Program - #1

Shaan Subbaiah B C
1BM18CS096

AI LAB

13. NOV. 20

```
# To calc Heuristic → f(x) = h(x) + g(x)
def f( self, start , goal ):
    return self.h( start.data, goal ) + start.level

# To calc h(x) using manhatten dist.
def h( self, start, goal ):
    ~~tmp = 0~~ dist = 0
    ~~for (i in range (0, self.n)):~~
        ~~for( j in range (0, self.n)):~~
    for( i in range (0, self.n)):
        d1, d2 = grid.index(i), target.index(i)
        x1, y1 = d1 % 3 , d1 // 3
        x2, y2 = d2 % 3 , d2 // 3
        dist += abs (x1 - x2) + abs ( y1 - y2 )
    return dist

# main function for astar
def astar ( state, target ):
    frontier = [Node (state , 1 )]
    while frontier :
        frontier . sort ( key = lambda x : x.cost)
```

Shaan

```
state = frontier.pop(0)
print(f'Level : {state.level}')
printGrid(state)
count += 1          # increment level
if(state.grid == target:
    print(f"Success").
    return

if (count >= 3):
    print(f"NOSOLUTION")
    return

neighbours = possible_moves(state)
    for neighbour in neighbours:
        neighbour = Node(neighbour, state.level+1)
        neighbour.cost = F(neighbour, target)

        if not inFrontier(frontier, neighbour):
            frontier.append(neighbour)

print(f"Fail").
```