Name: Ankit raj Roll No.:-11233036 Department/Semester/Group: CSE/4/G1

EXPERIMENT NO :-04

AIM:- . Solve 8-puzzle problem using Best First Search. Write a program to Implement A*

```
import heapq

def a_star_search(start, goal):
   def heuristic(state):
      distance = 0
      for i in range(len(state)):
         if state[i] != 0:
            x1, y1 = divmod(i, 3)
            x2, y2 = divmod(goal.index(state[i]), 3)
            distance += abs(x1 - x2) + abs(y1 - y2)
      return distance

   def get_neighbors(state):
      neighbors = []
      zero_index = state.index(0)
      x, y = divmod(zero_index, 3)
      moves = [(-1, 0), (1, 0), (0, -1), (0, 1)]
      for dx, dy in moves:
         nx, ny = x + dx, y + dy
         if 0 <= nx < 3 and 0 <= ny < 3:
            new_index = nx * 3 + ny
            new_state = list(state)
            new_state[zero_index], new_state[new_index] = new_state[new_index], 
new_state[zero_index]
            neighbors.append((tuple(new_state), zero_index, new_index))
      return neighbors

   def move_description(prev_index, new_index):
      directions = {(-1, 0): "up", (1, 0): "down", (0, -1): "left", (0, 1): "right"}
      x1, y1 = divmod(prev_index, 3)
      x2, y2 = divmod(new_index, 3)
      dx, dy = x2 - x1, y2 - y1
      return directions[(dx, dy)]

   open_set = []
   heapq.heappush(open_set, (0 + heuristic(start), 0, start, [], None))
   visited = set()
   while open_set:
      f, g, current, path, last_move = heapq.heappop(open_set)
      if current in visited:
         continue
```

```
        visited.add(current)
        path = path + [(current, last_move)]
        if current == goal:
            return path
        for neighbor, zero_index, new_index in get_neighbors(current):
            if neighbor not in visited:
                move = move_description(zero_index, new_index)
                heapq.heappush(open_set, (g + 1 + heuristic(neighbor), g + 1, neighbor, path,
move))
    return None

start_state = (1, 2, 3, 4, 5, 6, 7, 8, 0)
goal_state = (1, 2, 3, 0, 5, 6, 7, 8, 4)

solution = a_star_search(start_state, goal_state)
if solution:
    print("Solution found:")
    for step_num, (step, move) in enumerate(solution):
        print(f"Step {step_num}: {move if move else 'Start'}")
        for i in range(0, 9, 3):
            print(step[i:i+3])
        print()
else:
    print("No solution exists.")
```

OUTPUT:-

```
E:\python\python.exe "C:\Users\Ankit raj\Documents\4th sem exp\puzzle.py"

Solution found:

Step 0: Start

(1, 2, 3)

(4, 5, 6)

(7, 8, 0)


Step 1: up

(1, 2, 3)

(4, 5, 0)

(7, 8, 6)
```

Step 2: left

(1, 2, 3)

(4, 0, 5)

(7, 8, 6)

Step 3: left

(1, 2, 3)

(0, 4, 5)

(7, 8, 6)

Step 4: down

(1, 2, 3)

(7, 4, 5)

(0, 8, 6)

Step 5: right

(1, 2, 3)

(7, 4, 5)

(8, 0, 6)

Step 6: up

(1, 2, 3)

(7, 0, 5)

(8, 4, 6)

Step 7: right

(1, 2, 3)

(7, 5, 0)

(8, 4, 6)

```
Step 8: down
(1, 2, 3)
(7, 5, 6)
(8, 4, 0)

Step 9: left
(1, 2, 3)
(7, 5, 6)
(8, 0, 4)

Step 10: left
(1, 2, 3)
(7, 5, 6)
(0, 8, 4)

Step 11: up
(1, 2, 3)
(0, 5, 6)
(7, 8, 4)
Process finished with exit code 0
```