AI PRACTICAL NO. 7

Name:Girish Nhavkar

Roll no:9560

Div:TE COMPS A (BATCH B)

```
class BlockWorld:

    # Constructor
    def __init__(self, initial_state, goal_state):
        self.initial_state = initial_state
        self.goal_state = goal_state

    def evaluate_state(self, state):
        # Evaluate how close the state is to the goal state
        score = 0
        for block in state:
            # print(f"Block: {block}")
            # print(f"state[block]: {state[block]}")
            # We increment the score only if the state is same as goal state.
            if state[block] == self.goal_state[block]:
                score += 1
        return score


    def generate_neighbors(self, state):
        # Generate neighboring states by applying valid actions
        # For now we consider only 3 actions, move, stack or unstack.
        neighbors = []
        for action in ['move', 'stack', 'unstack']:
            for block in state:
                neighbor = state.copy()
                # print(neighbor)
                if action == 'move':
                    # Move the block to a different position
                    # For simplicity, let's assume we can move any block to
any position
                    neighbor[block] = (block[0], 'new_position')
                elif action == 'stack':
                    # Stack the block on top of another block
                    # For simplicity, let's assume we can stack any block on
any other block
                    neighbor[block] = ('on_top_of', 'existing_block')
                elif action == 'unstack':
                    # Unstack the block from the top of another block
                    # For simplicity, let's assume we can unstack any block
from any other block
                    neighbor[block] = ('on_table', None)
```

```
                neighbors.append(neighbor)
        return neighbors

    def hill_climbing(self):
        current_state = self.initial_state
        while True:
            current_score = self.evaluate_state(current_state)
            neighbors = self.generate_neighbors(current_state)
            best_neighbor = current_state
            best_score = current_score
            for neighbor in neighbors:
                neighbor_score = self.evaluate_state(neighbor)
                if neighbor_score > best_score:
                    best_neighbor = neighbor
                    best_score = neighbor_score
            if best_score <= current_score:
                # No better neighbor found
                break
            current_state = best_neighbor
        return current_state

# Example usage:
initial_state = {'A': ('on_table', None), 'B': ('on_top_of', 'A'), 'C':
('on_top_of', 'B'), 'D': ('on_top_of', 'C'), 'E': ('on_table', None), 'F':
('on_top_of', 'E'), 'G': ('on_top_of', 'F')}
goal_state = {'A': ('on_table', None), 'B': ('on_table', None), 'C':
('on_table', None), 'D': ('on_table', None), 'E': ('on_top_of', 'A'), 'F':
('on_top_of', 'E'), 'G': ('on_top_of', 'F')}
block_world = BlockWorld(initial_state, goal_state)
final_state = block_world.hill_climbing()
print("Final State:")
for block, position in final_state.items():
    print(f"Block {block}: {position[0]} {position[1]}")

state = {
    'A': ('on_table', None),  # Block A is on the table
    'B': ('on_top_of', 'A'),  # Block B is stacked on top of Block A
    'C': ('on_table', None)   # Block C is on the table
}

#output
# Final State:
# Block A: on_table None
# Block B: on_table None
# Block C: on_table None
# Block D: on_table None
# Block E: on_table None
# Block F: on_top_of E
```

```
# Block G: on_top_of F
```

OP:

```
PS C:\Girish\TE\AI> & "C:/Users/Girish Nhavkar/AppData/Local/Programs/Python/Python312/python.exe" c:/Girish/TE/AI/exp7/hillclimbalgo.py
Final State:
Block A: on_table None
Block B: on_table None
Block C: on_table None
Block D: on_table None
Block E: on_table None
Block F: on_top_of E
Block G: on_top_of F
PS C:\Girish\TE\AI>
```