👩🏻‍💻

Assignment #1”yara rabaya”:

▼ ✍🏻 Questions:

2.Provide stack overflow observations:

The stack overflow occurred when we reached the value of 66. At this value and beyond, the compiler starts to behave abnormally, assigning the factorial value as zero.

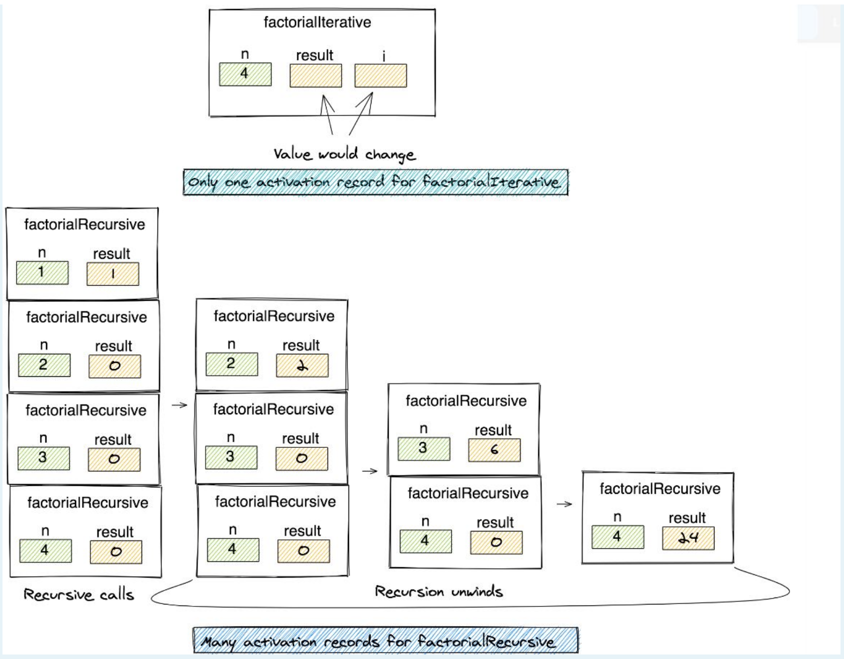
3.You should discuss what you have found:

1-The iterative method typically outperforms the recursive method in terms of execution time.

2-we faced stack overflow for larger 'n' values in each methods.

3-If we carefully analyze the problem by examining each method separately, considering the example of n=4, we observe that the iterative version requires only one function record on the activation stack. In the case of the recursive approach, there are 4 function records stored on the activation stack until the recursion begins to unwind.

Now, envision the implications when calculating the factorial for larger numbers such as 10 or 20. The accumulation of numerous records on the activation stack places a significant strain on the system's memory :



4.Conclude the work as a whole:

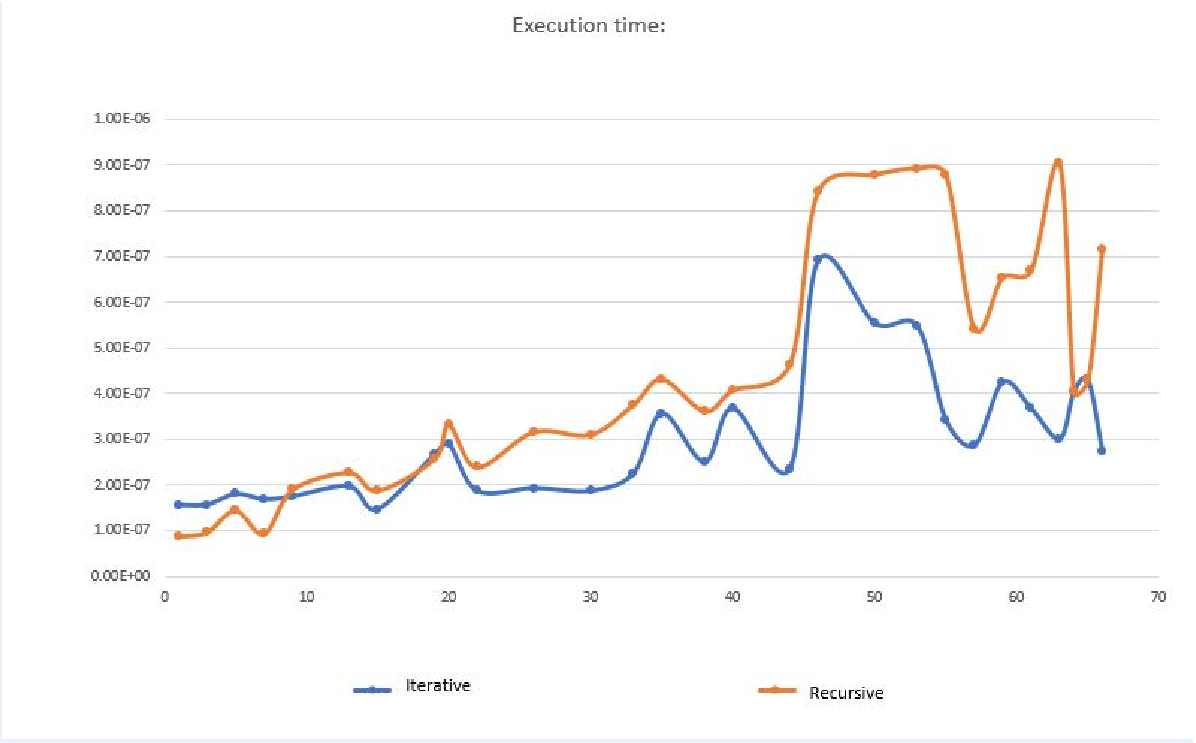
1-Both methods are impractical when there is a need to find the factorial of a large number, but if we are faced with choosing one of them, the iterative method will be preferred because it consumes less storage space in memory and is usually faster in performance also the iterative solution it run in constant memory.

2-Due to these outcomes and realizing practically that both methods are inefficient for larger values, I researched and found an alternative method, which is the gamma function:

```
#include <bits/stdc++.h>
int main (){
  cout << tgamma(66+1) << endl; //prints out 66!
}
```

1. Show execution time results in a presentable way (using tables and charts):

The value of INTEGER:	Factorial value:	Execution time of Iterative :	Execution time of Recursive :
5	120	1.8e-07 s	1.45e-07 s
7	5040	1.69e-07 s	9.2e-08 s
9	362880	1.75e-07 s	1.9e-07 s
13	6227020800	1.97e-07 s	2.27e-07 s
15	1307674368000	1.47e-07 s	1.88e-07 s
19	121645100408832000	2.67e-07 s	2.57e-07 s
20	2432902008176640000	2.89e-07 s	3.31e-07 s
22	17196083355034583040	1.87e-07 s	2.39e-07 s
26	16877220553537093632	1.93e-07 s	3.16e-07 s
30	9682165104862298112	1.87e-07 s	3.1e-07 s
33	3400198294675128320	2.25e-07 s	3.75e-07 s
35	6399018521010896896	3.56e-07 s	4.31e-07 s
40	18376134811363311616	3.68e-07 s	4.07e-07 s
44	2673996885588443136	2.35e-07 s	4.63e-07 s
46	1150331055211806720	6.91e-07 s	8.42e-07 s
50	15188249005818642432	5.54e-07 s	8.79e-07 s
53	13175843659825807360	5.47e-07 s	8.93e-07 s
55	6711489344688881664	3.41e-07 s	8.78e-07 s
57	6404118670120845312	2.86e-07 s	5.42e-07 s
59	162129586585337856	4.24e-07 s	6.54e-07 s
61	3098476543630901248	3.67e-07 s	6.68e-07 s
63	1585267068834414592	3e-07 s	9.03e-07 s
64	9223372036854775808	4.03e-07 s	4.03e-07 s
65	9223372036854775808	4.29e-07 s	4.29e-07 s
66	0	2.74e-07 s	7.15e-07 s



▼ ✍🏻 ***Information sources & online compiler:***

GDB online Debugger | Compiler - Code, Compile, Run, Debug online C, C++

Online GDB is online compiler and debugger for C/C++. You can compile, run and debug code with gdb online. Using gcc/g++ as compiler and gdb as debugger. Currently C and C++ languages are supported.

https://www.onlinegdb.com/

Chrono in C++ - GeeksforGeeks

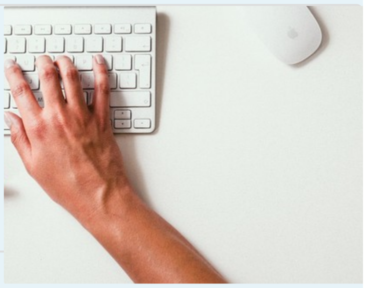
A Computer Science portal for geeks. It contains well written, well thought and well explained computer science and programming articles, quizzes and practice/competitive programming/company

https://www.geeksforgeeks.org/chrono-in-c/

How do you write a C++ factorial program without using recursive?

Answer (1 of 5): A key observation about recursion: The call stack is the place where temporary information is stored between recursive calls. You need to determine what is being stored on the stack. Create the variables

https://www.quora.com/How-do-you-write-a-C++-factorial-program-without-using-recursive

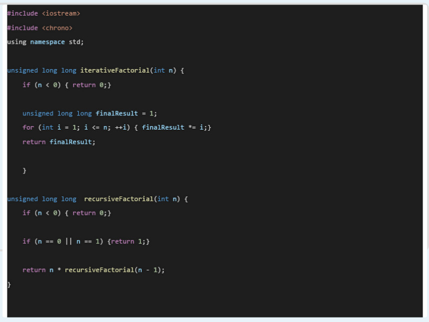


▼ ⚠️the source *code :*

yara code.docx

#include <iostream> #include <chrono> using namespace std; unsigned long long iterativeFactorial(int n) {   if (n < 0) { return 0;}   unsigned long long finalResult = 1;   for (int i =

https://docs.google.com/document/d/1HJ1DOUla-ZIdgyp2AFov1ihL-DbizGdi/edit?usp=sharing&ouid=104509552534706405055&rtpof=true&sd=true



```
#include <iostream>
#include <chrono>
using namespace std;

unsigned long long iterativeFactorial(int n) {
    if (n < 0) { return 0;}

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

    }

unsigned long long  recursiveFactorial(int n) {
    if (n < 0) { return 0;}

    if (n == 0 || n == 1) {return 1;}

    return n * recursiveFactorial(n - 1);
}
```