Name: Alaa Baniodeh

Student-ID: 202111487

- ***Execution time result:***

n	**Iterative Time (seconds)**	**Recursive Time (seconds)**
5	4	5
10	2	3
20	3	4
10000	3	6

- ## Stack Overflow Observations:

Attempting to calculate the factorial for `n = 10000` using the recursive approach may result in a stack overflow due to the depth of the recursion exceeding the stack limit. The iterative approach, being more space-efficient, is likely to handle this case without issues.

- ## Discussion and Conclusion:

- The iterative approach generally performs better in terms of execution time compared to the recursive approach, especially for large values of **n**.
- The recursive approach has a higher risk of stack overflow for large values of **n** due to the depth of the recursion.
- For large factorials, an iterative approach or optimized algorithms would be more suitable.
- The choice between iterative and recursive approaches depends on factors like performance, readability, and memory usage.

- **Factorial Implementations:**

```cpp
#include<iostream>
#include<ctime>
using namespace std;


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

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

int main(){

    time_t start_time = time(NULL);
    int n , y , result;

    cout<<"plz enter a num ....> ";
    cin>>n;

    cout<<"1)iterative \n 2)recursive "<<endl;
    cout<<"plz enter a function ... ";
    cin>>y;
```

```cpp
int main(){

    time_t start_time = time(NULL);
    int n , y , result;

    cout<<"plz enter a num ....> ";
    cin>>n;

    cout<<"1)iterative \n 2)recursive "<<endl;
    cout<<"plz enter a function ... ";
    cin>>y;

    if(y==1){
        result = iterative_fact(n);
        cout<<"the result is = "<<result<<endl;
    }else{
        result = recursive_fact(n);
        cout<<"the result is = "<<result<<endl;
    }
    time_t end_time = time(NULL);

    cout << "Execution Time: " << end_time - start_time << " seconds" <<endl;

    return 0;
}
```