Name: Devanshi Jain
Er-No: 22162101006
Batch: 51
Branch: CBA

Institute of Computer Technology
B. Tech Computer Science and Engineering

Sub: Algorithm Analysis and Design

Practical 4

Trigent is an early pioneer in IT outsourcing and offshore software development business. Thousands of employees working in this company kindly help to find out the employee's details (i.e employee ID, employee salary etc) to implement Recursive Binary search and Linear search (or Sequential Search) and determine the time taken to search an element. Repeat the experiment for different values of n, the number of elements in the list to be searched and plot a graph of the time taken versus n.
Design the algorithm for the same and implement using the programming language of your choice. Make comparative analysis for various use cases & input size.

Using the algorithm search for the following

1. The designation which has highest salary package
2. The Name of the Employee who has the lowest salary
3. The Mobile number who is youngest employee
4. Salary of the employee who is oldest in age

CODE:

```
from flask import Flask, render_template, request
import matplotlib.pyplot as plt
import io
import base64

app = Flask(__name__)

class Employee:
    def __init__(self, id, name, designation, salary, age, mobile_number):
        self.id = id
        self.name = name
        self.designation = designation
        self.salary = salary
        self.age = age
        self.mobile_number = mobile_number

employees = [
    {'employeeID': 'E001', 'name': 'Amit Jain', 'age': 52, 'salary': 90000,
'designation': 'Project Manager', 'mobile_number': '1234456789', 'email':
'amit.jain@gmail.com', 'department': 'Management', 'address': 'MG Road, Delhi'},
    {'employeeID': 'E002', 'name': 'Rosy Singh', 'age': 25, 'salary': 60000,
'designation': 'Software Developer', 'mobile_number': '1122334455', 'email':
'rosy.singh@gmail.com', 'department': 'IT', 'address': 'LBS Marg, Mumbai'},
    {'employeeID': 'E003', 'name': 'Hitesh Patel', 'age': 28, 'salary': 70000,
'designation': 'Finance', 'mobile_number': '6677889900', 'email':
'hitesh.patel@gmail.com', 'department': 'Finance', 'address': 'CG
road,Ahmedabad'},
    {'employeeID': 'E004', 'name': 'Sapna Gupta', 'age': 55, 'salary': 80000,
'designation': 'Assistant', 'mobile_number': '0978563421', 'email':
'sapna.gupta@gmail.com', 'department': 'IT', 'address': 'Shastrinagar,Jaipur'},
    {'employeeID': 'E005', 'name': 'Paras Sen', 'age': 35, 'salary': 85000,
'designation': 'Physical Fitness', 'mobile_number': '6754382798', 'email':
'paras.sen@gmail.com', 'department': 'Fitness', 'address': 'Jamaica
pura.odisha'},
]

def linear_search(arr, target):
    count = 0
    for i in range(len(arr)):
        count += 1
        if arr[i] == target:
            count += 1
            return count
    return count
```

```python
def binary_search(arr, target):
    count = 0
    start = 0
    end = len(arr) - 1
    while start <= end:
        count += 1
        mid = (start + end) // 2
        if arr[mid] == target:
            count += 1
            return count
        elif arr[mid] < target:
            count += 1
            start = mid + 1
        else:
            count += 1
            end = mid - 1
    return count

def find_highest_salary_designation(employees):
    if not employees:
        return None
    highest_salary_employee = max(employees, key=lambda emp: emp['salary'])
    return highest_salary_employee['designation']

def find_lowest_salary_employee(employees):
    if not employees:
        return None
    lowest_salary_employee = min(employees, key=lambda emp: emp['salary'])
    return lowest_salary_employee['name']

def find_youngest_employee_mobile(employees):
    if not employees:
        return None
    youngest_employee = min(employees, key=lambda emp: emp['age'])
    return youngest_employee['mobile_number']

def find_oldest_employee_salary(employees):
    if not employees:
        return None
    oldest_employee = max(employees, key=lambda emp: emp['age'])
    return oldest_employee['salary']

@app.route('/')
def home():
```

```
    return render_template('p4.html', results={})

@app.route('/submit', methods=['POST'])
def submit():
    numbers = list(map(int, request.form['numbers'].split(',')))
    linear_search_counts = []
    binary_search_counts = []
    for num in numbers:
        number_list = list(range(1, num + 1))
        linear_search_counts.append(linear_search(number_list.copy(), num))
        binary_search_counts.append(binary_search(number_list.copy(), num))

    # Plotting the graph
    plt.figure()
    plt.plot(numbers, linear_search_counts, label='Linear Search')
    plt.plot(numbers, binary_search_counts, label='Binary Search')
    plt.xlabel('Number of Elements')
    plt.ylabel('Operations Count')
    plt.title('Time Complexity Analysis of Linear and Binary Search')
    plt.legend()

    # Save the plot to a bytes buffer
    buf = io.BytesIO()
    plt.savefig(buf, format='png')
    buf.seek(0)
    graph_url = base64.b64encode(buf.getvalue()).decode('utf-8')
    buf.close()

    # Find employee details
    highest_salary_designation = find_highest_salary_designation(employees)
    lowest_salary_employee_name = find_lowest_salary_employee(employees)
    youngest_employee_mobile = find_youngest_employee_mobile(employees)
    oldest_employee_salary = find_oldest_employee_salary(employees)

    results = {
        'numbers': numbers,
        'linear_search_counts': linear_search_counts,
        'binary_search_counts': binary_search_counts,
        'employees': employees,
        'highest_salary_designation': highest_salary_designation,
        'lowest_salary_employee_name': lowest_salary_employee_name,
        'youngest_employee_mobile': youngest_employee_mobile,
        'oldest_employee_salary': oldest_employee_salary
    }
```

```
    return render_template('p4.html', results=results, graph_url=graph_url)

if __name__ == '__main__':
    app.run(debug=True)
```

```
<!DOCTYPE html>
<html lang="en">
<head>
    <meta charset="UTF-8">
    <meta name="viewport" content="width=device-width, initial-scale=1.0">
    <title>Search Algorithm Comparison</title>
    <link rel="stylesheet" href="{{ url_for('static', filename='styles.css') }}">
</head>
<body>
    <div class="container">
        <h1>Search Algorithm Comparison</h1>
        <form action="/submit" method="post">
            <label for="numbers">Enter Numbers (comma-separated):</label>
            <input type="text" id="numbers" name="numbers" required>
            <button type="submit">Submit</button>
        </form>

        {% if results %}
            <div class="results">
                <h2>Results</h2>
                <p><strong>Highest Salary Designation:</strong> {{
results.highest_salary_designation }}</p>
                <p><strong>Lowest Salary Employee Name:</strong> {{
results.lowest_salary_employee_name }}</p>
                <p><strong>Youngest Employee Mobile:</strong> {{
results.youngest_employee_mobile }}</p>
                <p><strong>Oldest Employee Salary:</strong> {{
results.oldest_employee_salary }}</p>

                <h3>Graph of Search Algorithm Operations</h3>
                <img src="{{ graph_url }}" alt="Search
Algorithm Graph">

                <h3>Employee List</h3>
                <table>
                    <thead>
                        <tr>
```

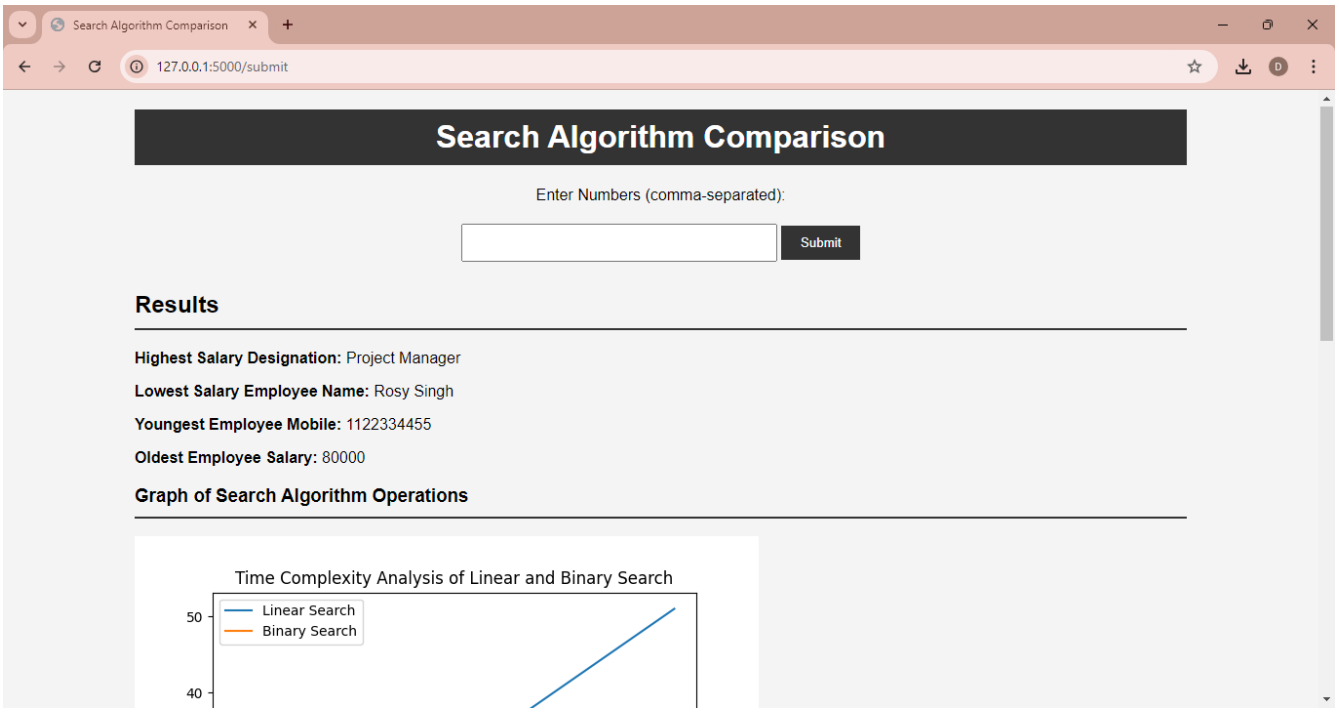
```
                        <th>ID</th>
                        <th>Name</th>
                        <th>Designation</th>
                        <th>Salary</th>
                        <th>Age</th>
                        <th>Mobile Number</th>
                    </tr>
                </thead>
                <tbody>
                    {% for employee in results.employees %}
                        <tr>
                            <td>{{ employee.employeeID }}</td>
                            <td>{{ employee.name }}</td>
                            <td>{{ employee.designation }}</td>
                            <td>{{ employee.salary }}</td>
                            <td>{{ employee.age }}</td>
                            <td>{{ employee.mobile_number }}</td>
                        </tr>
                    {% endfor %}
                </tbody>
            </table>

            <h3>Operations Count</h3>
            <table>
                <thead>
                    <tr>
                        <th>Number of Elements</th>
                        <th>Linear Search Count</th>
                        <th>Binary Search Count</th>
                    </tr>
                </thead>
                <tbody>
                    {% for i in range(results.numbers|length) %}
                        <tr>
                            <td>{{ results.numbers[i] }}</td>
                            <td>{{ results.linear_search_counts[i] }}</td>
                            <td>{{ results.binary_search_counts[i] }}</td>
                        </tr>
                    {% endfor %}
                </tbody>
            </table>
        </div>
    {% endif %}
</div>
</body>
```

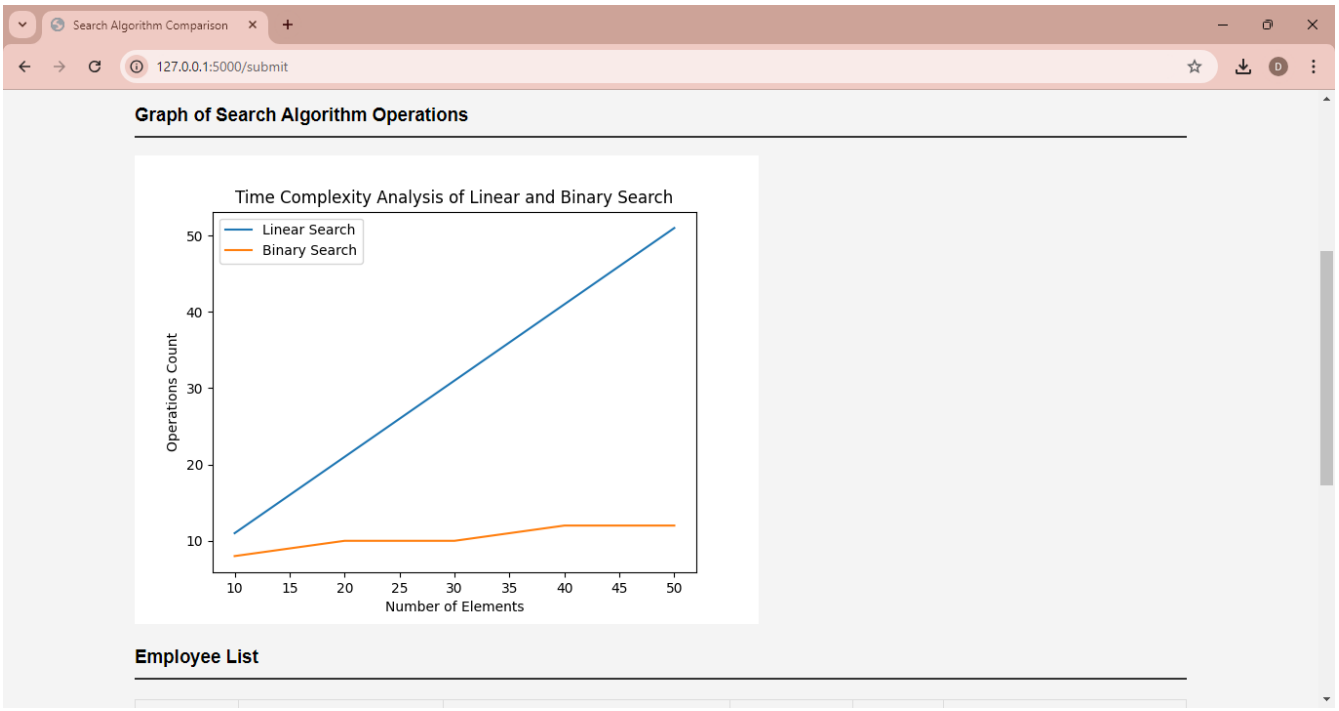
```
</html>
```

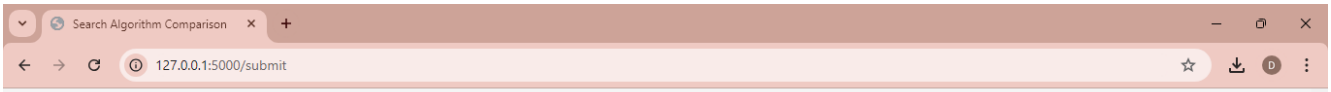
```
body {
    font-family: Arial, sans-serif;
    margin: 0;
    padding: 0;
    background-color: #f4f4f4;
}

.container {
    width: 80%;
    margin: auto;
    overflow: hidden;
}

h1 {
    background: #333;
    color: #fff;
    padding: 10px 0;
    text-align: center;
}

form {
    margin: 20px 0;
    text-align: center;
}

label {
    display: block;
    margin: 10px 0;
}

input[type="text"] {
    padding: 10px;
    width: 300px;
    margin: 10px 0;
}

button {
    padding: 10px 20px;
    background: #333;
    color: #fff;
```

```
    border: none;
    cursor: pointer;
}

button:hover {
    background: #555;
}

.results {
    margin: 20px 0;
}

.results h2, .results h3 {
    border-bottom: 2px solid #333;
    padding-bottom: 10px;
}

.results table {
    width: 100%;
    border-collapse: collapse;
    margin: 20px 0;
}

.results table, th, td {
    border: 1px solid #ddd;
}

.results th, td {
    padding: 10px;
    text-align: center;
}

.results th {
    background-color: #f4f4f4;
}
```

OUTPUT:


Search Algorithm Comparison
127.0.0.1:5000/submit
Search Algorithm Comparison
Enter Numbers (comma-separated):
Submit
Results
Highest Salary Designation: Project Manager
Lowest Salary Employee Name: Rosy Singh
Youngest Employee Mobile: 1122334455
Oldest Employee Salary: 80000
Graph of Search Algorithm Operations
Time Complexity Analysis of Linear and Binary Search
Linear Search
Binary Search
50
40


Search Algorithm Comparison
127.0.0.1:5000/submit
Graph of Search Algorithm Operations
Time Complexity Analysis of Linear and Binary Search
Linear Search
Binary Search
Operations Count
50
40
30
20
10
10
15
20
25
30
35
40
45
50
Number of Elements
Employee List



Employee List

ID	Name	Designation	Salary	Age	Mobile Number
E001	Amit Jain	Project Manager	90000	52	1234456789
E002	Rosy Singh	Software Developer	60000	25	1122334455
E003	Hitesh Patel	Finance	70000	28	6677889900
E004	Sapna Gupta	Assistant	80000	55	0978563421
E005	Paras Sen	Physical Fitness	85000	35	6754382798

Operations Count

Number of Elements	Linear Search Count	Binary Search Count
10	11	8
20	21	10
30	31	10
40	41	12
50	51	12