 0],
            [0, 0, 0, 0, 0, 0, 0, 0, 0],
            [0, 0, 0, 0, 0, 0, 0, 0, 0],
            [0, 0, 0, 0, 0, 0, 0, 0, 0],
            [0, 0, 0, 0, 0, 0, 0, 0, 0],
            [0, 0, 0, 0, 0, 0, 0, 0, 0],
            [0, 0, 0, 0, 0, 0, 0, 0, 0],
            [0, 0, 0, 0, 0, 0, 0, 0, 0],
            [0, 0, 0, 0, 0, 0, 0, 0, 0]
            ]
    # If D is pressed reset the board to default
    if event.key == pygame.K_d:
            rs = 0
            error = 0
            flag2 = 0
            grid =[
```

```
                    [7, 8, 0, 4, 0, 0, 1, 2, 0],
                    [6, 0, 0, 0, 7, 5, 0, 0, 9],
                    [0, 0, 0, 6, 0, 1, 0, 7, 8],
                    [0, 0, 7, 0, 4, 0, 2, 6, 0],
                    [0, 0, 1, 0, 5, 0, 9, 3, 0],
                    [9, 0, 4, 0, 6, 0, 0, 0, 5],
                    [0, 7, 0, 3, 0, 0, 0, 1, 2],
                    [1, 2, 0, 0, 0, 7, 4, 0, 0],
                    [0, 4, 9, 2, 0, 6, 0, 0, 7]
                ]
if flag2 == 1:
        if solve(grid, 0, 0)== False:
                error = 1
        else:
                rs = 1
        flag2 = 0
if val != 0:
        draw_val(val)
        # print(x)
        # print(y)
        if valid(grid, int(x), int(y), val)== True:
                grid[int(x)][int(y)]= val
                flag1 = 0
```

```
            else:
                grid[int(x)][int(y)]= 0
                raise_error2()
            val = 0

        if error == 1:
            raise_error1()
        if rs == 1:
            result()
        draw()
        if flag1 == 1:
            draw_box()
        instruction()

        # Update window
        pygame.display.update()

# Quit pygame window
pygame.quit()
```