Hemanth Kumar V.C
1BM18CS037
sec 'A'
Date: 10/11/2020

AI - LAB

program 1:-

* Implement A* search

```
class Node:
    def __init__(self, data, level, fval):

    def shuffle(self, puz, x1, y1, x2, y2):
        """ Move the blank space in the given direction and if the position
            value are out of limits the return none """
        if x2 >= 0 & x2 < len(self.data) and y2 >= 0 & y2 < len(self.data):
            temp_puz = []
            temp_puz = self.copy(puz)
            temp = temp_puz[x2][y2]
            temp_puz[x2][y2] = temp_puz[x1][y1]
            temp_puz[x1][y1] = temp
            return temp_puz
        else:
            return None

    def copy(self, root):
        temp = []
        for i in root:
            t = []
            for j in i:
                t.append(j)
```


Scanned with CamScanner

```
            temp.append(k)
            return temp

    def find(self, puz, x):
        for i in range(0, len(self.data)):
            for j in range(0, len(self.data)):
                if puz[i][j] == x:
                    return i,j

class Puzzle:
    def __init__(self, size):
        self.n = size
        self.open = []
        self.closed = []
    def accept(self):
        puz = []
        for i in range(0, self.n):
            temp = input().split(" ")
            puz.append(temp)
        return puz

    def f(self, start, goal):
        return self.h(start.data, goal) + start.level

    def h(self, start, goal):
        temp = 0
        for i in range(0, self.n):
            for j in range(0, self.n):
                if start[i][j] != goal[i][j] and start[i][j] != '_':
                    temp += 1
        return temp
```


Scanned with CamScanner

Hemanth kumar V.C
1BM18CS037
Sec 'A'
Date: 10/11/2020.

AI-LAB.

```
def process(self):
    print("Enter the start state matrix \n")
    start = self.accept()
    print("Enter the goal state matrix \n")
    goal = self.accept()

start = Node(start,0,0)
start.fval = self.f(start,goal)
self.open.append(start)
print("\n\n")
while True:
    cur = self.open[0]
    print(" ")
    print(" | ")
    print(" \|/ ")
    print(" /|\ ")
    for i in cur.data:
        for j in i:
            print(j, end=" ")
        print(" ")
    if(self.h(cur.data,goal)==0):
        break
    for i in cur.generate_child():
        i.fval = self.f(i,goal)
        self.open.append(i)
```


Scanned with CamScanner

```
        self.closed.append(cur)
        del self.open[0]

        self.open.sort(key = lambda x: x.fval, reverse = False)

        i
puz = Puzzle(3)
puz.process()

print("\n A* SEARCH METHOD")
```


Scanned with CamScanner