EXPERIMENT NO - 10 DATE: 19/4/22

AIM OF THE EXPERIMENT: WRITE A PROGRAM IN PYTHON TO SOLVE ANY ONE AI PROBLEM

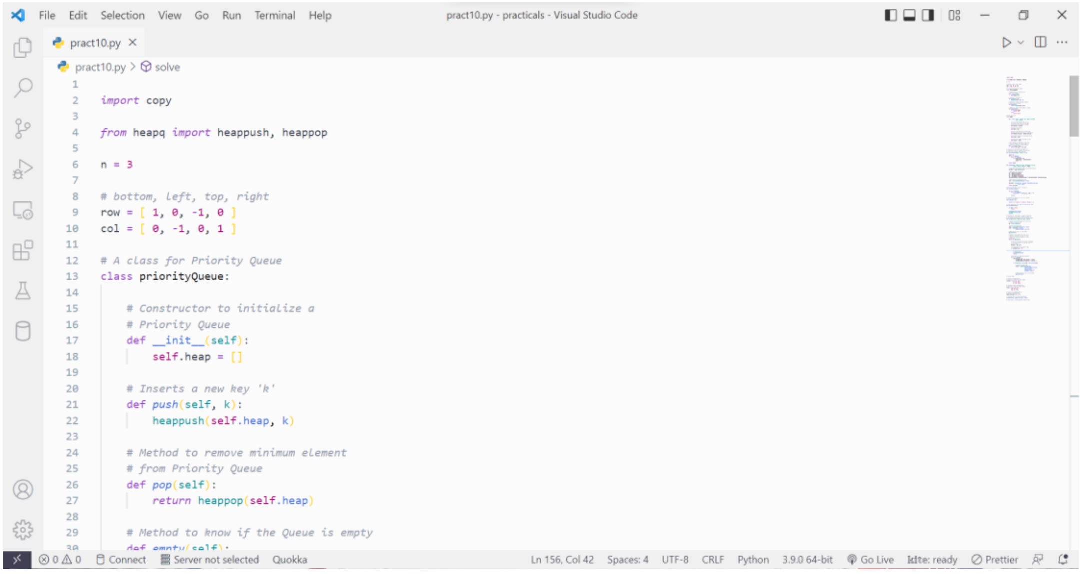
THEORY:

8 Puzzle Problem:

Given a 3×3 board with 8 tiles (every tile has one number from 1 to 8) and one empty space. The objective is to place the numbers on tiles to match the final configuration using the empty space. We can slide four adjacent (left, right, above, and below) tiles into the empty space.

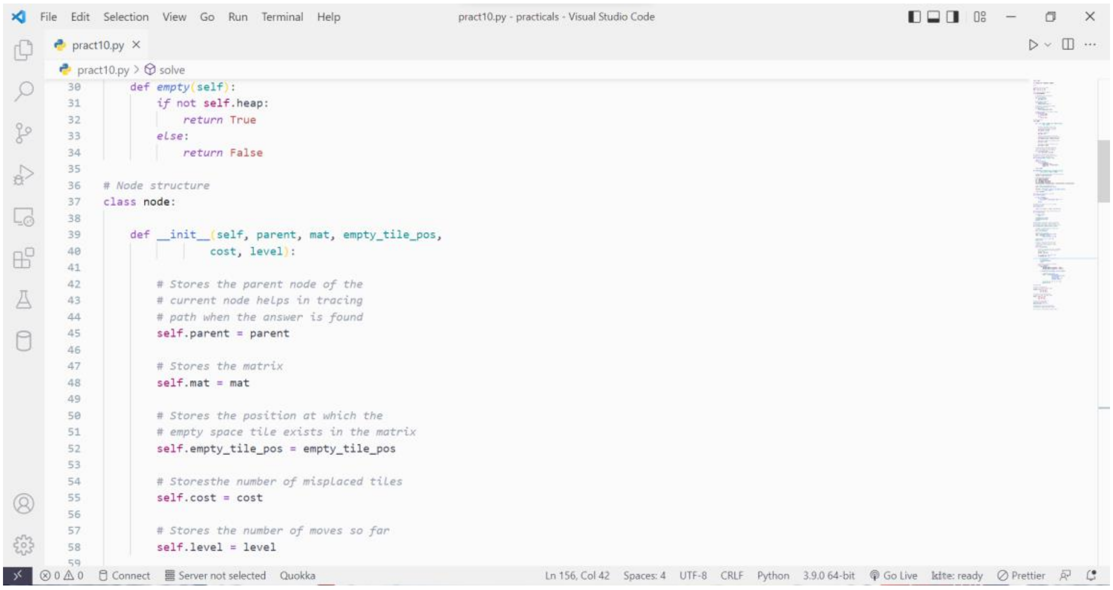
SOURCE CODE:

Below is the code of 8 Puzzle Problem



```python

import copy

from heapq import heappush, heappop

n = 3

# bottom, left, top, right
row = [ 1, 0, -1, 0 ]
col = [ 0, -1, 0, 1 ]

# A class for Priority Queue
class priorityQueue:

    # Constructor to initialize a
    # Priority Queue
    def __init__(self):
        self.heap = []

    # Inserts a new key 'k'
    def push(self, k):
        heappush(self.heap, k)

    # Method to remove minimum element
    # from Priority Queue
    def pop(self):
        return heappop(self.heap)

    # Method to know if the Queue is empty
```

Fig 10.1 Source Code



```python
    def empty(self):
        if not self.heap:
            return True
        else:
            return False

# Node structure
class node:

    def __init__(self, parent, mat, empty_tile_pos,
                 cost, level):

        # Stores the parent node of the
        # current node helps in tracing
        # path when the answer is found
        self.parent = parent

        # Stores the matrix
        self.mat = mat

        # Stores the position at which the
        # empty space tile exists in the matrix
        self.empty_tile_pos = empty_tile_pos

        # Storesthe number of misplaced tiles
        self.cost = cost

        # Stores the number of moves so far
        self.level = level
```

Fig 10.2 Source Code

```python
    # This method is defined so that the
    # priority queue is formed based on
    # the cost variable of the objects
    def __lt__(self, nxt):
        return self.cost < nxt.cost

# Function to calculate the number of
# misplaced tiles ie. number of non-blank
# tiles not in their goal position
def calculateCost(mat, final) -> int:

    count = 0
    for i in range(n):
        for j in range(n):
            if ((mat[i][j]) and
                (mat[i][j] != final[i][j])):
                count += 1

    return count

def newNode(mat, empty_tile_pos, new_empty_tile_pos,
            level, parent, final) -> node:

    # Copy data from parent matrix to current matrix
    new_mat = copy.deepcopy(mat)

    # Move tile by 1 position
    x1 = empty_tile_pos[0]
    y1 = empty_tile_pos[1]
```

Fig 10.3 Source Code

```python
    x2 = new_empty_tile_pos[0]
    y2 = new_empty_tile_pos[1]
    new_mat[x1][y1], new_mat[x2][y2] = new_mat[x2][y2], new_mat[x1][y1]

    # Set number of misplaced tiles
    cost = calculateCost(new_mat, final)

    new_node = node(parent, new_mat, new_empty_tile_pos,
                    cost, level)
    return new_node

# Function to print the N x N matrix
def printMatrix(mat):

    for i in range(n):
        for j in range(n):
            print("%d " % (mat[i][j]), end = " ")

        print()

# Function to check if (x, y) is a valid
# matrix coordinate
def isSafe(x, y):

    return x >= 0 and x < n and y >= 0 and y < n

# Print path from root node to destination node
def printPath(root):

    if root == None:
```

Fig 10.4 Source Code

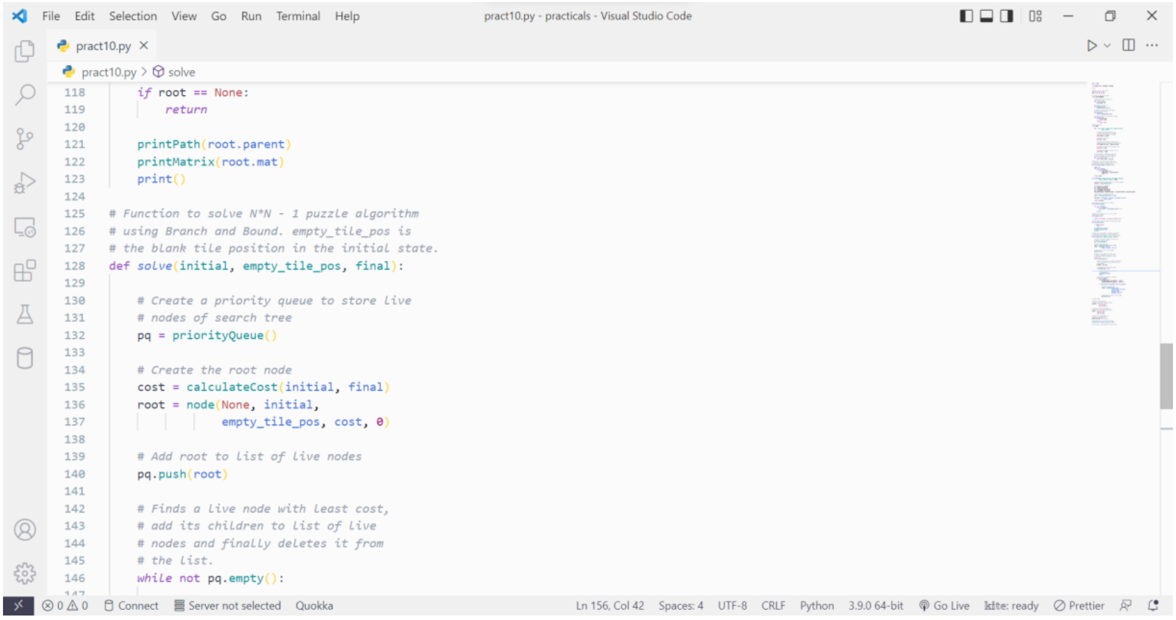


Fig 10.5 Source Code

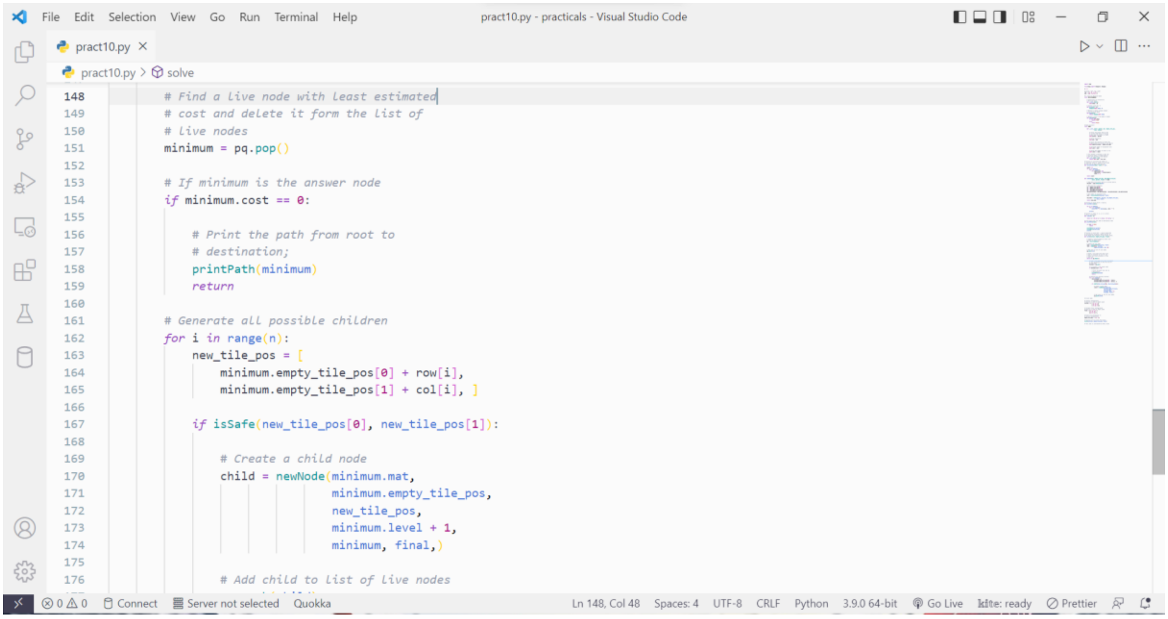
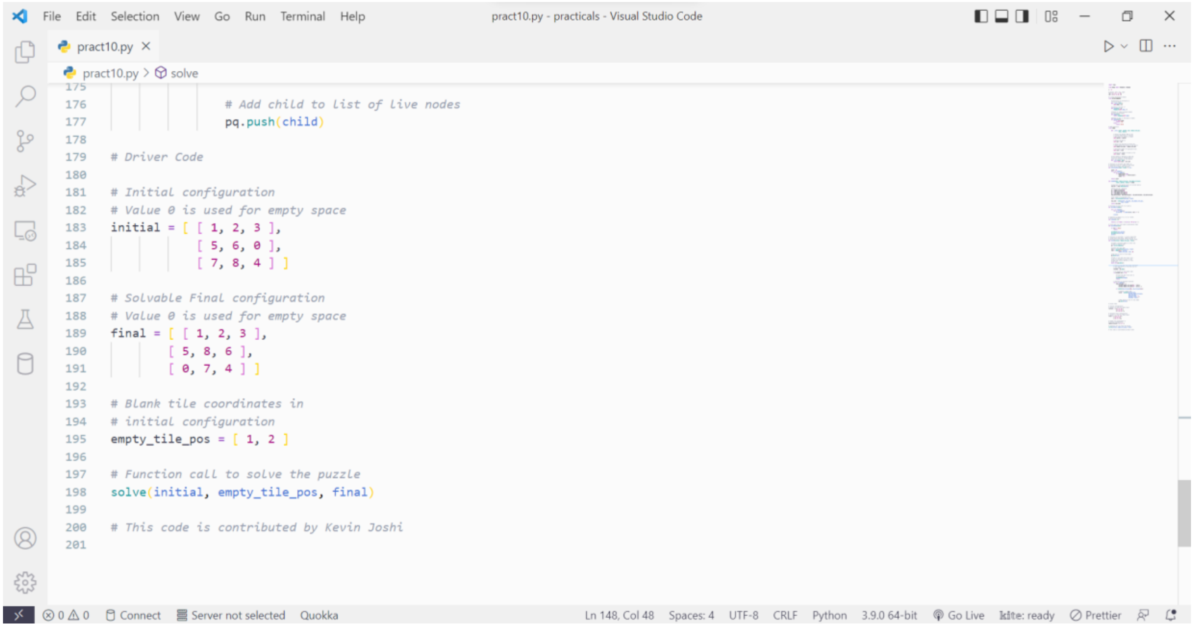


Fig 10.6 Source Code



```python
                # Add child to list of live nodes
                pq.push(child)

# Driver Code

# Initial configuration
# Value 0 is used for empty space
initial = [ [ 1, 2, 3 ],
            [ 5, 6, 0 ],
            [ 7, 8, 4 ] ]

# Solvable Final configuration
# Value 0 is used for empty space
final = [ [ 1, 2, 3 ],
          [ 5, 8, 6 ],
          [ 0, 7, 4 ] ]

# Blank tile coordinates in
# initial configuration
empty_tile_pos = [ 1, 2 ]

# Function call to solve the puzzle
solve(initial, empty_tile_pos, final)

# This code is contributed by Kevin Joshi
```

Fig 10.7 Source Code

OUTPUT:

```
Microsoft Windows [Version 10.0.22000.613]
(c) Microsoft Corporation. All rights reserved.

D:\B.Tech\6th semester\AI lab\practicals>python -u "d:\B.Tech\6th semester\AI lab\practicals\pract10.py"
1  2  3
5  6  0
7  8  4

1  2  3
5  0  6
7  8  4

1  2  3
1  2  3
5  8  6
7  0  4

1  2  3
5  8  6
0  7  4

D:\B.Tech\6th semester\AI lab\practicals>
```

Fig 10.8 Output