SIMULATED ANNEALING

IMPLEMENTATION

```
print("USN: 1BM22CS324")
print("V. DHANUSH REDDY")
import random
import math

# Function to calculate the number of conflicts in a given board configuration
def calculate_conflicts(board):
    conflicts = 0
    n = len(board)
    for i in range(n):
        for j in range(i + 1, n):
            # Check if two queens are attacking each other
            if board[i] == board[j] or abs(board[i] - board[j]) == j - i:
                conflicts += 1
    return conflicts

# Function to simulate the annealing process
def simulated_annealing(board, max_iterations=10000, initial_temperature=1000,
cooling_rate=0.99):
    current_board = board[:]  # Start with the user-provided initial board
    current_cost = calculate_conflicts(current_board)
    temperature = initial_temperature

    for iteration in range(max_iterations):
        if current_cost == 0:  # Found a solution
            return current_board, iteration
```

```
        # Generate a new board by making a small random change (i.e., move one queen)
        new_board = current_board[:]
        queen_index = random.randint(0, len(board) - 1)
        new_board[queen_index] = random.randint(0, len(board) - 1)

        new_cost = calculate_conflicts(new_board)

        # Accept the new board with a certain probability
        if new_cost < current_cost or random.random() < math.exp((current_cost - new_cost) / temperature):
            current_board = new_board
            current_cost = new_cost

        # Print the current state of the board and the current cost
        print(f"Iteration {iteration}: Cost = {current_cost}, Temperature = {temperature:.2f}")
        print_board(current_board)
        print("\n")

        # Cool down the temperature
        temperature *= cooling_rate

        if temperature < 1e-3:  # Stop if the temperature is too low
            break

    return current_board, iteration

# Function to print the board
def print_board(board):
    n = len(board)
    for i in range(n):
```

```
        row = ['Q' if board[j] == i else '.' for j in range(n)]
        print(" ".join(row))

# Main function to get input and solve the N-Queens problem
def main():
    # Get the number of queens from the user
    n = int(input("Enter the number of queens: "))

    # Get initial positions from the user (0-indexed)
    print("Enter the initial positions of the queens as a list of row indices (0-indexed):")
    board = list(map(int, input().split()))

    if len(board) != n:
        print("Error: The number of positions provided does not match the number of queens.")
        return

    # Run the simulated annealing algorithm
    solution, iterations = simulated_annealing(board)

    # Print the solution
    print(f"\nSolution found in {iterations} iterations:")
    print_board(solution)

# Run the main function
main()
```

OUTPUT:

```
USN: 1BM22CS324
V. DHANUSH REDDY
Enter the number of queens: 4
Enter the initial positions of the queens as a list of row indices (0-indexed):
3 1 2 0
Iteration 0: Cost = 3, Temperature = 1000.00
. . . Q
. Q . .
Q . Q .
. . . .


Iteration 1: Cost = 3, Temperature = 990.00
. . . Q
. Q . .
Q . Q .
. . . .


Iteration 2: Cost = 2, Temperature = 980.10
. Q . Q
. . . .
Q . Q .
. . . .
```

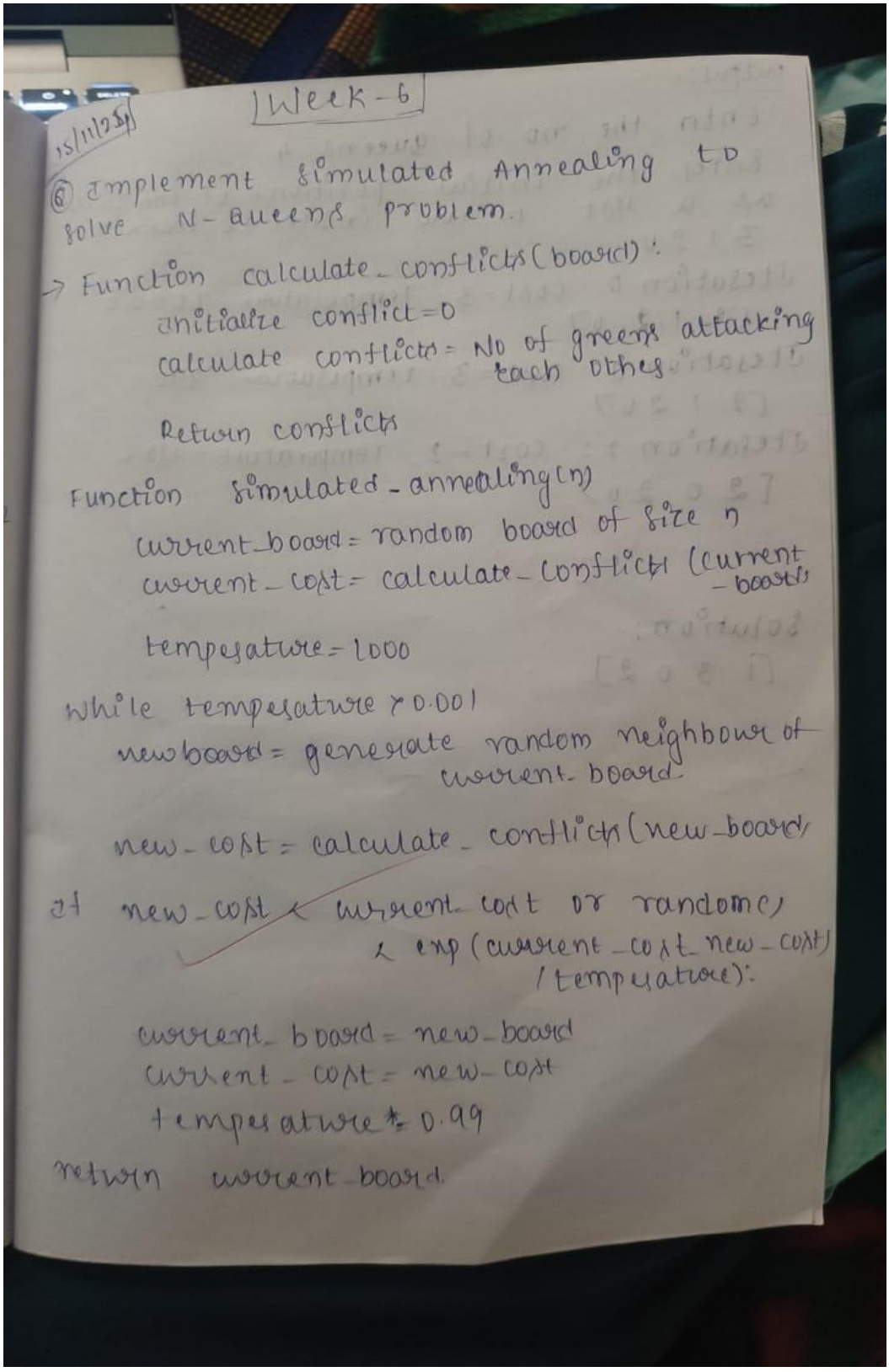
```
Iteration 152: Cost = 3, Temperature = 217.04
. Q . .
Q . . Q
. . . .
. . Q .


Iteration 153: Cost = 0, Temperature = 214.87
. Q . .
. . . Q
Q . . .
. . Q .



Solution found in 154 iterations:
. Q . .
. . . Q
Q . . .
. . Q .
```

OBSERVATION



Week-6

15/11/24

⑥ Implement simulated Annealing to solve N-Queens problem.

→ Function calculate_conflicts(board):
initialize conflict=0
calculate conflicts = No of queens attacking each other
Return conflicts

Function simulated_annealing(n)
current_board = random board of size n
current_cost = calculate_conflicts(current_board)
temperature = 1000
While temperature > 0.001
new board = generate random neighbour of current_board
new_cost = calculate_conflicts(new_board)
if new_cost < current_cost or random() < exp(current_cost-new_cost)/temperature):
current_board = new_board
current_cost = new_cost
temperature *= 0.99
return current_board

Output:-

Enter the no of queens: 4

Enter the initial position of the queen as a list of row indices (0-indexed)

3 1 2 0

Iteration 0: cost=3, Temperature = 1000.0

[2, 1, 2, 0]

Iteration 1: cost=3, temperature = 990.0

[2 1 2 0]

Iteration 2: cost=2, Temperature = 980.0

[2 0 2 0]

Solution:-

[1 3 0 2].

15/1/24