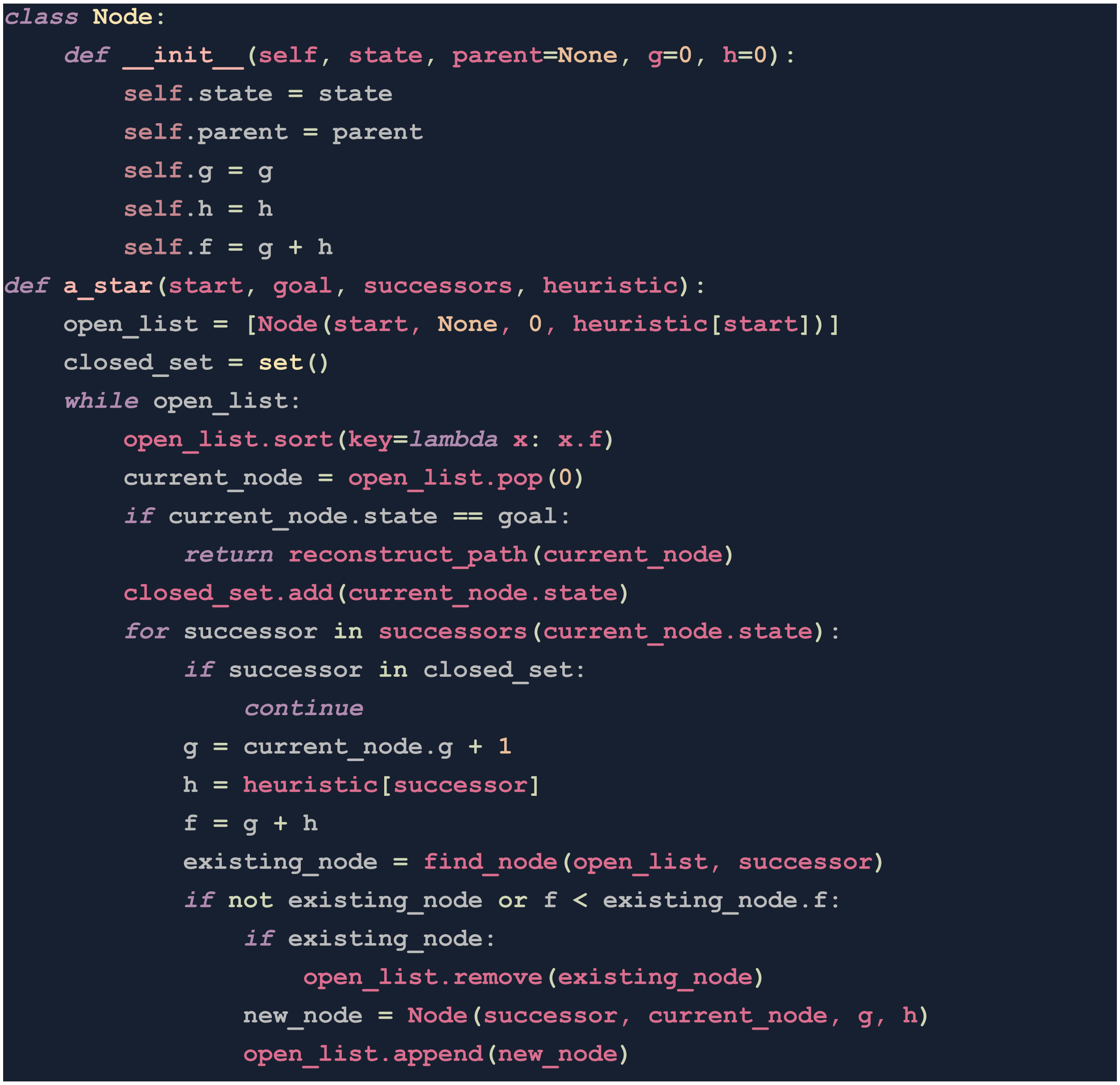
Batch: IAI-2	**Experiment Number: 3**
Roll Number: 16010422234	**Name: Chandana Ramesh Galgali**

Aim of the Experiment: Implementation of Informed search algorithm - A*

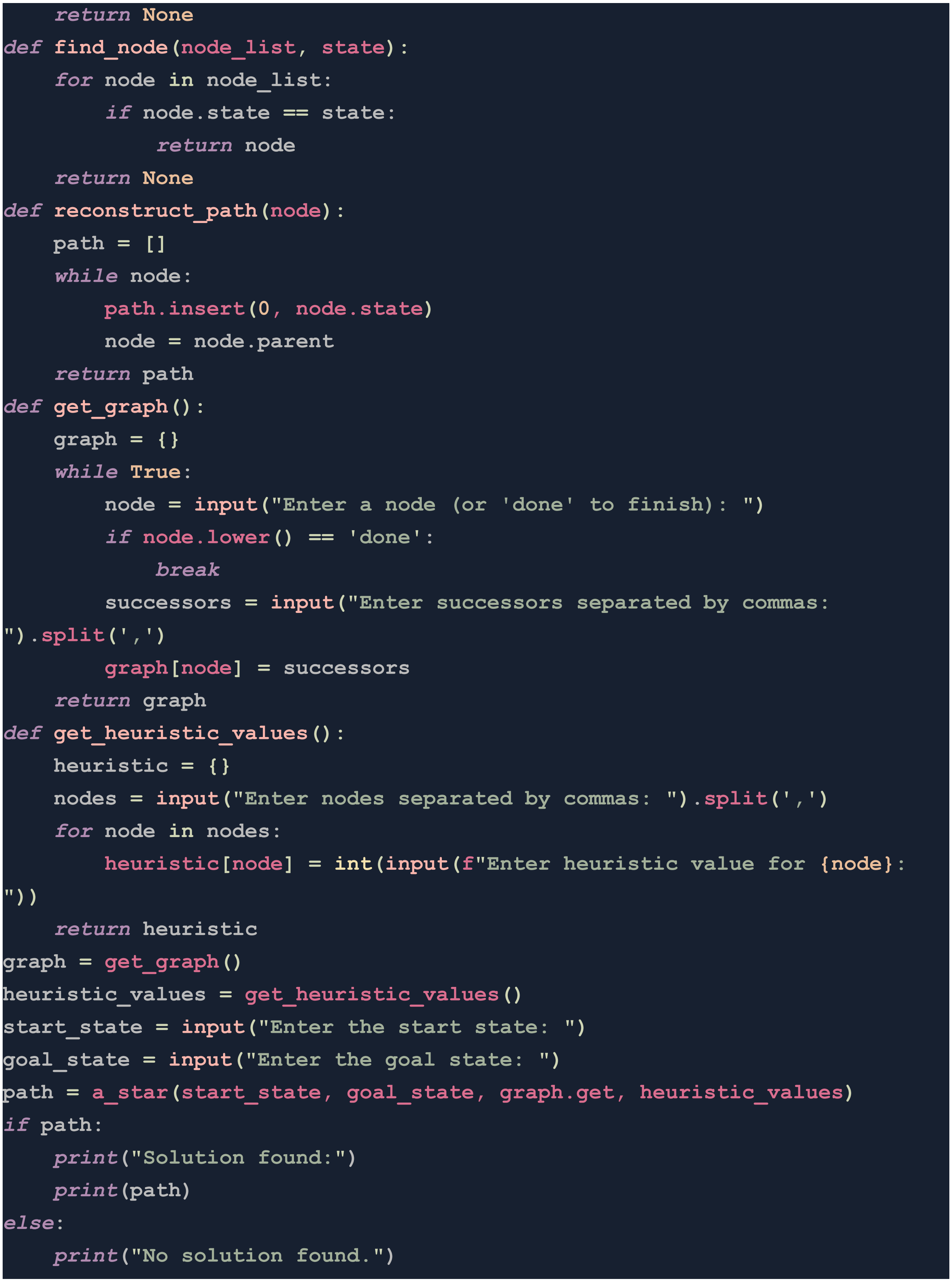
Program/Steps:

1. Implement A* algorithm as discussed for graph traversal.
2. Print the contents of fringe/OPEN, CLOSED/Visited and the solution.

Code:

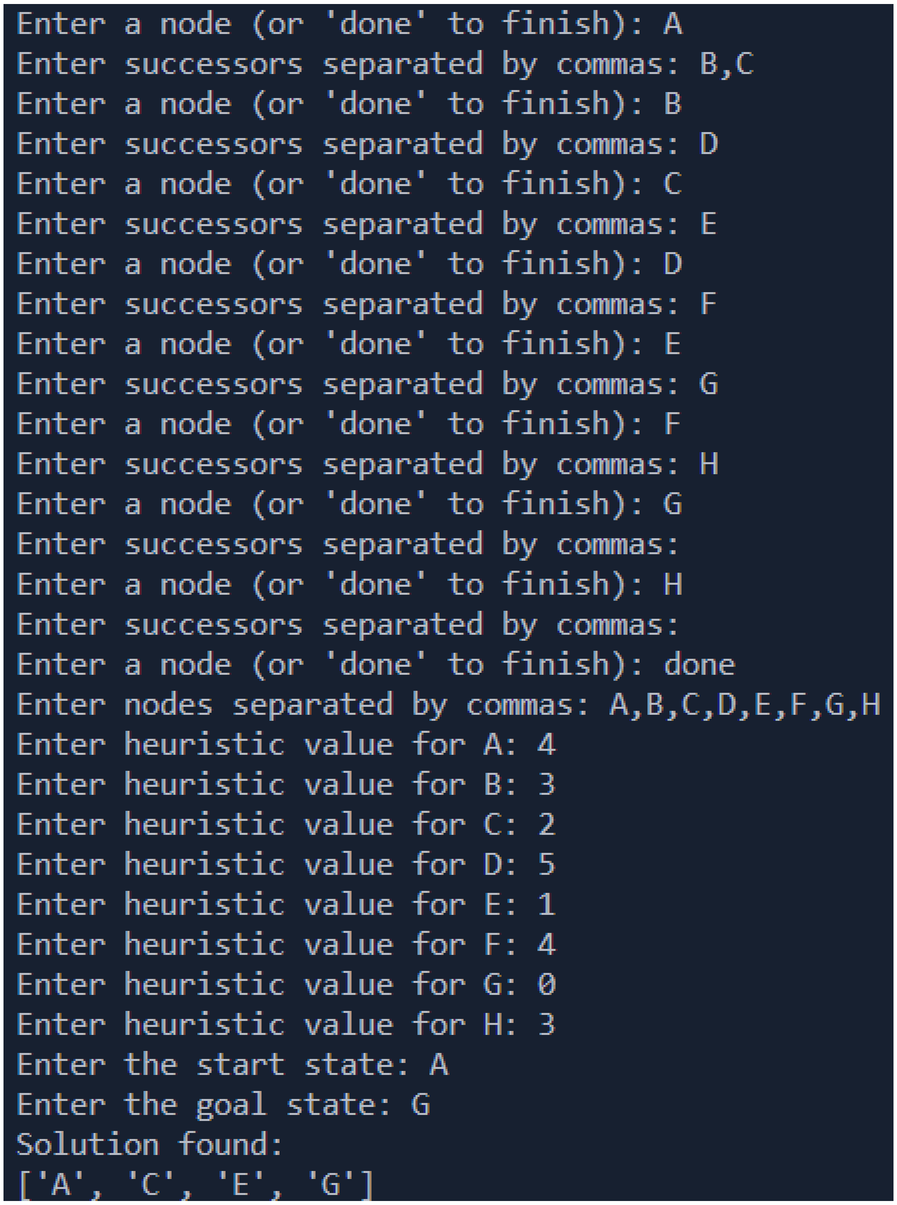


```python
class Node:
    def __init__(self, state, parent=None, g=0, h=0):
        self.state = state
        self.parent = parent
        self.g = g
        self.h = h
        self.f = g + h
def a_star(start, goal, successors, heuristic):
    open_list = [Node(start, None, 0, heuristic[start])]
    closed_set = set()
    while open_list:
        open_list.sort(key=lambda x: x.f)
        current_node = open_list.pop(0)
        if current_node.state == goal:
            return reconstruct_path(current_node)
        closed_set.add(current_node.state)
        for successor in successors(current_node.state):
            if successor in closed_set:
                continue
            g = current_node.g + 1
            h = heuristic[successor]
            f = g + h
            existing_node = find_node(open_list, successor)
            if not existing_node or f < existing_node.f:
                if existing_node:
                    open_list.remove(existing_node)
                new_node = Node(successor, current_node, g, h)
                open_list.append(new_node)
```



```
    return None
def find_node(node_list, state):
    for node in node_list:
        if node.state == state:
            return node
    return None
def reconstruct_path(node):
    path = []
    while node:
        path.insert(0, node.state)
        node = node.parent
    return path
def get_graph():
    graph = {}
    while True:
        node = input("Enter a node (or 'done' to finish): ")
        if node.lower() == 'done':
            break
        successors = input("Enter successors separated by commas: 
").split(',')
        graph[node] = successors
    return graph
def get_heuristic_values():
    heuristic = {}
    nodes = input("Enter nodes separated by commas: ").split(',')
    for node in nodes:
        heuristic[node] = int(input(f"Enter heuristic value for {node}: 
"))
    return heuristic
graph = get_graph()
heuristic_values = get_heuristic_values()
start_state = input("Enter the start state: ")
goal_state = input("Enter the goal state: ")
path = a_star(start_state, goal_state, graph.get, heuristic_values)
if path:
    print("Solution found:")
    print(path)
else:
    print("No solution found.")
```

Output/Result:



```
Enter a node (or 'done' to finish): A
Enter successors separated by commas: B,C
Enter a node (or 'done' to finish): B
Enter successors separated by commas: D
Enter a node (or 'done' to finish): C
Enter successors separated by commas: E
Enter a node (or 'done' to finish): D
Enter successors separated by commas: F
Enter a node (or 'done' to finish): E
Enter successors separated by commas: G
Enter a node (or 'done' to finish): F
Enter successors separated by commas: H
Enter a node (or 'done' to finish): G
Enter successors separated by commas:
Enter a node (or 'done' to finish): H
Enter successors separated by commas:
Enter a node (or 'done' to finish): done
Enter nodes separated by commas: A,B,C,D,E,F,G,H
Enter heuristic value for A: 4
Enter heuristic value for B: 3
Enter heuristic value for C: 2
Enter heuristic value for D: 5
Enter heuristic value for E: 1
Enter heuristic value for F: 4
Enter heuristic value for G: 0
Enter heuristic value for H: 3
Enter the start state: A
Enter the goal state: G
Solution found:
['A', 'C', 'E', 'G']
```

Outcomes: Analyze and formalize the problem (as a state space, graph, etc.) and select the appropriate search method and write the algorithm

Conclusion (Based on the Results and outcomes achieved):

The successful implementation of the A* algorithm demonstrated its capability to efficiently navigate through a search space, considering both the current cost and the estimated future cost. The algorithm is widely applicable in various fields, including artificial intelligence, robotics, and pathfinding applications.

References:

1. Stuart Russell and Peter Norvig, Artificial Intelligence: A Modern Approach, Second Edition, Pearson Publication
2. Luger, George F. Artificial Intelligence : Structures and strategies for complex problem solving, 2009, 6th Edition, Pearson Education