Practical No. 1

Title -: Implement DFS and BFS Algorithm. Use and Undirected Graph and develop a Recursive Algorithm for searching all the vertices of the graph or tree data structure.

Code -:

```
# Breadth First Search:

graph = {
  '1' : ['2','5'],
  '2' : ['3', '4'],
  '5' : ['6'],
  '3' : [],
  '4' : ['6'],
  '6' : []
}

visited = []
queue = []

def breadthFirstSearch(visited, graph, node):
  visited.append(node)
  queue.append(node)

  while queue:
    m = queue.pop(0)
    print (m, end = " ")

    for neighbour in graph[m]:
      if neighbour not in visited:
        visited.append(neighbour)
        queue.append(neighbour)

print("Breadth-First Search: ")
breadthFirstSearch(visited, graph, '1')
```

Output -:

```
Microsoft Windows [Version 10.0.26100.3775]
(c) Microsoft Corporation. All rights reserved.

D:\1 - Study\Images\Virtual Studio\GitHub\Lab-Practical-TE>"D:/1 - Study/Images/Virtual Studio/
Breadth-First Search:
1 2 5 3 4 6
D:\1 - Study\Images\Virtual Studio\GitHub\Lab-Practical-TE>
```

```
# Depth First Search:

graph = {
  '1' : ['2','5'],
  '2' : ['3', '4'],
  '5' : ['6'],
  '3' : [],
  '4' : ['6'],
  '6' : []
}

visited = set()

def depthFirstSearch(visited, graph, node):
    if node not in visited:
        print (node)
        visited.add(node)
        for neighbour in graph[node]:
            depthFirstSearch(visited, graph, neighbour)

print("Depth-First Search")
depthFirstSearch(visited, graph, '1')
```

Output -:

```
Microsoft Windows [Version 10.0.26100.3775]
(c) Microsoft Corporation. All rights reserved.

D:\1 - Study\Images\Virtual Studio\GitHub\Lab-Practical-TE>"D:/1 - Study/Images/Virtual
Depth-First Search
1
2
3
4
6
5

D:\1 - Study\Images\Virtual Studio\GitHub\Lab-Practical-TE>
```

Practical No. 2

Title -: Implement A* Algorithm for any game search problem

Code -:

```
import heapq

def a_star(grid, start, goal):
    def heuristic(a, b):

        return abs(a[0] - b[0]) + abs(a[1] - b[1])

    rows, cols = len(grid), len(grid[0])
    open_set = []
    heapq.heappush(open_set, (0, start))

    came_from = {}
    g_score = {start: 0}

    while open_set:
        _, current = heapq.heappop(open_set)

        if current == goal:

            path = []
            while current in came_from:
                path.append(current)
                current = came_from[current]
            path.append(start)
            return path[::-1]

        for dx, dy in [(-1,0), (1,0), (0,-1), (0,1)]:
            neighbor = (current[0] + dx, current[1] + dy)

            if 0 <= neighbor[0] < rows and 0 <= neighbor[1] < cols:
                if grid[neighbor[0]][neighbor[1]] == 1:
                    continue

                tentative_g = g_score[current] + 1
                if neighbor not in g_score or tentative_g < g_score[neighbor]:
                    g_score[neighbor] = tentative_g
                    f_score = tentative_g + heuristic(neighbor, goal)
```

```
            heapq.heappush(open_set, (f_score, neighbor))
            came_from[neighbor] = current

    return None

grid = [
    [0, 0, 0, 0],
    [1, 1, 0, 1],
    [0, 0, 0, 0],
    [0, 1, 1, 0],
]
start = (0, 0)
goal = (3, 3)

path = a_star(grid, start, goal)
print("Path:", path)
```

Output -:

```
Microsoft Windows [Version 10.0.26100.3775]
(c) Microsoft Corporation. All rights reserved.

D:\1 - Study\Images\Virtual Studio\GitHub\Lab-Practical-TE>"D:/1 - Study/Images/Virtual Studio/Python/python.exe"
Path: [(0, 0), (0, 1), (0, 2), (1, 2), (2, 2), (2, 3), (3, 3)]

D:\1 - Study\Images\Virtual Studio\GitHub\Lab-Practical-TE>
```

Practical No. 3

Title -: Implement Greedy search algorithm for Selection Sort.

Code -:

```
def selection_sort_greedy(arr):
    n = len(arr)
    print("\nList before Sorting: ", arr,"\n")
    for i in range(n):
        min_idx = i
        for j in range(i+1, n):
            if arr[j] < arr[min_idx]:
                min_idx = j
        arr[i], arr[min_idx] = arr[min_idx], arr[i]
        print("List After Pass ",i+1,": ",arr)
    return arr

n=int(input("Length of List: "))
arr=[]
for i in range(n):
    element=int(input("Enter List Element: "))
    arr.append(element)
print("\nSorted List is:", selection_sort_greedy(arr))
```

Output -:

```
Microsoft Windows [Version 10.0.26100.3775]
(c) Microsoft Corporation. All rights reserved.

D:\1 - Study\Images\Virtual Studio\GitHub\Lab-Practical-TE>"D:/1 - Study/Images/Virtual
.py"
Length of List: 5
Enter List Element: 11
Enter List Element: 33
Enter List Element: 55
Enter List Element: 77
Enter List Element: 99

List before Sorting:  [11, 33, 55, 77, 99]

List After Pass  1 :  [11, 33, 55, 77, 99]
List After Pass  2 :  [11, 33, 55, 77, 99]
List After Pass  3 :  [11, 33, 55, 77, 99]
List After Pass  4 :  [11, 33, 55, 77, 99]
List After Pass  5 :  [11, 33, 55, 77, 99]

Sorted List is: [11, 33, 55, 77, 99]

D:\1 - Study\Images\Virtual Studio\GitHub\Lab-Practical-TE>
```

Practical No. 4

Title -: Implement a solution for a Constraint Satisfaction Problem using Branch and Bound and Backtracking for n-queens problem or a graph coloring problem.

Code -:

```
def is_safe(queens, row, col):
    for r in range(row):
        c = queens[r]
        if c == col or abs(c - col) == abs(r - row):
            return False
    return True

def solve_n_queens(n):
    solutions = []

    def backtrack(row, queens):
        if row == n:
            solutions.append(queens[:])
            return
        for col in range(n):
            if is_safe(queens, row, col):
                queens[row] = col
                backtrack(row + 1, queens)

    backtrack(0, [-1] * n)
    return solutions

def print_board(solution):
    for row in solution:
        print(' '.join('Q' if i == row else '.' for i in range(len(solution))))
    print()

n = 8
all_solutions = solve_n_queens(n)
print(f"Total Solutions for {n}-Queens: {len(all_solutions)}\n")
print("Sample Solution:")
print_board(all_solutions[0])
```

Output -:

```
Microsoft Windows [Version 10.0.26100.3775]
(c) Microsoft Corporation. All rights reserved.

D:\1 - Study\Images\Virtual Studio\GitHub\Lab-Practical-TE>"D:/1 - Study/Images/Virtual Studio/Python/python.exe"
Total Solutions for 8-Queens: 92

Sample Solution:
Q . . . . . . .
. . . . Q . . .
. . . . . . . Q
. . . . . Q . .
. . Q . . . . .
. . . . . . Q .
. Q . . . . . .
. . . Q . . . .


D:\1 - Study\Images\Virtual Studio\GitHub\Lab-Practical-TE>
```

Practical No. 5

Title -: Develop an elementary chatbot for any suitable customer interaction application.

Code -:

```
def pizza_bot():
    print("🍕 Welcome to PizzaBot!")
    print("How can I help you today? (Type 'quit' to exit)\n")

    while True:
        user_input = input("You: ").lower()

        if "quit" in user_input or "bye" in user_input:
            print("PizzaBot: Thanks for chatting! Have a cheesy day!")
            break

        elif "menu" in user_input or "show" in user_input:
            print("PizzaBot: Here's our menu:\n- Margherita\n- Pepperoni\n- Veggie\n- BBQ
Chicken")

        elif "order" in user_input or "want" in user_input:
            print("PizzaBot: Great! What pizza would you like to order?")

        elif "margherita" in user_input:
            print("PizzaBot: Margherita pizza added to your order!")

        elif "pepperoni" in user_input:
            print("PizzaBot: Pepperoni pizza added to your order!")

        elif "veggie" in user_input:
            print("PizzaBot: Veggie pizza added to your order!")

        elif "bbq" in user_input or "chicken" in user_input:
            print("PizzaBot: BBQ Chicken pizza added to your order!")

        elif "price" in user_input or "cost" in user_input:
            print("PizzaBot: All pizzas are ₹299 each.")

        elif "thanks" in user_input or "thank you" in user_input:
            print("PizzaBot: You're welcome! 😊 ")
```

```
    else:
        print("PizzaBot: Sorry, I didn’t understand that. You can ask about our menu, prices, or place an order.")

pizza_bot()
```

Output -:

```
(c) Microsoft Corporation. All rights reserved.

D:\1 - Study\Images\Virtual Studio\GitHub\Lab-Practical-TE>"D:/1 - Study/Images/Virtual Studio/Python/python.exe"
🍕 Welcome to PizzaBot!
How can I help you today? (Type 'quit' to exit)

You: hii
PizzaBot: Sorry, I didn’t understand that. You can ask about our menu, prices, or place an order.
You: menu
PizzaBot: Here's our menu:
- Margherita
- Pepperoni
- Veggie
- BBQ Chicken
You: Pepperoni
PizzaBot: Pepperoni pizza added to your order!
You: price
PizzaBot: All pizzas are ₹299 each.
You: ok
PizzaBot: Sorry, I didn’t understand that. You can ask about our menu, prices, or place an order.
You: thanks
PizzaBot: You're welcome! 😊
You:
```

Practical No. 6

Title -: Implement any one of the following Expert System

I. Information management
II. Hospitals and medical facilities
III. Help desks management
IV. Employee performance evaluation
V. Stock market trading
VI. Airline scheduling and cargo schedules.

Code -:

```
def evaluate_performance(attendance, projects_completed, teamwork_score):
    # Rule-based evaluation
    if attendance >= 90 and projects_completed >= 5 and teamwork_score >= 8:
        return "Excellent"
    elif attendance >= 80 and projects_completed >= 3 and teamwork_score >= 6:
        return "Good"
    elif attendance >= 70 and projects_completed >= 2 and teamwork_score >= 5:
        return "Average"
    else:
        return "Needs Improvement"

print("Employee Performance Evaluation System")

attendance = int(input("Enter Attendance %: "))
projects = int(input("Enter Number of Projects Completed: "))
teamwork = int(input("Enter Teamwork Score (1-10): "))

result = evaluate_performance(attendance, projects, teamwork)
print("\nEmployee Evaluation:", result)
```

Output -:

```
Microsoft Windows [Version 10.0.26100.3775]
(c) Microsoft Corporation. All rights reserved.

D:\1 - Study\Images\Virtual Studio\GitHub\Lab-Practical-TE>"D:/1 - Study/Images/Virtual Stud
.py"
Employee Performance Evaluation System
Enter Attendance %: 88
Enter Number of Projects Completed: 2
Enter Teamwork Score (1-10): 4

Employee Evaluation: Needs Improvement

D:\1 - Study\Images\Virtual Studio\GitHub\Lab-Practical-TE>
```