Algorithms Ass1

Jana Soghier 202210337

DR.THAER

Practice Assignment #1 (Empirical analysis of Factorial Calculation)

THE ARAB AMERICAN UNIVERSITY
FACULTY OF ENGINEERING AND IT
Tel .970-4 – 2510801/1-5(Ext.1234)
Fax : 970-4-2510886

الجامعة العربية الأمريكية
كلية الهندسة وتكنولوجيا المعلومات
970-4-2510801/1-5 (Ext.1234) ☎
فاكس : 970-4-2510886

Algorithms Analysis and Design (230213150)
Fall 2024/2025

Practice Assignment #1 (Empirical analysis of Factorial Calculation)

1. Due date: Saturday, 30/11/2025 (***No Assignments will be accepted after the mentioned date, with no exceptions.***)
2. The assignment is an ***individual*** effort. Copying the assignment will be treated as a cheating attempt, which may lead to *FAILING* the course.
3. Try your best to be precise and organized in presenting/analyzing results.
4. The report must be submitted in ***pdf*** format. You need to include all your ***code*** as well.

This assignment focuses on factorial calculation, a fundamental mathematical operation that can be solved using various methods. We will explore two such methods: iterative and recursive. By implementing and comparing both approaches, you will gain insights into their efficiency and limitations. This exercise provides an opportunity to measure execution times for different input values, observe how large values affect the stack, and learn how to handle large results by selecting appropriate data types. Through this assignment, you'll strengthen your understanding of programming and algorithmic practices.

Instructions:

- Implement an **iterative** function to calculate the factorial of a given integer ***n***.
- Implement a **recursive** function to calculate the factorial of **n**.
- Test both functions with various values of **n**, starting from small values and gradually increasing to larger values.
- Measure and record the execution time for each test case. You can use built-in timing libraries or functions available in your chosen programming language.
- Choose a sufficiently large value of **n** for both functions.
- Attempt to calculate the factorial for large values of **n** to observe the stack overflow issue. Record the outcome.
- ***Note***: Use appropriate data types to handle large results like (**long long**)

Deliverables:

1. Source Code: Upload the complete source code to the designated GitHub repository.

2. Report

Write a technical report about the experiments you have made (maximum of 2 pages). The report should contain the following:

1. Show execution time results in a presentable way (using ***tables*** and ***charts***).
2. Provide stack overflow observations.
3. You should ***discuss*** what you have found.
4. Conclude the work as a whole.

All the best

The code

```cpp
#include <iostream>
using namespace std;
 long long iterative(int n) {
    long long fact = 1;
    for (int j = 1; j <= n; ++j) {
        fact *= j;}
    std::cout << fact << std::endl;
    return fact;
}
long long recursive(int n) {
    if (n == 0)
        return 1;
   else {
        return n * recursive(n - 1);}

}

int main()
{
    int n;
   cin>>n;
    iterative(n);
    recursive(n);
}
```

Measure and record the execution time for each test case

Iterative

N	*5*	*15*	*70*	*100*	*500*	*1000*	*10000*	*10^7*	*10^10*
Time(ns)	149	218	393	492	1308	2771	27558	26633018	.....

```cpp
#include <iostream>
#include <chrono>

using namespace std;

long long iterative(int n) {
    long long fact = 1;
    for (int j = 1; j <= n; ++j) {
        fact *= j;
    }
    return fact;
}

int main() {
    // Measure the execution time
    auto start = chrono::high_resolution_clock::now();

    iterative(10000000);

    auto end = chrono::high_resolution_clock::now();
    auto duration = chrono::duration_cast<chrono::nanoseconds>(end - start);

    cout << "Execution time: " << duration.count() << " nanoseconds" << endl;

    return 0;
}
```

Recursive

N	5	15	70	100	500	1000	10000	10^10
Time(ns)	153	242	784	1414	13564	24870	291999	.....

```cpp
#include <iostream>
#include <chrono>

using namespace std;

long long recursive(int n) {
    if (n == 0)
        return 1;
    else
        return n * recursive(n - 1);
}

int main() {
    // Measure the execution time
    auto start = chrono::high_resolution_clock::now();

    recursive(10000000);

    auto end = chrono::high_resolution_clock::now();
    auto duration = chrono::duration_cast<chrono::nanoseconds>(end - start);

    cout << "Execution time: " << duration.count() << " nanoseconds" << endl;

    return 0;
}
```

input

```
..Program finished with exit code 139
```

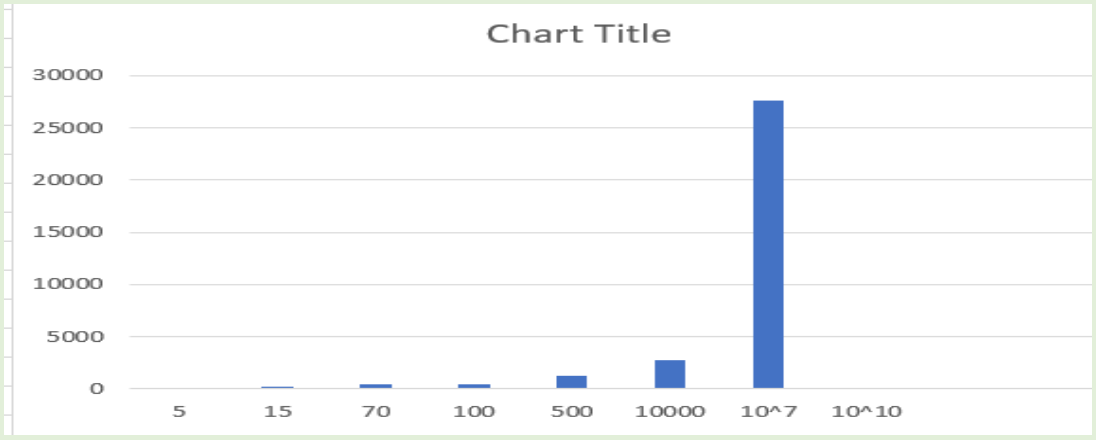
Chart Title
30000
25000
20000
15000
10000
5000
0
5
15
70
100
500
10000
10^7
10^10

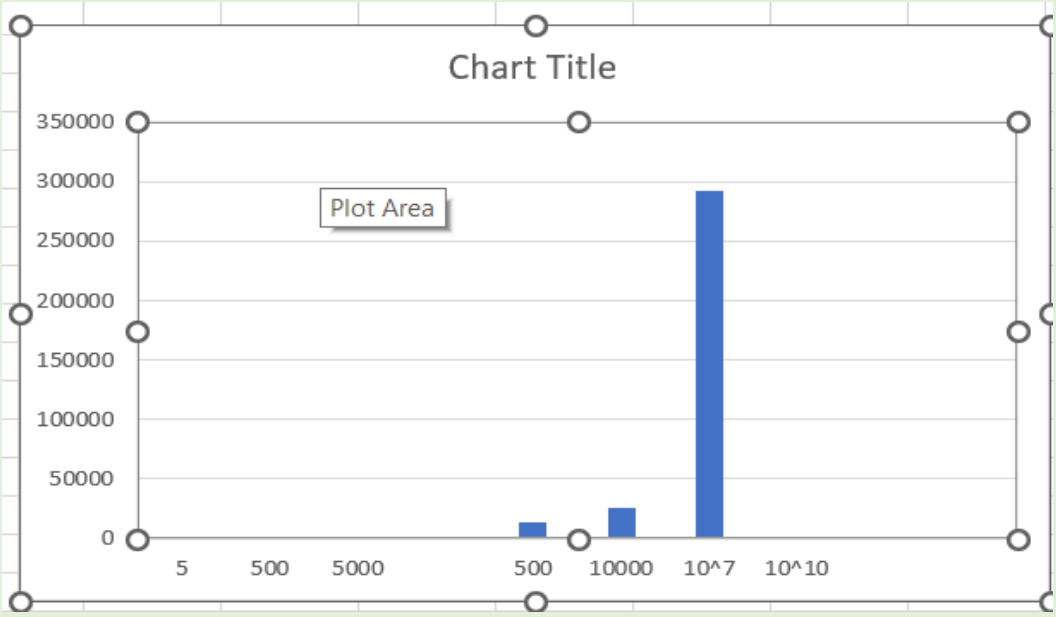
Chart Title
350000
300000
250000
200000
150000
100000
50000
0
Plot Area
5
500
5000
500
10000
10^7
10^10

```
#include <iostream>
using namespace std;
 long long iterative(int n) {
    long long fact = 1;
    for (int j = 1; j <= n; ++j) {
        fact *= j;}
    std::cout << fact << std::endl;
    return fact;
}
long long recursive(int n) {
    if (n == 0)
        return 1;
   else {
        return n * recursive(n - 1);}
}
int main()
{   int n;
   cin>>n;
    iterative(n);
    recursive(n);}
```