Problem Set-4

M.Vishnupriya

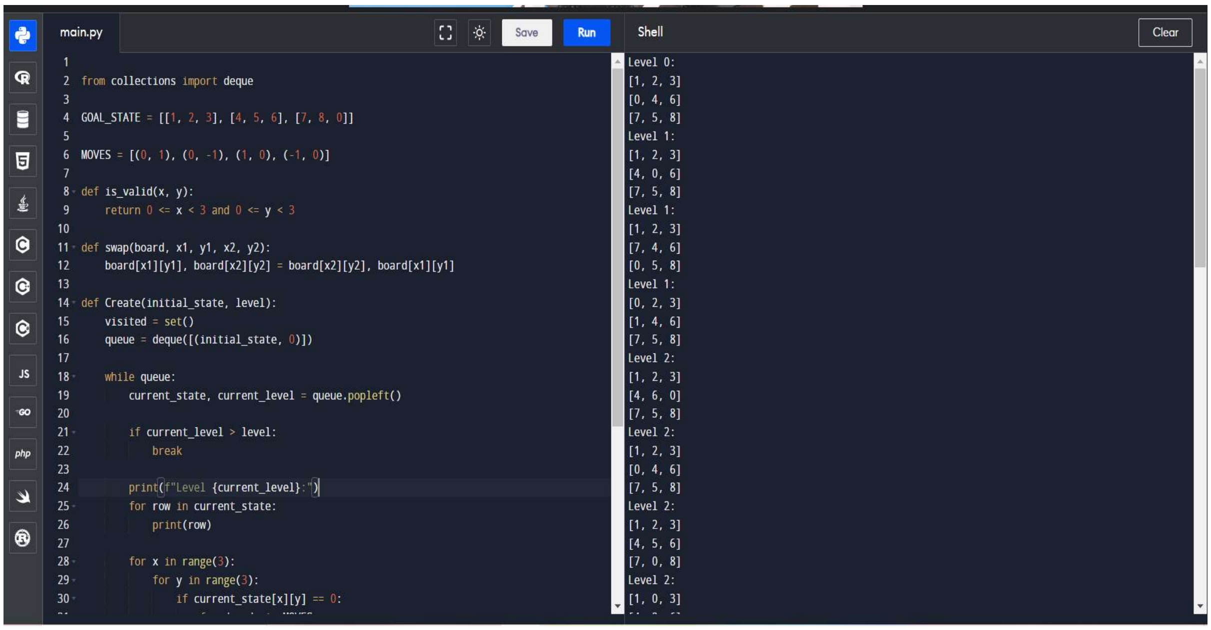
AP21110010134

CSE-C

Code:-

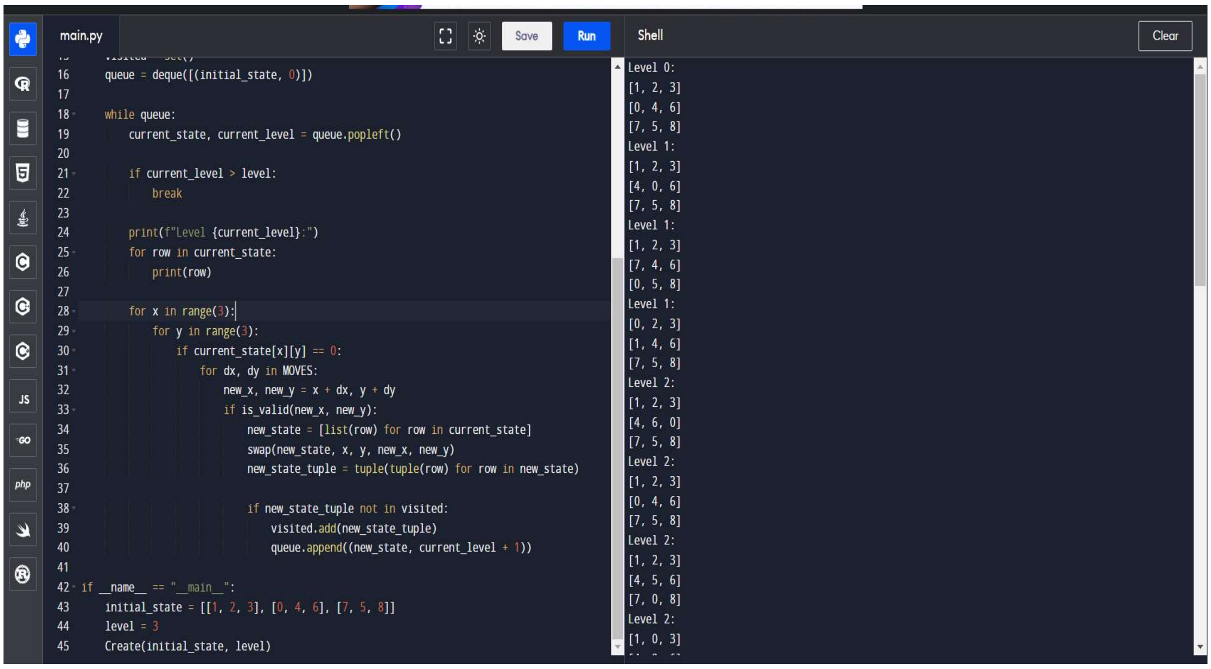
```
from collections import deque

GOAL_STATE = [[1, 2, 3], [4, 5, 6], [7, 8, 0]]

MOVES = [(0, 1), (0, -1), (1, 0), (-1, 0)]

def is_valid(x, y):
    return 0 <= x < 3 and 0 <= y < 3

def swap(board, x1, y1, x2, y2):
    board[x1][y1], board[x2][y2] = board[x2][y2], board[x1][y1]

def Create(initial_state, level):
    visited = set()
    queue = deque([(initial_state, 0)])

    while queue:
        current_state, current_level = queue.popleft()

        if current_level > level:
            break

        print(f"Level {current_level}:")
        for row in current_state:
            print(row)

        for x in range(3):
            for y in range(3):
                if current_state[x][y] == 0:
                    for dx, dy in MOVES:
                        new_x, new_y = x + dx, y + dy
                        if is_valid(new_x, new_y):
                            new_state = [list(row) for row in current_state]
                            swap(new_state, x, y, new_x, new_y)
                            new_state_tuple = tuple(tuple(row) for row in new_state)

                            if new_state_tuple not in visited:
                                visited.add(new_state_tuple)
                                queue.append((new_state, current_level + 1))

if __name__ == "__main__":
    initial_state = [[1, 2, 3], [0, 4, 6], [7, 5, 8]]
    level = 3
    Create(initial_state, level)
```

Input:-



```python

from collections import deque

GOAL_STATE = [[1, 2, 3], [4, 5, 6], [7, 8, 0]]

MOVES = [(0, 1), (0, -1), (1, 0), (-1, 0)]

def is_valid(x, y):
    return 0 <= x < 3 and 0 <= y < 3

def swap(board, x1, y1, x2, y2):
    board[x1][y1], board[x2][y2] = board[x2][y2], board[x1][y1]

def Create(initial_state, level):
    visited = set()
    queue = deque([(initial_state, 0)])

    while queue:
        current_state, current_level = queue.popleft()

        if current_level > level:
            break

        print(f"Level {current_level}:")
        for row in current_state:
            print(row)

        for x in range(3):
            for y in range(3):
                if current_state[x][y] == 0:
```

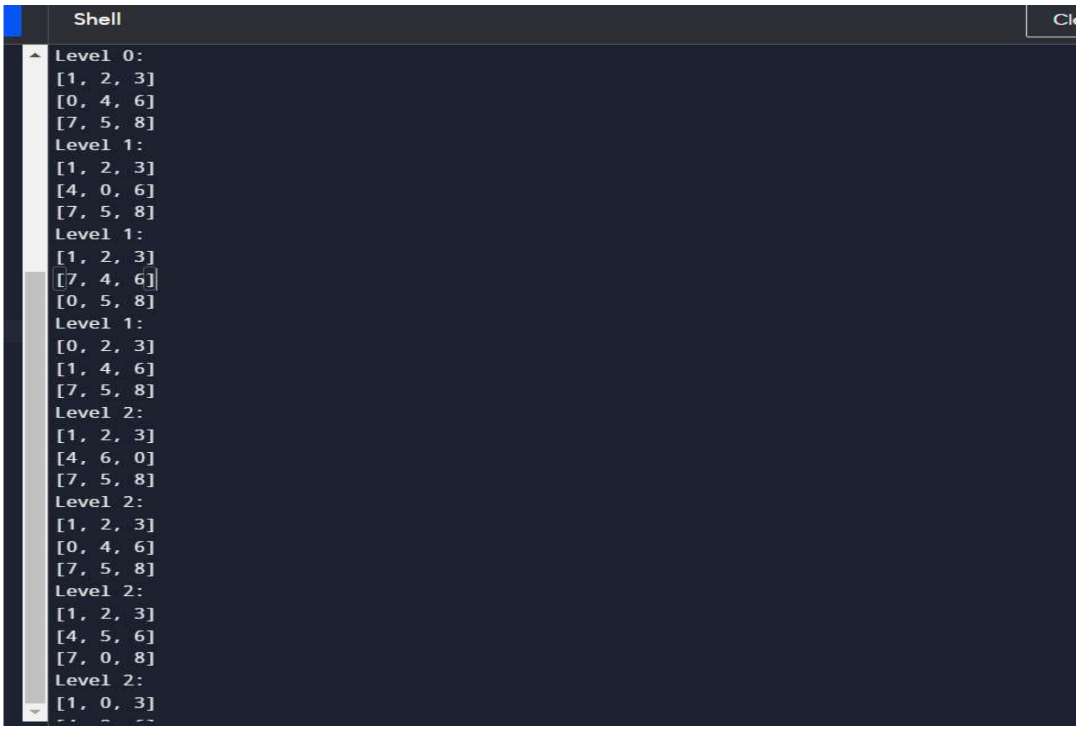
```
Level 0:
[1, 2, 3]
[0, 4, 6]
[7, 5, 8]
Level 1:
[1, 2, 3]
[4, 0, 6]
[7, 5, 8]
Level 1:
[1, 2, 3]
[7, 4, 6]
[0, 5, 8]
Level 1:
[0, 2, 3]
[1, 4, 6]
[7, 5, 8]
Level 2:
[1, 2, 3]
[4, 6, 0]
[7, 5, 8]
Level 2:
[1, 2, 3]
[0, 4, 6]
[7, 5, 8]
Level 2:
[1, 2, 3]
[4, 5, 6]
[7, 0, 8]
Level 2:
[1, 0, 3]
```



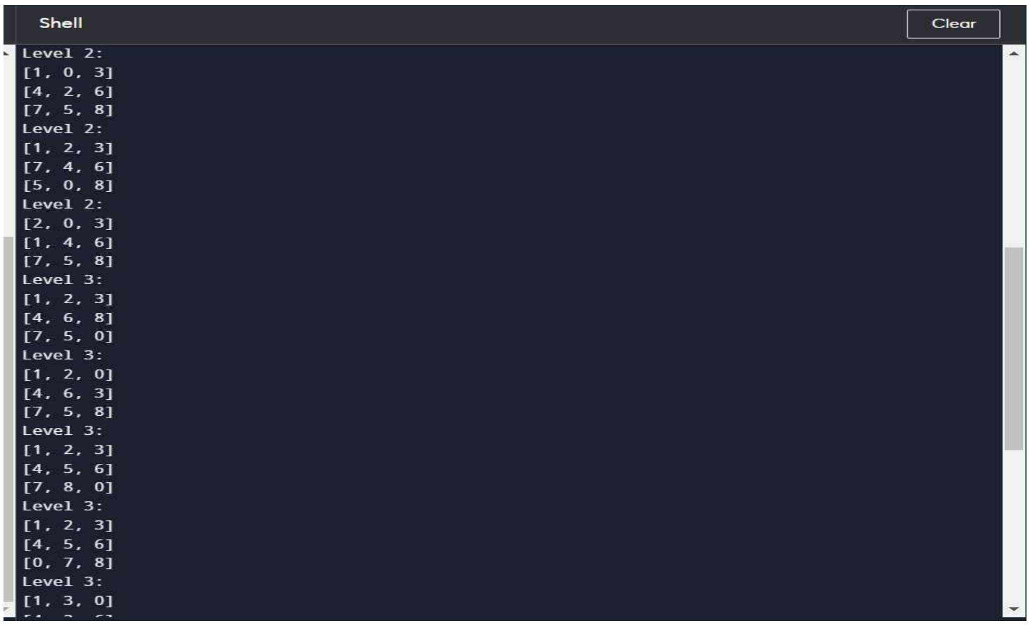
```python
    queue = deque([(initial_state, 0)])

    while queue:
        current_state, current_level = queue.popleft()

        if current_level > level:
            break

        print(f"Level {current_level}:")
        for row in current_state:
            print(row)

        for x in range(3):
            for y in range(3):
                if current_state[x][y] == 0:
                    for dx, dy in MOVES:
                        new_x, new_y = x + dx, y + dy
                        if is_valid(new_x, new_y):
                            new_state = [list(row) for row in current_state]
                            swap(new_state, x, y, new_x, new_y)
                            new_state_tuple = tuple(tuple(row) for row in new_state)

                            if new_state_tuple not in visited:
                                visited.add(new_state_tuple)
                                queue.append((new_state, current_level + 1))

if __name__ == "__main__":
    initial_state = [[1, 2, 3], [0, 4, 6], [7, 5, 8]]
    level = 3
    Create(initial_state, level)
```

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

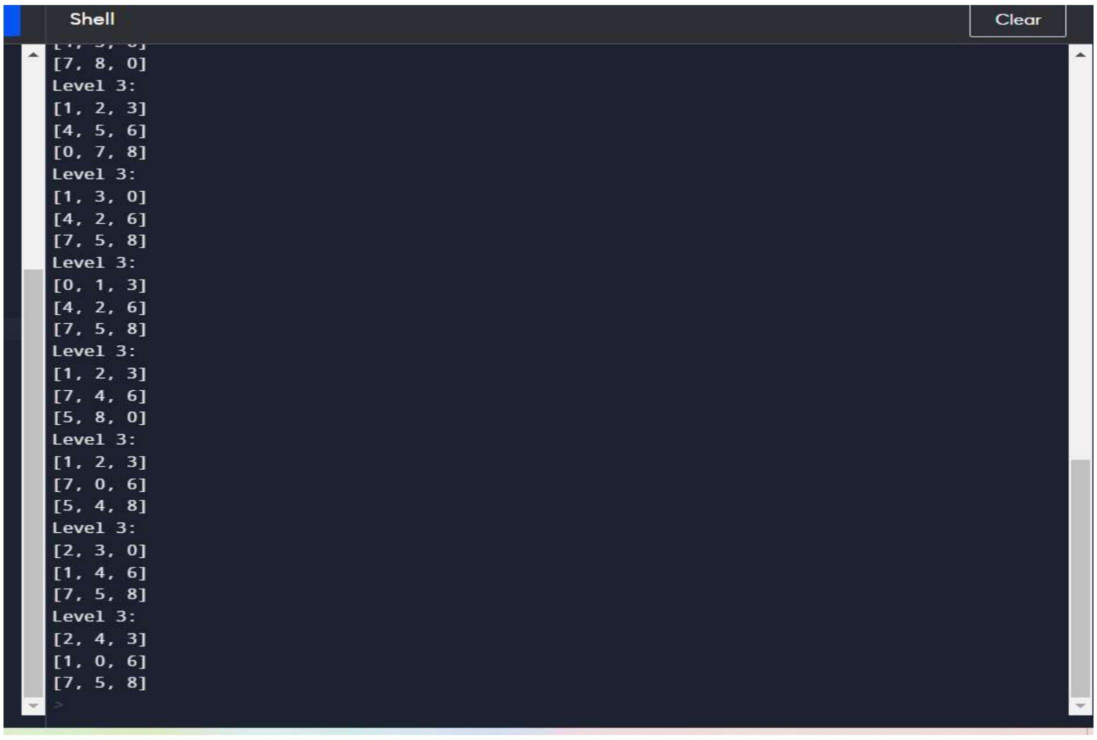
Output:-



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



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



```
[7, 8, 0]
Level 3:
[1, 2, 3]
[4, 5, 6]
[0, 7, 8]
Level 3:
[1, 3, 0]
[4, 2, 6]
[7, 5, 8]
Level 3:
[0, 1, 3]
[4, 2, 6]
[7, 5, 8]
Level 3:
[1, 2, 3]
[7, 4, 6]
[5, 8, 0]
Level 3:
[1, 2, 3]
[7, 0, 6]
[5, 4, 8]
Level 3:
[2, 3, 0]
[1, 4, 6]
[7, 5, 8]
Level 3:
[2, 4, 3]
[1, 0, 6]
[7, 5, 8]
>
```