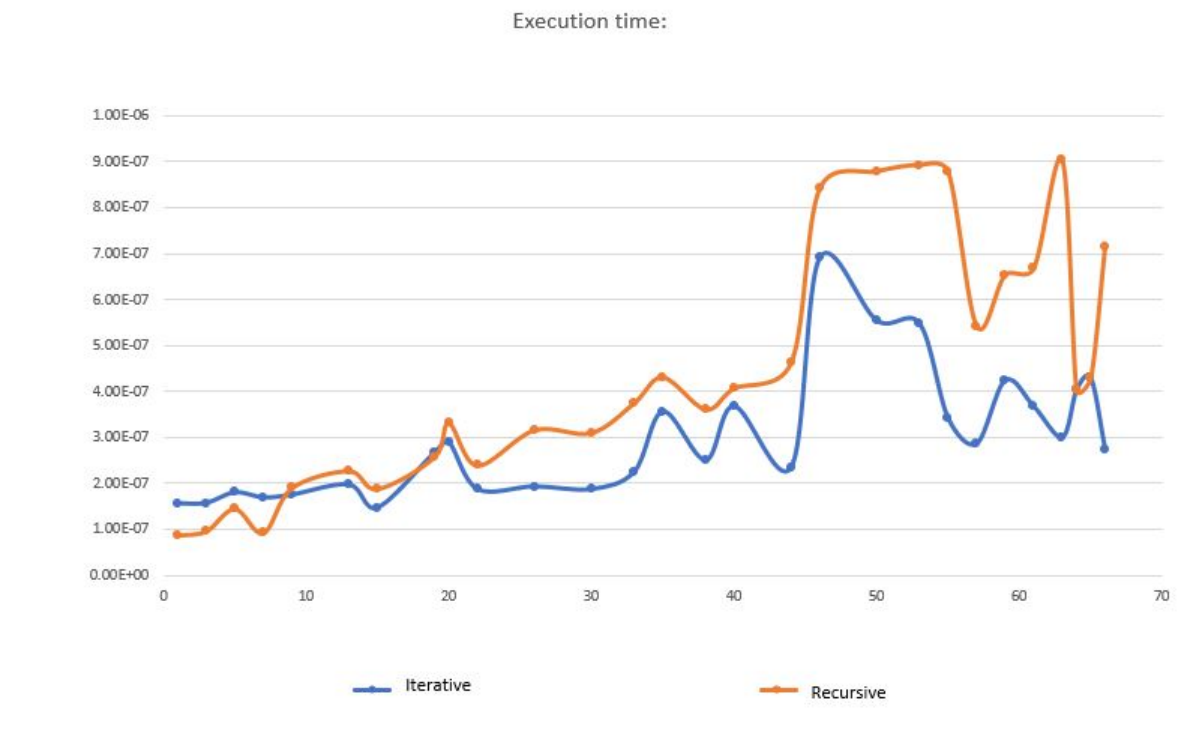
Assignment #1”yara jehad rabaya”:

▼ ✍️ Questions:

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



The value of INTEGER:	Factorial value:	Factorial value For Recursive	Execution time of Recursive :	Execution time of Iterative :
5	120	120	1.45e-07 s	1.8e-07 s
7	5040	5040	9.2e-08 s	1.69e-07 s
9	362880	362880	1.9e-07 s	1.75e-07 s
13	6227020800	6227020800	2.27e-07 s	1.97e-07 s
15	1307674368000	1307674368000	1.88e-07 s	1.47e-07 s
19	121645100408832000	121645100408832000	2.57e-07 s	2.67e-07 s
20	2432902008176640000	2432902008176640000	3.31e-07 s	2.89e-07 s
22	17196083355034583040	17196083355034583040	2.39e-07 s	1.87e-07 s
26	16877220553537093632	16877220553537093632	3.16e-07 s	1.93e-07 s
30	9682165104862298112	9682165104862298112	3.1e-07 s	1.87e-07 s
33	3400198294675128320	3400198294675128320	3.75e-07 s	2.25e-07 s
35	6399018521010896896	6399018521010896896	4.31e-07 s	3.56e-07 s
40	18376134811363311616	18376134811363311616	4.07e-07 s	3.68e-07 s
44	2673996885588443136	2673996885588443136	4.63e-07 s	2.35e-07 s
46	1150331055211806720	1150331055211806720	8.42e-07 s	6.91e-07 s
50	15188249005818642432	15188249005818642432	8.79e-07 s	5.54e-07 s
53	13175843659825807360	13175843659825807360	8.93e-07 s	5.47e-07 s
55	6711489344688881664	6711489344688881664	8.78e-07 s	3.41e-07 s
57	6404118670120845312	6404118670120845312	5.42e-07 s	2.86e-07 s
59	162129586585337856	162129586585337856	6.54e-07 s	4.24e-07 s
61	3098476543630901248	3098476543630901248	6.68e-07 s	3.67e-07 s
63	1585267068834414592	1585267068834414592	9.03e-07 s	3e-07 s
64	9223372036854775808	9223372036854775808	4.03e-07 s	4.03e-07 s
65	9223372036854775808	9223372036854775808	4.29e-07 s	4.29e-07 s
66	0	0	7.15e-07 s	2.74e-07 s
.	———	———	———-	———-
.	———	———	———--	———-
.	———	———	———	———-
40000	0	nan	0.000124832 s	———-

2.Provide stack overflow observations:

1-we faced stack overflow in recursive method for larger 'n' values (eg: if ‘n’ more than 30000)in recursive method, and this was observed when the program was terminating without producing any result .

2-we faced data type overflow in recursive and iterative method and this was observed when the program give the result =0 for larger 'n' values (Note: in my device for ‘n’ more than 65) .

3.You should discuss what you have found:

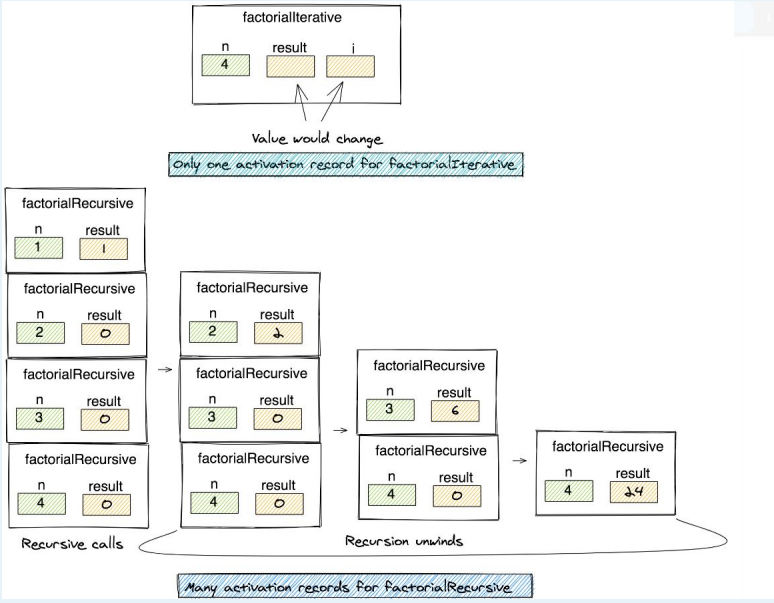
1-Stack overflow: For very large values of 'n', a stack overflow might occur due to an excessive number of recursive calls, exhausting the available memory for the call stack. This can cause the program to terminate without producing a result.

2-Data type overflow : Factorial calculations for large numbers can lead to extremely large results that might exceed the capacity of the chosen data type.

1-The iterative method typically outperforms the recursive method: In terms of execution time and memory.

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 become impractical when calculating the factorial of a large number. However, if a choice between them is necessary, the iterative method is preferable. It consumes less memory space, usually delivers better performance, and runs within 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! }
```

▼ ⚠️**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 = 1; i <= n; ++i) { finalResult *= i;} return

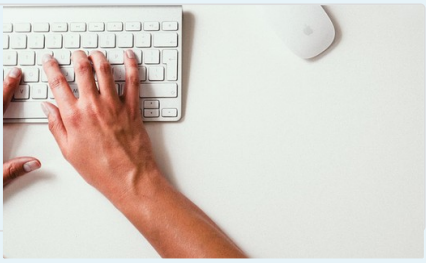
https://docs.google.com/document/d/1HJ1DOUla-ZIdgyp2AFov1ihL-DbizGdi/edit?usp=sharing&ouid=104509552534706405055&rtpof=true&sd=true

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

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



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 interview Questions.

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

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/