Practical No: 2 Roll No:

Subject: Artificial Intelligence

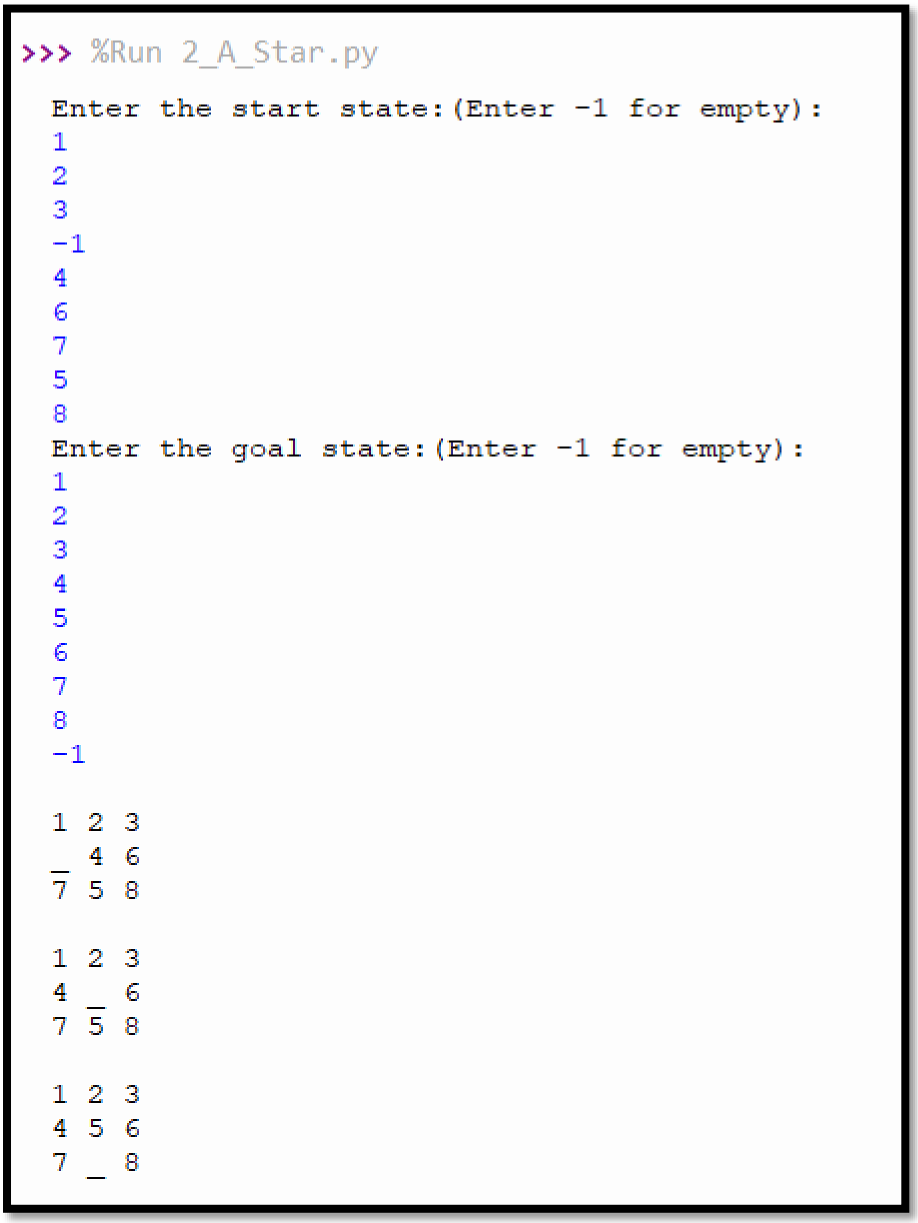
Title: Implement A* Algorithm for any game search problem

Program Code :

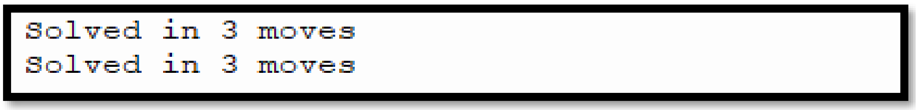
```
g = 0
def print_board(elements):
   for i in range(9):
      if i % 3 == 0:
         print()
      if elements[i] == -1:
         print("_", end=" ")
      else:
         print(elements[i], end=" ")
   print()

def solvable(start):
   inv = 0

   for i in range(9):
      if start[i] == -1:
         continue
      for j in range(i + 1, 9):
         if start[j] == -1:
            continue
         if start[i] > start[j]:
            inv += 1
   return inv % 2 == 0

def heuristic(start, goal):
   global g
   h = 0
   for i in range(9):
      if start[i] != -1:
         h += abs(goal.index(start[i]) - i) // 3 + abs(goal.index(start[i]) - i) % 3
   return h + g

def moveleft(start, position):
   start[position], start[position - 1] = start[position - 1], start[position]

def moveright(start, position):
   start[position], start[position + 1] = start[position + 1], start[position]

def moveup(start, position):
   start[position], start[position - 3] = start[position - 3], start[position]
```

```
def movedown(start, position):
    start[position], start[position + 3] = start[position + 3], start[position]

def movetile(start, goal):
    emptyat = start.index(-1)
    row = emptyat // 3
    col = emptyat % 3
    t1, t2, t3, t4 = start[:], start[:], start[:], start[:]
    f1, f2, f3, f4 = 100, 100, 100, 100

    if col - 1 >= 0:
        moveleft(t1, emptyat)
        f1 = heuristic(t1, goal)
    if col + 1 < 3:
        moveright(t2, emptyat)
        f2 = heuristic(t2, goal)
    if row + 1 < 3:
        movedown(t3, emptyat)
        f3 = heuristic(t3, goal)
    if row - 1 >= 0:
        moveup(t4, emptyat)
        f4 = heuristic(t4, goal)

    min_heuristic = min(f1, f2, f3, f4)

    if f1 == min_heuristic:
        moveleft(start, emptyat)
    elif f2 == min_heuristic:
        moveright(start, emptyat)
    elif f3 == min_heuristic:
        movedown(start, emptyat)
    elif f4 == min_heuristic:
        moveup(start, emptyat)

def solveEight(start, goal):
    global g
    g += 1
    movetile(start, goal)
    print_board(start)
    f = heuristic(start, goal)
    if f == g:
        print("Solved in {} moves".format(f))
        return
    solveEight(start, goal)

def main():
    global g
    start = []
    goal = []
    print("Enter the start state:(Enter -1 for empty):")
    for _ in range(9):
        start.append(int(input()))

    print("Enter the goal state:(Enter -1 for empty):")
    for _ in range(9):
        goal.append(int(input()))

    print_board(start)

    # To check if solvable
    if solvable(start):
        solveEight(start, goal)
        print("Solved in {} moves".format(g))
    else:
        print("Not possible to solve")

if __name__ == '__main__':
    main()
```

Output :



```
>>> %Run 2_A_Star.py
  Enter the start state:(Enter -1 for empty):
  1
  2
  3
  -1
  4
  6
  7
  5
  8
  Enter the goal state:(Enter -1 for empty):
  1
  2
  3
  4
  5
  6
  7
  8
  -1

  1 2 3
  _ 4 6
  7 5 8

  1 2 3
  4 _ 6
  7 5 8

  1 2 3
  4 5 6
  7 _ 8
```



```
Solved in 3 moves
Solved in 3 moves
```