Program 4

8 puzzle using Iterative Deepening search

```
def dfs(route, depth):
    if depth == 0:
        return
    if route[-1] == goal:
        return route
    for mov in get_moves(route[-1]):
        if mov not in route:
            next_route = dfs(route + [mov], depth-1)
            if next_route:
                return next_route
    for depth in itertools.count():
        route = dfs([puzzle], depth)
        if route:
            return route
```

```
def possible_moves(state):
    b = state.index(0)
    d = []
    if b not in [0,1,2]:
        d.append('u')
    if b not in [6,7,8]:
        d.append('d')
    if b not in [0,3,6]:
        d.append('l')
    if b not in [2,5,8]:
        d.append('r')
    pos_moves = []
    for i in d:
        pos_moves.append(generate(state, i, b))
    return pos_moves

def generate(state, m, b):
    temp = state.copy()
    if m == 'd':
        temp[b+3], temp[b] = temp[b], temp[b+3]
    if m == 'u':
        temp[b-3], temp[b] = temp[b], temp[b-3]
    if m == 'l':
        temp[b-1], temp[b] = temp[b], temp[b-1]
    if m == 'r':
        temp[b+1], temp[b] = temp[b], temp[b+1]
    return temp

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

route = id_dfs(initial, goal, pos_moves)
if route:
    print("Success! 9! is possible")
    print("Path:", route)
else:
    print("Fail!")
```

Output:

Success! 9! is possible 8 puzzle program

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

```
Harshala Rani-1bm21cs074
Success!! It is possible to solve 8 Puzzle problem
Path: [[1, 2, 3, 0, 4, 6, 7, 5, 8], [1, 2, 3, 4, 0, 6, 7, 5, 8], [1, 2, 3, 4, 5, 6, 7, 0, 8], [1, 2, 3, 4, 5, 6, 7, 8, 0]]
```