Fr. Conceicao Rodrigues College of Engineering Fr. Agnel Ashram, Bandstand, Bandra (W), Mumbai - 400050

Department of Computer Engineering Academic Term II: 23-24

Class: B.E (Computer), Sem – VI Subject Name: Artificial Intelligence Student

Name: Nimish Ravindra Patil **Roll No: 9565**

Practical No:	**7**
Title:	Block World Problem solving by hill climbing approach
Date of Performance:	**18/03/2024**
Date of Submission:	**25/03/2024**

Rubrics for Evaluation:

Sr. No	Performance Indicator	Excellent	Good	Below Average	Marks
1	On time Completion & Submission (01)	01 (On Time)	NA	00 (Not on Time)	
2	Logic/Algorithm Complexity analysis (03)	03(Correct)	02(Partial)	01 (Tried)	
3	Coding Standards (03): Comments/indention/Naming conventions Test Cases /Output	03(All used)	02 (Partial)	01 (rarely followed)	
4	Post Lab Assignment (03)	03(done well)	2 (Partially Correct)	1(submitted)	
Total					

Signature of the Teacher:

Source code:

```
class BlockWorldProblem:
    def __init__(self, initial_state, goal_score):
        self.current_state = initial_state
        self.goal_score = goal_score

    def evaluate_state(self, state): score = 0
        for block, resting_place in
        state.items():
            if block == resting_place:
                score += 1
            else:
                score -= 1
        return score

    def find_possible_moves(self):
        possible_moves = [] for block in
        self.current_state.keys():
            for resting_place in self.current_state.keys():
                if block != resting_place:
                    possible_moves.append((block, resting_place))
        return possible_moves

    def make_move(self, move): new_state
        = self.current_state.copy() block,
        resting_place = move
        new_state[block] = resting_place
        return new_state

    def hill_climbing(self, max_iterations=9999):
        current_score = self.evaluate_state(self.current_state)
        iterations = 0

        print("Initial State:") for block, resting_place in
        self.current_state.items(): print(f"Block {block} is on
        {resting_place}")

        while iterations < max_iterations:
            possible_moves = self.find_possible_moves() new_states =
            [self.make_move(move) for move in possible_moves] best_state =
            max(new_states, key=self.evaluate_state) best_score =
            self.evaluate_state(best_state)
```

```
            if best_score >= current_score:
                self.current_state = best_state
                current_score = best_score if
                current_score >= self.goal_score:
                    print("\nFinal State:")
                    for block, resting_place in self.current_state.items():
                        print(f"Block {block} is on {resting_place}")
                    return self.current_state
            else:
                print("No better move found.")

            return self.current_state iterations

            += 1

        print("Maximum iterations reached.")
        return self.current_state


# Example usage:
initial_state = {'A': 'B', 'B': 'C', 'C': 'C'}
goal_score = 3

block_world_problem = BlockWorldProblem(initial_state, goal_score)
solution = block_world_problem.hill_climbing()
```

Output: