Name: Ayush Patel   Class B   Batch 55 Enrolment: 22162171038

Institute of Computer Technology
B. Tech Computer Science and Engineering

Sub: Algorithm Analysis and Design

Practical 2

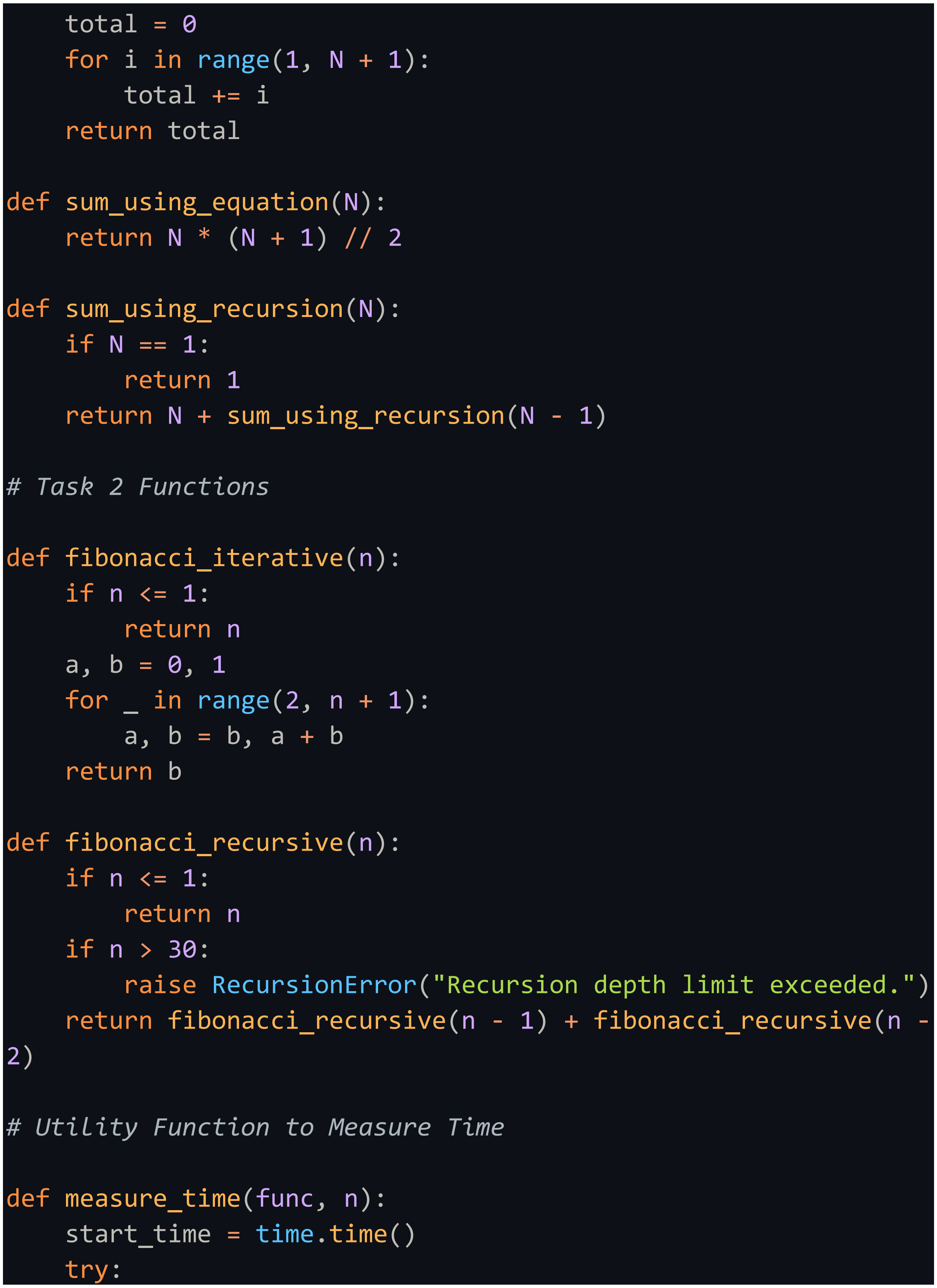
(1) MPSoft Technologies Pvt. Ltd. is a fast growing IT industry and wants to implement a function to calculate the monthly income generated from all projects from their N no of clients like C1,C2,C3,C4….CN. The team wants to compare the time/steps required to execute this function on various inputs and analyse the complexity of each combination. Also draw a comparative chart. In each of the following functions N will be passed by user.
Design the algorithm for the same and implement using the programming language of your choice. Make comparative analysis for various use cases & input size.

1. To calculate the sum of 1 to N number using loop.
2. To calculate the sum of 1 to N number using the equation.
3. To calculate sum of 1 to N numbers using recursion

Code:

App.py: (Combine code of task 1 and 2)

```
from flask import Flask, render_template, request, redirect,
url_for, Response
import time
import matplotlib.pyplot as plt
import io
from matplotlib.backends.backend_agg import FigureCanvasAgg as
FigureCanvas
import sys

app = Flask(__name__)
sys.setrecursionlimit(1000000)

# Task 1 Functions

def sum_using_loop(N):
```



```
    total = 0
    for i in range(1, N + 1):
        total += i
    return total

def sum_using_equation(N):
    return N * (N + 1) // 2

def sum_using_recursion(N):
    if N == 1:
        return 1
    return N + sum_using_recursion(N - 1)

# Task 2 Functions

def fibonacci_iterative(n):
    if n <= 1:
        return n
    a, b = 0, 1
    for _ in range(2, n + 1):
        a, b = b, a + b
    return b

def fibonacci_recursive(n):
    if n <= 1:
        return n
    if n > 30:
        raise RecursionError("Recursion depth limit exceeded.")
    return fibonacci_recursive(n - 1) + fibonacci_recursive(n -
2)

# Utility Function to Measure Time

def measure_time(func, n):
    start_time = time.time()
    try:
```

Name: Ayush Patel   Class B   Batch 55 Enrolment: 22162171038

```
        result = func(n)
        if isinstance(result, int):
            end_time = time.time()
            return end_time - start_time, result
        else:
            return float('inf'), None
    except RecursionError:
        return float('inf'), None

# Main Route

@app.route("/", methods=["GET", "POST"])
def main():
    # Task 1 - Sum of Numbers
    if request.method == "POST":
        if 'generate' in request.form:
            return redirect(url_for('task1'))

        # Task 1 Processing
        if 'N' in request.form:
            try:
                N = int(request.form["N"])
                if N < 0:
                    return render_template("task1.html",
error="Please enter a non-negative integer.", N=None)

                loop_time, loop_result =
measure_time(sum_using_loop, N)
                equation_time, equation_result =
measure_time(sum_using_equation, N)
                recursion_time, recursion_result =
measure_time(sum_using_recursion, N)

                recursion_data = (recursion_result,
recursion_time) if recursion_result is not None else ("Skipped
due to depth limit", None)
```

```
                return render_template(
                    "task1.html",
                    N=N,
                    loop_data=(loop_result, loop_time),
                    equation_data=(equation_result,
equation_time),
                    recursion_data=recursion_data,
                    error=None
                )
            except ValueError:
                return render_template("index.html",
error="Invalid input. Please enter an integer.", N=None)

        # Task 2 Processing
        elif 'n' in request.form:
            try:
                n = int(request.form["n"])
                if n < 0:
                    return render_template("task2.html",
error="Please enter a non-negative integer.", n=None)

                iterative_time, iterative_result =
measure_time(fibonacci_iterative, n)
                recursive_time, recursive_result =
measure_time(fibonacci_recursive, n)

                iterative_data = (iterative_result,
iterative_time)
                if recursive_result is None:
                    recursive_data = (None, float('inf'))
                else:
                    recursive_data = (recursive_result,
recursive_time)

                return render_template(
                    "task2.html",
                    n=n,
                    iterative_data=iterative_data,
                    recursive_data=recursive_data,
                    error=None
                )
            except ValueError:
                return render_template("index.html",
error="Invalid input. Please enter an integer.", n=None)

    return render_template("index.html", N=None, error=None)

# Task 1 Route

@app.route('/task1', methods=["GET", "POST"])
def task1():
    if request.method == "POST":
        return render_template('task1.html')
    return render_template('task1.html')

@app.route("/practical_1.png")
def practical_1():
    input_sizes = [100, 1000, 5000, 10000, 20000, 50000,
100000]
    loop_times = []
    equation_times = []
    recursion_times = []

    for size in input_sizes:
        loop_times.append(measure_time(sum_using_loop,
size)[0])
        equation_times.append(measure_time(sum_using_equation,
size)[0])
        recursion_times.append(measure_time(sum_using_recursion
, size)[0])
```

```
    fig, ax = plt.subplots(figsize=(12, 6))
    ax.plot(input_sizes, loop_times, label='Loop', marker='o')
    ax.plot(input_sizes, equation_times, label='Equation',
marker='o')
    ax.plot(input_sizes, recursion_times, label='Recursion',
marker='o')
    ax.set_xlabel('Input Size (N)')
    ax.set_ylabel('Execution Time (seconds)')
    ax.set_title('Comparison of Execution Time for Sum of 1 to
N')
    ax.legend()
    ax.grid(True)

    output = io.BytesIO()
    FigureCanvas(fig).print_png(output)
    return Response(output.getvalue(), mimetype='image/png')

# Task 2 Route

@app.route('/task2', methods=["GET", "POST"])
def task2():
    iterative_data = None
    recursive_data = None
    n = None
    error = None
    if request.method == "POST":
        try:
            n = int(request.form["n"])
            if n < 0:
                error = "Please enter a non-negative integer."
            else:
                iterative_time, iterative_result =
measure_time(fibonacci_iterative, n)
                recursive_time, recursive_result =
measure_time(fibonacci_recursive, n)
```

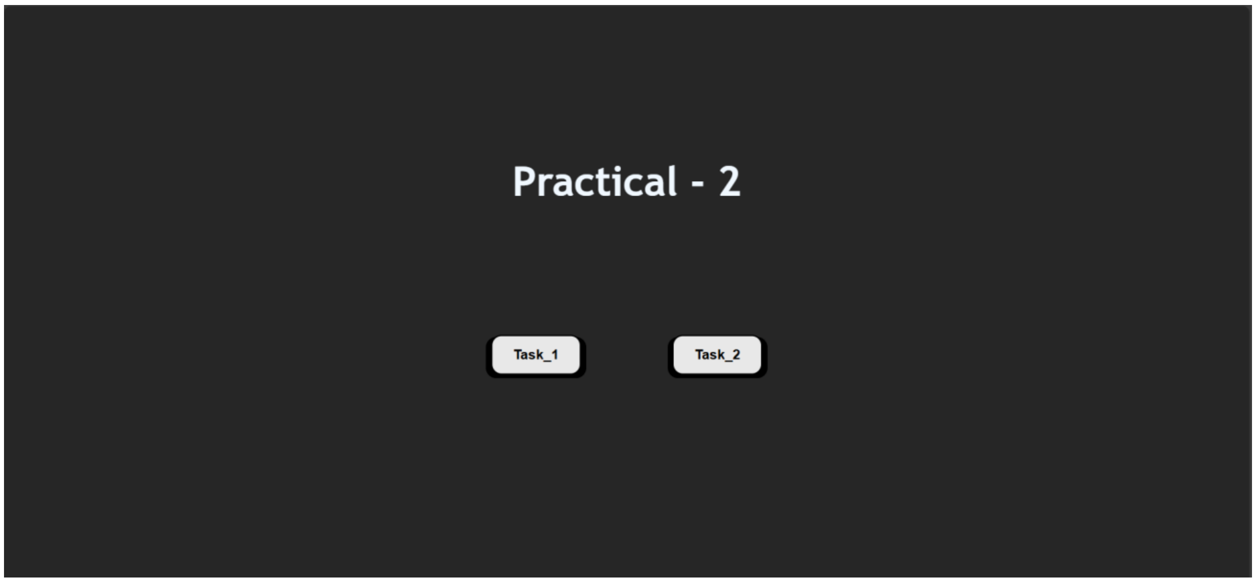
```
                iterative_data = (iterative_result,
iterative_time)
                if recursive_result is None:
                    recursive_data = (None, float('inf'))
                else:
                    recursive_data = (recursive_result,
recursive_time)
        except ValueError:
            error = "Invalid input. Please enter an integer."

    return render_template(
        "task2.html",
        n=n,
        iterative_data=iterative_data,
        recursive_data=recursive_data,
        error=error
    )

@app.route("/practical_2.png")
def practical_2():
    input_sizes = [5, 10, 15, 20, 25, 30, 35]
    iterative_times = []
    recursive_times = []

    for size in input_sizes:
        iterative_times.append(measure_time(fibonacci_iterative
, size)[0])
        recursive_times.append(measure_time(fibonacci_recursive
, size)[0])

    fig, ax = plt.subplots(figsize=(12, 6))
    ax.plot(input_sizes, iterative_times, label='Iterative',
marker='o')
    ax.plot(input_sizes, recursive_times, label='Recursive',
marker='o')
    ax.set_xlabel('Input Size (n)')
    ax.set_ylabel('Execution Time (seconds)')
    ax.set_title('Comparison of Execution Time for Fibonacci
Calculation')
    ax.legend()
    ax.grid(True)

    output = io.BytesIO()
    FigureCanvas(fig).print_png(output)
    return Response(output.getvalue(), mimetype='image/png')

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

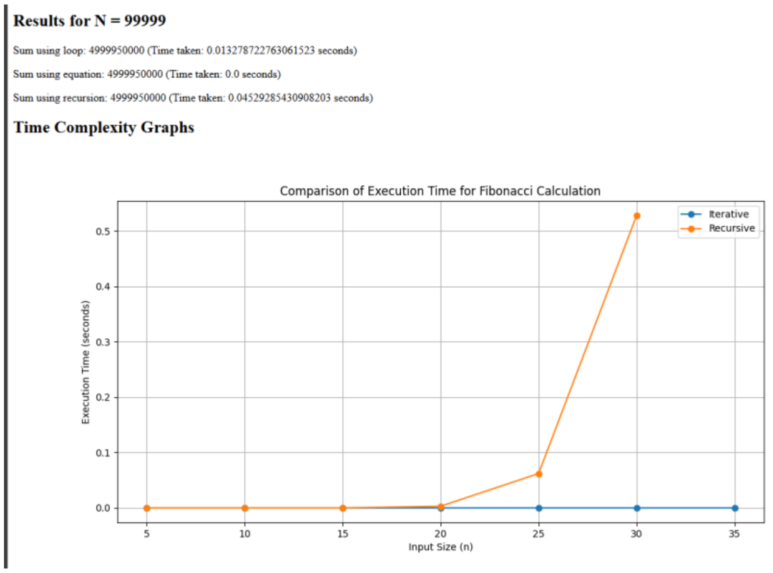
Task1.html:

```
<!DOCTYPE html>
<html lang="en">
<head>
    <meta charset="UTF-8">
    <meta name="viewport" content="width=device-width, initial-
scale=1.0">
    <title>Sum Calculation Methods</title>
</head>
<body>
    <h1>Sum Calculation Methods</h1>
    <form method="post" action="/">
        <label for="N">Enter a value for N:</label>
        <input type="text" id="N" name="N" required>
        <button type="submit">Calculate</button>
    </form>

    {% if error %}
        <p style="color:red;">{{ error }}</p>
    {% endif %}
    {% if N is not none %}
        <h2>Results for N = {{ N }}</h2>
        {% if loop_data %}
            <p>Sum using loop: {{ loop_data[0] }} (Time taken: {{ loop_data[1] }} seconds)</p>
        {% endif %}
        {% if equation_data %}
            <p>Sum using equation: {{ equation_data[0] }} (Time taken: {{ equation_data[1] }} seconds)</p>
        {% endif %}
        {% if recursion_data %}
            <p>Sum using recursion: {{ recursion_data[0] }}
            {% if recursion_data[1] is not none %}
                (Time taken: {{ recursion_data[1] }} seconds)
            {% else %}
                (Skipped due to recursion limit)
            {% endif %}
            </p>
        {% endif %}
    {% endif %}

    <h2>Time Complexity Graphs</h2>
    <img src="{{ url_for('practical_2') }}" alt="Time Complexity Graphs">
</body>
</html>
```

Output:

Name: Ayush Patel   Class B   Batch 55 Enrolment: 22162171038





```
Results for N = 99999

Sum using loop: 4999950000 (Time taken: 0.013278722763061523 seconds)

Sum using equation: 4999950000 (Time taken: 0.0 seconds)

Sum using recursion: 4999950000 (Time taken: 0.04529285430908203 seconds)

Time Complexity Graphs
```

(2) Suppose a newly-born pair of rabbits, one male, one female, are put in a field. Rabbits are able to mate at the age of one month so that at the end of its second month a female can produce another pair of rabbits. Suppose that our rabbits never die and that the female always produces one new pair (one male, one female) every month from the second month on. How many pairs will there be in one year? Apply appropriate algorithm/method to find out the above problem and also solve them using iteration and recursive method. Compare the performance of two methods by counting the number of steps executed on various inputs. Also draw a comparative chart.

Design the algorithm for the same and implement using the programming language of your choice. Make comparative analysis for various use cases & input size.

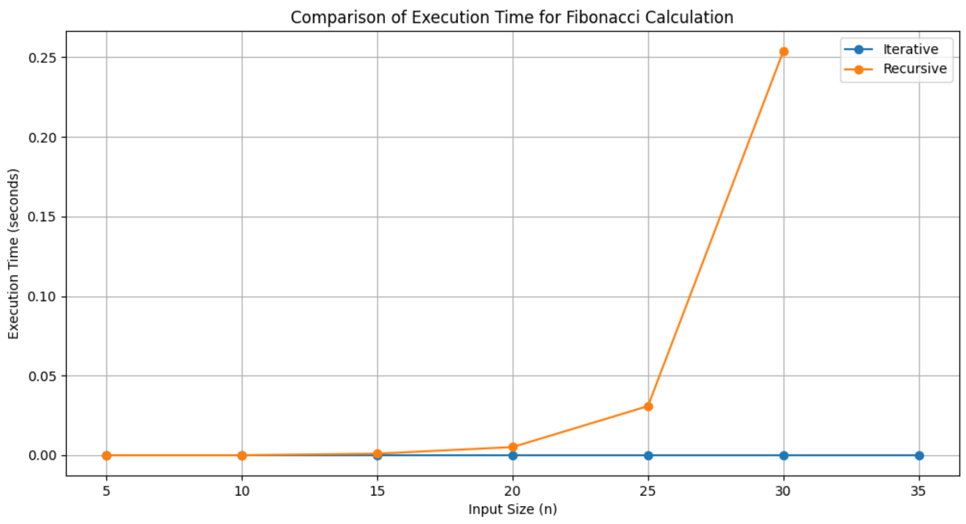
Name: Ayush Patel   Class B   Batch 55 Enrolment: 22162171038

Task2.html:

```html
<!DOCTYPE html>
<html lang="en">
  <head>
    <meta charset="UTF-8" />
    <meta name="viewport" content="width=device-width, initial-
scale=1.0"/>
    <title>Fibonacci Calculation</title>
  </head>
  <body>
    <h1>Fibonacci Calculation</h1>
    <form method="post">
      <label for="n">Enter a value for n:</label>
      <input type="number" id="n" name="n" required />
      <button type="submit">Calculate</button>
    </form>
    <h2>Results for n = {{ n }}:</h2>
    <p>
      Iterative Fibonacci Result: {{ iterative_data[0] }} (Time
taken: {{
      iterative_data[1] }} seconds)
    </p>
    <p>
      Recursive Fibonacci Result: {{ recursive_data[0] }} (Time
taken: {{
      recursive_data[1] }} seconds)
    </p>
    <p>Recursive calculation took too long or exceeded
limits.</p>
    <p style="color: red">{{ error }}</p>
    <h2>Execution Time Comparison</h2>
    <img
      src="{{ url_for('practical_2') }}"
      alt="Execution Time
 Comparison"
    />
  </body>
</html>
```

Output:

```
Results for n = 30:

Iterative Fibonacci Result: 832040 (Time taken: 0.0 seconds)

Recursive Fibonacci Result: 832040 (Time taken: 0.3681516647338867 seconds)

Recursive calculation took too long or exceeded limits.

None

Execution Time Comparison
```


Comparison of Execution Time for Fibonacci Calculation
Iterative
Recursive
Execution Time (seconds)
Input Size (n)