Rahul Rai

20110156

Computer Architecture: Assignment 1

Q1. (a) Using Recursion – Calculating the time for printing 100 numbers is not possible. In the worst case the algorithm needs to do 2^{100} instructions. Assuming the Clock rate to be 1Ghz, which means our laptop can handle approximately 10^{9} instructions. Then also 10^{20} seconds will be left.

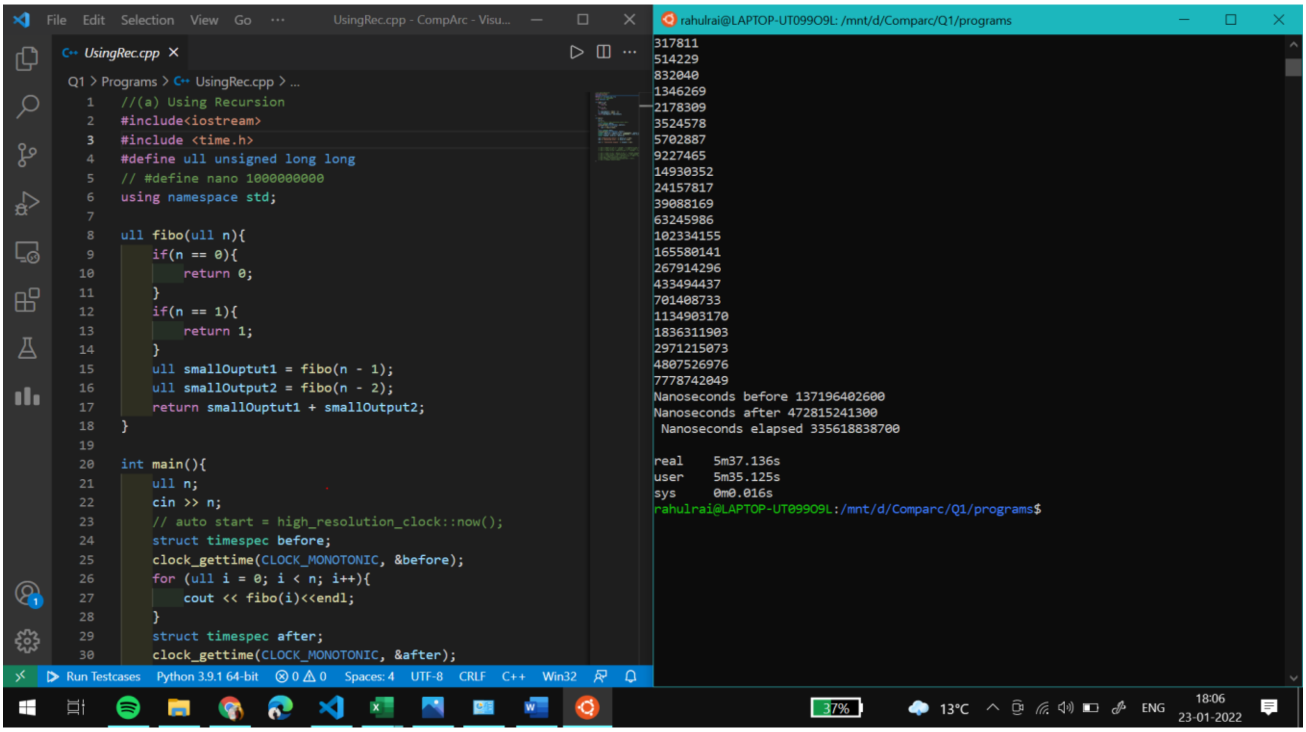
So, to calculate time using timespec, I calculated time to print first 50 numbers and after that calculated the time to print the next 50 numbers theoretically using the Time complexity of the algorithm.

To print 50 numbers, time required = 335618838700 nanoseconds

Time complexity of the Recursive Solution = 1.68^{n}

So Total Time = $335618838700 * (1.68)^{50}$

Total Time = $618.458291866 * 10^{20}$ nanoseconds

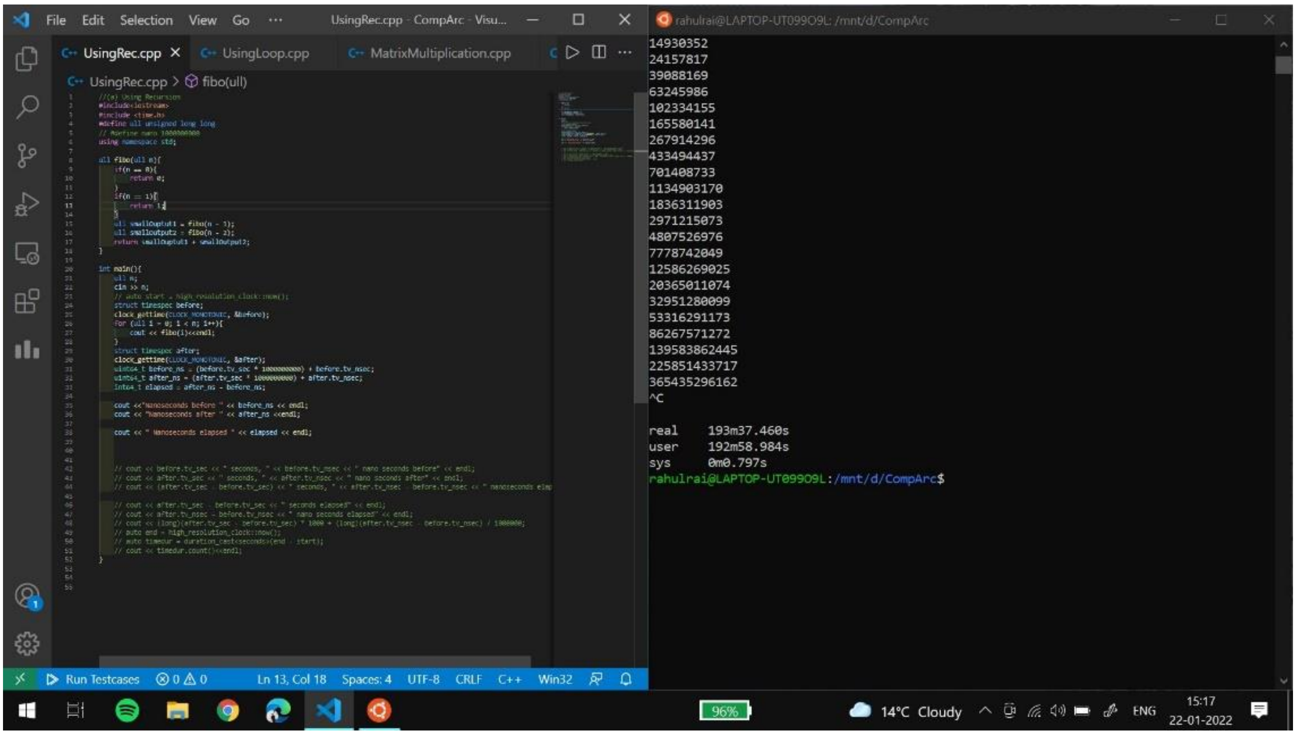


```
//(a) Using Recursion
#include<iostream>
#include <time.h>
#define ull unsigned long long
// #define nano 1000000000
using namespace std;

ull fibo(ull n){
    if(n == 0){
        return 0;
    }
    if(n == 1){
        return 1;
    }
    ull smallOuptut1 = fibo(n - 1);
    ull smallOutput2 = fibo(n - 2);
    return smallOuptut1 + smallOutput2;
}

int main(){
    ull n;
    cin >> n;
    // auto start = high_resolution_clock::now();
    struct timespec before;
    clock_gettime(CLOCK_MONOTONIC, &before);
    for (ull i = 0; i < n; i++){
        cout << fibo(i)<<endl;
    }
    struct timespec after;
    clock_gettime(CLOCK_MONOTONIC, &after);
```

```
317811
514229
832040
1346269
2178309
3524578
5702887
9227465
14930352
24157817
39088169
63245986
102334155
165580141
267914296
433494437
701408733
1134903170
1836311903
2971215073
4807526976
7778742049
Nanoseconds before 137196402600
Nanoseconds after 472815241300
 Nanoseconds elapsed 335618838700

real    5m37.136s
user    5m35.125s
sys     0m0.016s
rahulrai@LAPTOP-UT099O9L:/mnt/d/Comparc/Q1/programs$
```

Figure 1 Time using Timespec for first 50 numbers

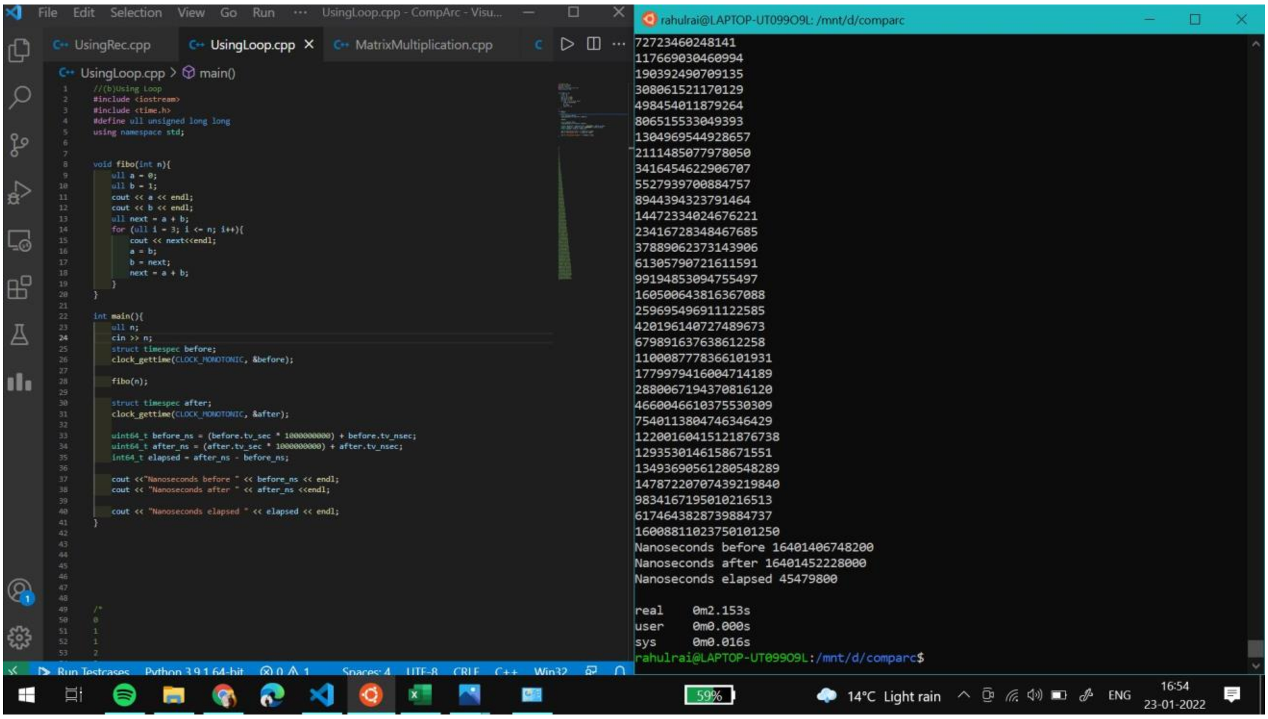
I also tried to run the program for 3 hours but it was able to print first 54-55 numbers only. Below is the attached screenshot.



```cpp
//(a) Using Recursion
#include<iostream>
#include <time.h>
#define ull unsigned long long
// #define nano 1000000000
using namespace std;

ull fibo(ull n){
    if(n == 0){
        return 0;
    }
    if(n == 1){
        return 1;
    }
    ull smallOuptut1 = fibo(n - 1);
    ull smalloutput2 = fibo(n - 2);
    return smallOuptut1 + smallOutput2;
}

int main(){
    ull n;
    cin >> n;
    // auto start = high_resolution_clock::now();
    struct timespec before;
    clock_gettime(CLOCK_MONOTONIC, &before);
    for (ull i = 0; i < n; i++){
        cout << fibo(i)<<endl;
    }
    struct timespec after;
    clock_gettime(CLOCK_MONOTONIC, &after);
    uint64_t before_ns = (before.tv_sec * 1000000000) + before.tv_nsec;
    uint64_t after_ns = (after.tv_sec * 1000000000) + after.tv_nsec;
    int64_t elapsed = after_ns - before_ns;

    cout <<"Nanoseconds before " << before_ns << endl;
    cout << "Nanoseconds after " << after_ns <<endl;

    cout << " Nanoseconds elapsed " << elapsed << endl;



    // cout << before.tv_sec << " seconds, " << before.tv_nsec << " nano seconds before" << endl;
    // cout << after.tv_sec << " seconds, " << after.tv_nsec << " nano seconds after" << endl;
    // cout << (after.tv_sec - before.tv_sec) << " seconds, " << after.tv_nsec - before.tv_nsec << " nanoseconds elap

    // cout << after.tv_sec - before.tv_sec << " seconds elapsed" << endl;
    // cout << after.tv_nsec - before.tv_nsec << " nano seconds elapsed" << endl;
    // cout << (long)(after.tv_sec - before.tv_sec) * 1000 + (long)(after.tv_nsec - before.tv_nsec) / 1000000;
    // auto end = high_resolution_clock::now();
    // auto timeDur = duration_cast<seconds>(end - start);
    // cout << timedur.count()<<endl;
}
```

```
14930352
24157817
39088169
63245986
102334155
165580141
267914296
433494437
701408733
1134903170
1836311903
2971215073
4807526976
7778742049
12586269025
20365011074
32951280099
53316291173
86267571272
139583862445
225851433717
365435296162
^C

real    193m37.460s
user    192m58.984s
sys     0m0.797s
rahulrai@LAPTOP-UT099O9L:/mnt/d/CompArc$
```

(b) Using Loop

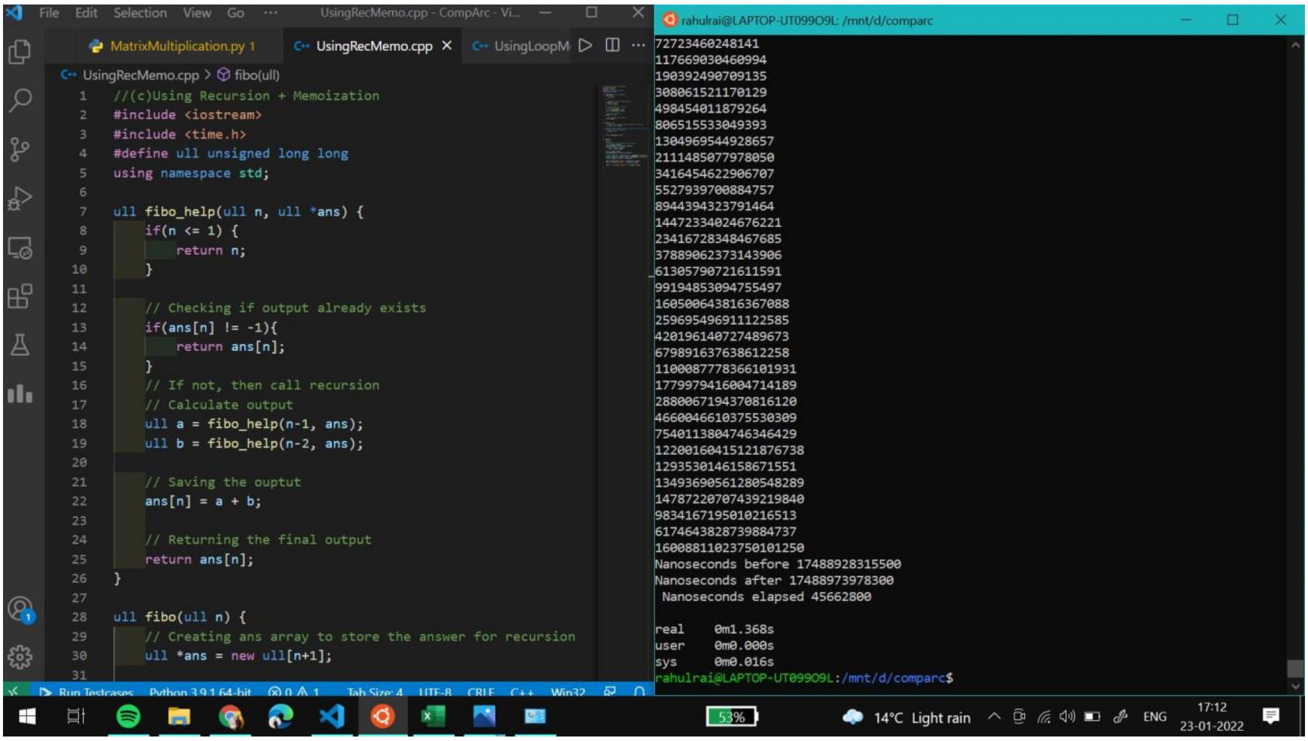


```cpp
//(b)Using Loop
#include <iostream>
#include <time.h>
#define ull unsigned long long
using namespace std;


void fibo(int n){
    ull a = 0;
    ull b = 1;
    cout << a << endl;
    cout << b << endl;
    ull next = a + b;
    for (ull i = 3; i <= n; i++){
        cout << next<<endl;
        a = b;
        b = next;
        next = a + b;
    }
}

int main(){
    ull n;
    cin >> n;
    struct timespec before;
    clock_gettime(CLOCK_MONOTONIC, &before);

    fibo(n);

    struct timespec after;
    clock_gettime(CLOCK_MONOTONIC, &after);

    uint64_t before_ns = (before.tv_sec * 1000000000) + before.tv_nsec;
    uint64_t after_ns = (after.tv_sec * 1000000000) + after.tv_nsec;
    int64_t elapsed = after_ns - before_ns;

    cout <<"Nanoseconds before " << before_ns << endl;
    cout << "Nanoseconds after " << after_ns <<endl;

    cout << "Nanoseconds elapsed " << elapsed << endl;
}








/*
0
1
1
2
```

```
72723460248141
117669030460994
190392490709135
308061521170129
498454011879264
806515533049393
1304969544928657
2111485077978050
3416454622906707
5527939700884757
8944394323791464
14472334024676221
23416728348467685
37889062373143906
61305790721611591
99194853094755497
160500643816367088
259695496911122585
420196140727489673
679891637638612258
1100087778366101931
1779979416004714189
2880067194370816120
4660046610375530309
7540113804746346429
12200160415121876738
1293530146158671551
13493690561280548289
14787220707439219840
9834167195010216513
6174643828739884737
16008811023750101250
Nanoseconds before 16401406748200
Nanoseconds after 16401452228000
Nanoseconds elapsed 45479800

real    0m2.153s
user    0m0.000s
sys     0m0.016s
rahulrai@LAPTOP-UT099O9L:/mnt/d/comparc$
```

Time taken = 45479800 nanoseconds

(c) Using Recursion + Memoization

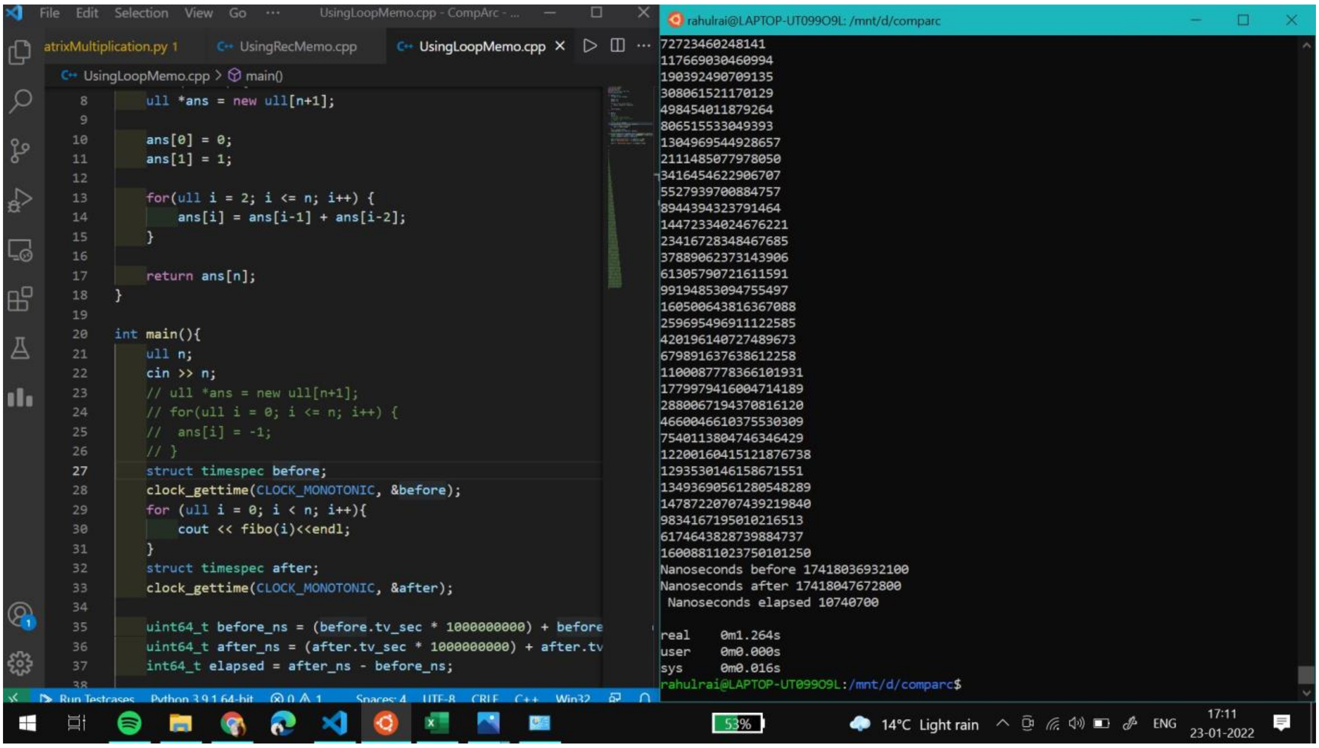

File Edit Selection View Go ···
UsingRecMemo.cpp - CompArc - Vi...
MatrixMultiplication.py 1
UsingRecMemo.cpp
UsingLoopM
UsingRecMemo.cpp > fibo(ull)
rahulrai@LAPTOP-UT099O9L: /mnt/d/comparc
Run Testcases Python 3.9.1 64-bit ⊗ 0 ⚠ 1 Tab Size: 4 UTF-8 CRLF C++ Win32
53%
14°C Light rain
ENG
17:12
23-01-2022

```cpp
//(c)Using Recursion + Memoization
#include <iostream>
#include <time.h>
#define ull unsigned long long
using namespace std;

ull fibo_help(ull n, ull *ans) {
    if(n <= 1) {
        return n;
    }

    // Checking if output already exists
    if(ans[n] != -1){
        return ans[n];
    }
    // If not, then call recursion
    // Calculate output
    ull a = fibo_help(n-1, ans);
    ull b = fibo_help(n-2, ans);

    // Saving the ouptut
    ans[n] = a + b;

    // Returning the final output
    return ans[n];
}

ull fibo(ull n) {
    // Creating ans array to store the answer for recursion
    ull *ans = new ull[n+1];
```

```
72723460248141
117669030460994
190392490709135
308061521170129
498454011879264
806515533049393
1304969544928657
2111485077978050
3416454622906707
5527939700884757
8944394323791464
14472334024676221
23416728348467685
37889062373143906
61305790721611591
99194853094755497
160500643816367088
259695496911122585
420196140727489673
679891637638612258
1100087778366101931
1779979416004714189
2880067194370816120
4660046610375530309
7540113804746346429
12200160415121876738
1293530146158671551
13493690561280548289
14787220707439219840
9834167195010216513
6174643828739884737
16008811023750101250
Nanoseconds before 17488928315500
Nanoseconds after 17488973978300
 Nanoseconds elapsed 45662800

real    0m1.368s
user    0m0.000s
sys     0m0.016s
rahulrai@LAPTOP-UT099O9L:/mnt/d/comparc$
```

Time taken = 45662800 nanoseconds

(d) Using Loop and Memo


File Edit Selection View Go ···
UsingLoopMemo.cpp - CompArc - ...
atrixMultiplication.py 1
UsingRecMemo.cpp
UsingLoopMemo.cpp
UsingLoopMemo.cpp > main()
rahulrai@LAPTOP-UT099O9L: /mnt/d/comparc
Run Testcases Python 3.9.1 64-bit ⊗ 0 ⚠ 1 Spaces: 4 UTF-8 CRLF C++ Win32
53%
14°C Light rain
ENG
17:11
23-01-2022

```cpp
    ull *ans = new ull[n+1];

    ans[0] = 0;
    ans[1] = 1;

    for(ull i = 2; i <= n; i++) {
        ans[i] = ans[i-1] + ans[i-2];
    }

    return ans[n];
}

int main(){
    ull n;
    cin >> n;
    // ull *ans = new ull[n+1];
    // for(ull i = 0; i <= n; i++) {
    //  ans[i] = -1;
    // }
    struct timespec before;
    clock_gettime(CLOCK_MONOTONIC, &before);
    for (ull i = 0; i < n; i++){
        cout << fibo(i)<<endl;
    }
    struct timespec after;
    clock_gettime(CLOCK_MONOTONIC, &after);

    uint64_t before_ns = (before.tv_sec * 1000000000) + before
    uint64_t after_ns = (after.tv_sec * 1000000000) + after.tv
    int64_t elapsed = after_ns - before_ns;
```

```
72723460248141
117669030460994
190392490709135
308061521170129
498454011879264
806515533049393
1304969544928657
2111485077978050
3416454622906707
5527939700884757
8944394323791464
14472334024676221
23416728348467685
37889062373143906
61305790721611591
99194853094755497
160500643816367088
259695496911122585
420196140727489673
679891637638612258
1100087778366101931
1779979416004714189
2880067194370816120
4660046610375530309
7540113804746346429
12200160415121876738
1293530146158671551
13493690561280548289
14787220707439219840
9834167195010216513
6174643828739884737
16008811023750101250
Nanoseconds before 17418036932100
Nanoseconds after 17418047672800
 Nanoseconds elapsed 10740700

real    0m1.264s
user    0m0.000s
sys     0m0.016s
rahulrai@LAPTOP-UT099O9L:/mnt/d/comparc$
```

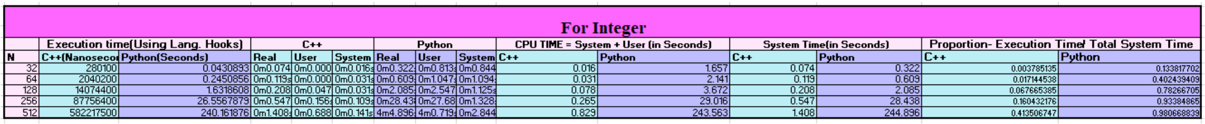
Time taken = 10740700 nanoseconds

Time using Timespec (in nanoseconds)			
		Time	**Speedup**
Program 1	Using Recursion	6.18458E+22	1
Program 2	Using Loop	45479800	7.35374E-16
Program 3	Using Recursion and Memoi	45662800	7.38333E-16
Program 4	Using Loop and Memoizatio	10740700	1.73669E-16

SpeedUp = Time required by Program 1/ Time required by program I where i =2, 3, 4

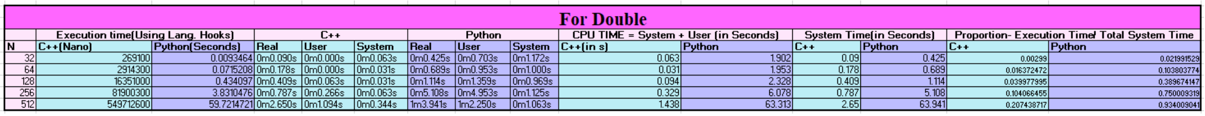
Q2. For the codes, Go to Q1/Programs/

For part (a) and part(b), refer the Excel sheet.



For Integer

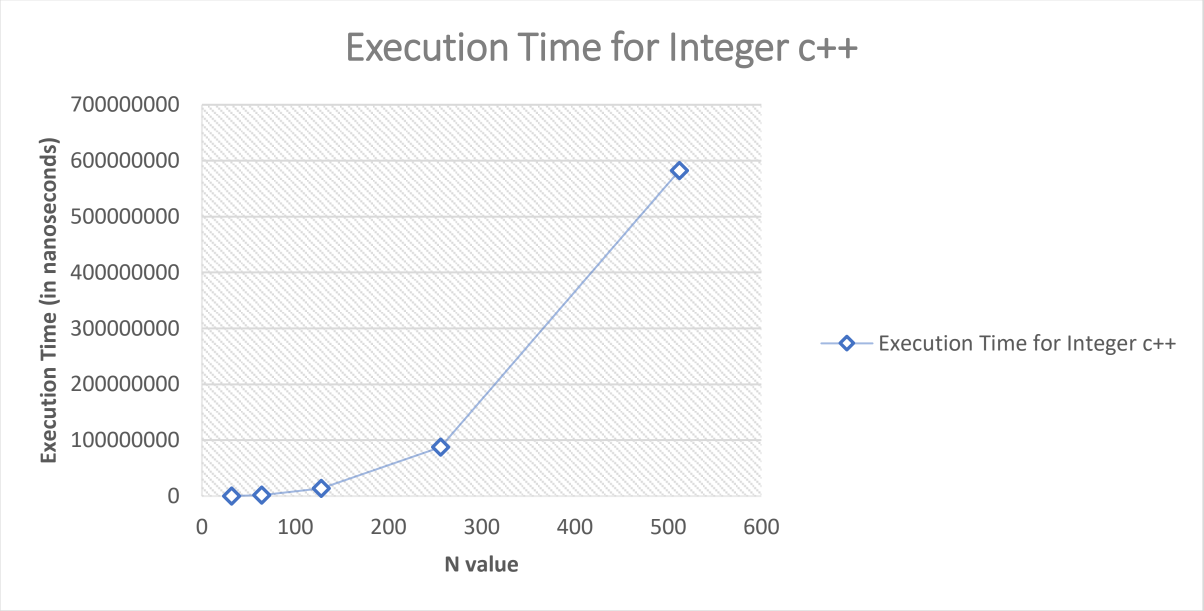
	Execution time(Using Lang. Hooks)		C++			Python			CPU TIME = System + User (in Seconds)		System Time(in Seconds)		Proportion- Execution Time/ Total System Time	
N	C++(Nanosecor	Python(Seconds)	Real	User	System	Real	User	System	C++	Python	C++	Python	C++	Python
32	280100	0.0430893	0m0.074	0m0.000	0m0.016s	0m0.322	0m0.813s	0m0.844	0.016	1.657	0.074	0.322	0.003785135	0.133817702
64	2040200	0.2450856	0m0.119s	0m0.000	0m0.031s	0m0.609	0m1.047s	0m1.094	0.031	2.141	0.119	0.609	0.017144538	0.402439409
128	14074400	1.6318608	0m0.208	0m0.047	0m0.031s	0m2.085	0m2.547	0m1.125s	0.078	3.672	0.208	2.085	0.067665385	0.78266705
256	87756400	26.5567879	0m0.547	0m0.156s	0m0.109s	0m28.43	0m27.68	0m1.328	0.265	29.016	0.547	28.438	0.160432176	0.93384865
512	582217500	240.161876	0m1.408s	0m0.688	0m0.141s	4m4.896	4m0.719s	0m2.844	0.829	243.563	1.408	244.896	0.413506747	0.980668839

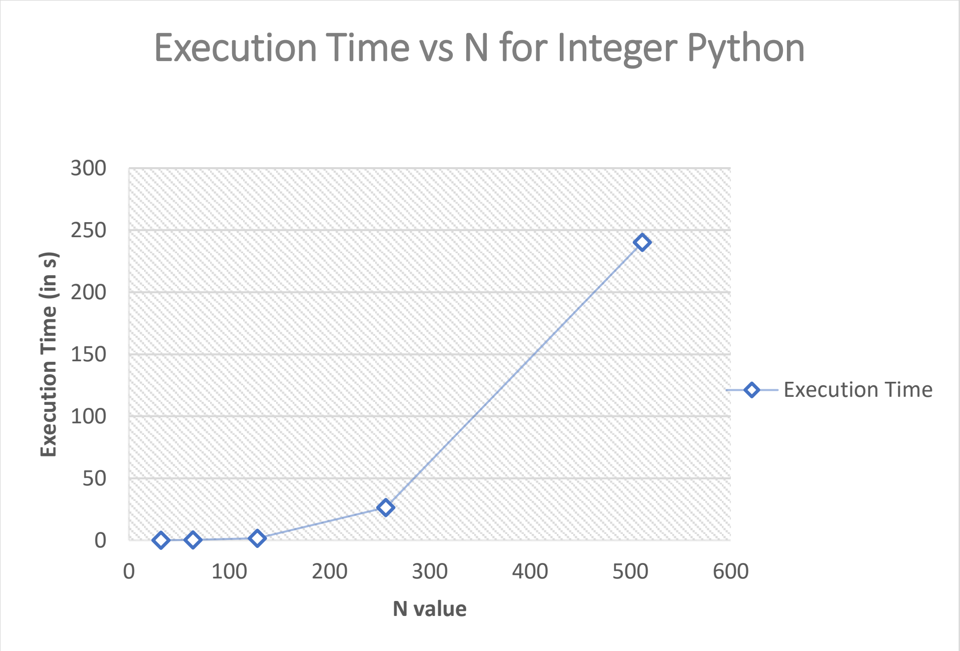


For Double

	Execution time(Using Lang. Hooks)		C++			Python			CPU TIME = System + User (in Seconds)		System Time(in Seconds)		Proportion- Execution Time/ Total System Time	
N	C++(Nano)	Python(Seconds)	Real	User	System	Real	User	System	C++(in s)	Python	C++	Python	C++	Python
32	269100	0.0093464	0m0.090s	0m0.000s	0m0.063s	0m0.425s	0m0.703s	0m1.172s	0.063	1.902	0.09	0.425	0.00299	0.021991529
64	2914300	0.0715208	0m0.178s	0m0.000s	0m0.031s	0m0.689s	0m0.953s	0m1.000s	0.031	1.953	0.178	0.689	0.016372472	0.103803774
128	16351000	0.434097	0m0.409s	0m0.063s	0m0.031s	0m1.114s	0m1.359s	0m0.969s	0.094	2.328	0.409	1.114	0.039977995	0.389674147
256	81900300	3.8310476	0m0.787s	0m0.266s	0m0.063s	0m5.108s	0m4.953s	0m1.125s	0.329	6.078	0.787	5.108	0.104066455	0.750009319
512	549712600	59.7214721	0m2.650s	0m1.094s	0m0.344s	1m3.941s	1m2.250s	0m1.063s	1.438	63.313	2.65	63.941	0.207438717	0.934009041

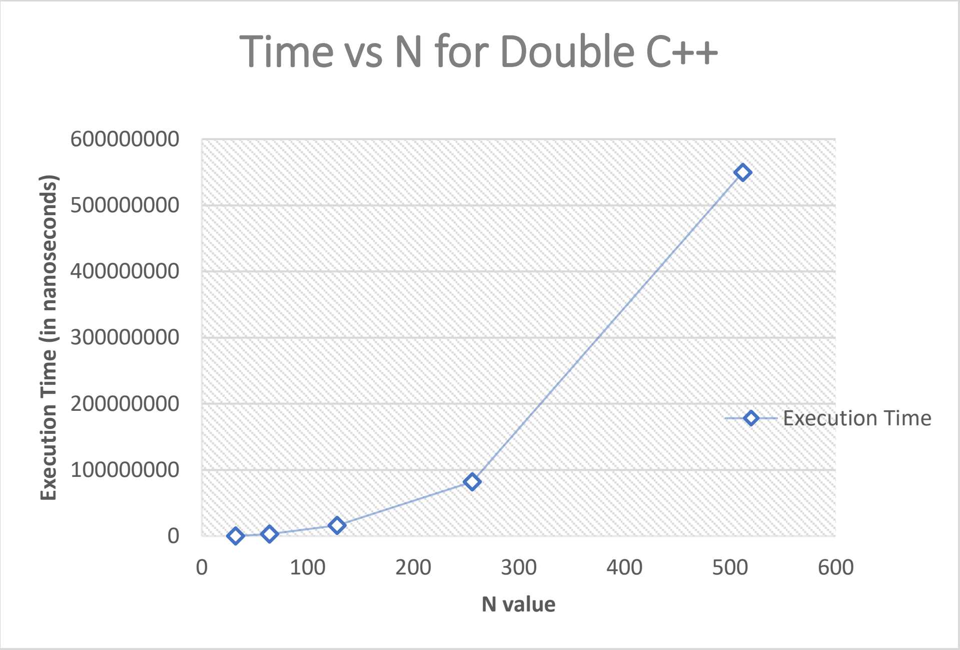
(c) Execution Time vs N

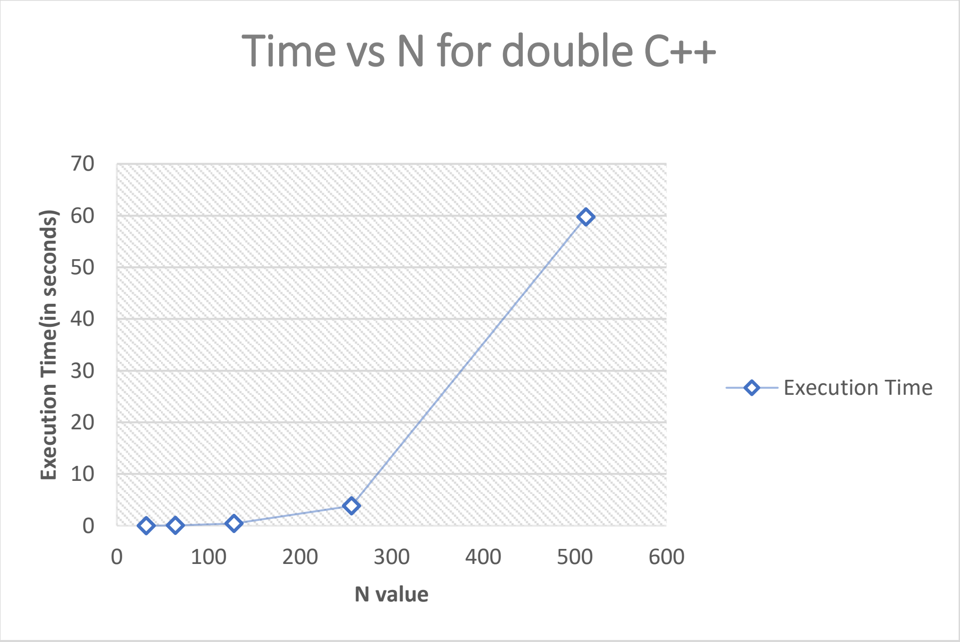




Less the execution time, better is the performance and vice versa.

For integers, with increasing value of N, we can see that Python (240 seconds approx for N = 512) is taking more time than C++ (only a few seconds for N = 512)to execute the program. This implies that C++ is relatively faster than python to execute the program.

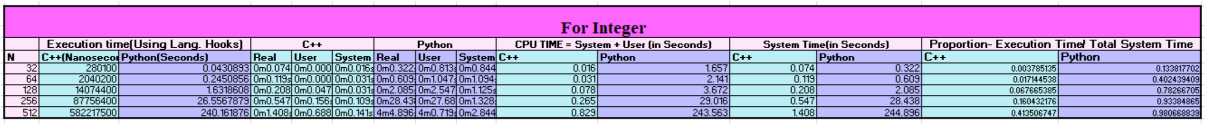




Considering the worst-case scenario for both the cases that is for N = 512

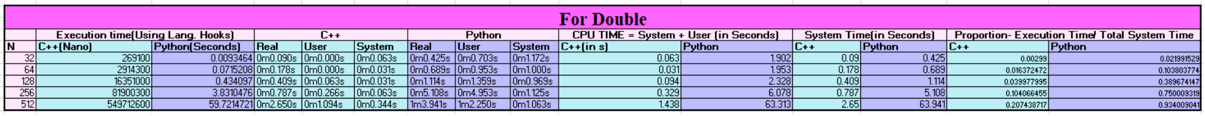
Less Execution time implies better performance. For N = 512, time taken by C++ is in seconds while python takes 59 seconds approx. to execute the same program. From this we can conclude that C++ is faster than python even in computing the double values.

Common observation in both the cases is the execution time increase as the value of n increases or in other words we can say that the program becomes slower when n value increases.



For Integer

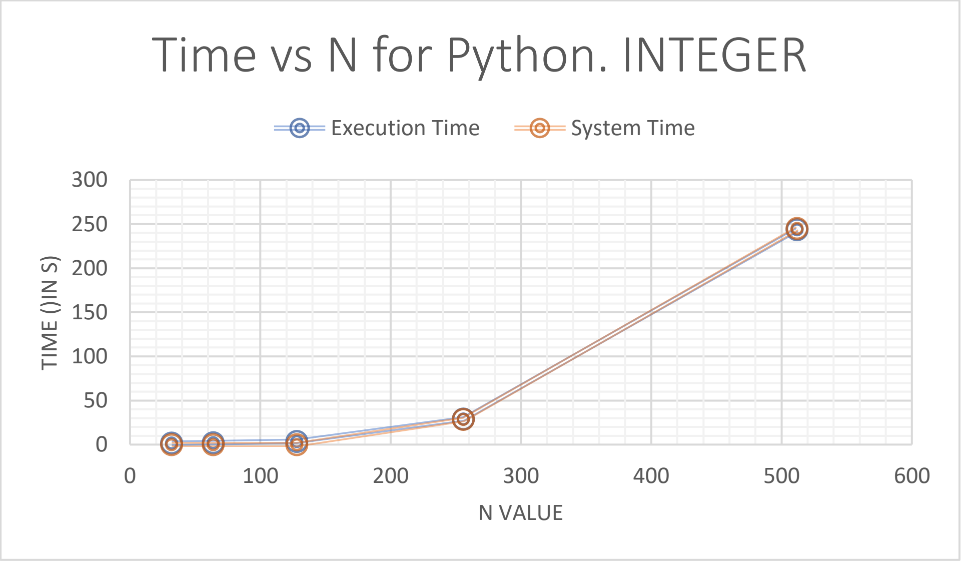
	Execution time(Using Lang. Hooks)		C++			Python			CPU TIME = System + User (in Seconds)		System Time(in Seconds)		Proportion- Execution Time/ Total System Time	
N	C++(Nanosecor	Python(Seconds)	Real	User	System	Real	User	System	C++	Python	C++	Python	C++	Python
32	280100	0.0430893	0m0.074	0m0.000	0m0.016	0m0.322	0m0.813	0m0.844	0.016	1.657	0.074	0.322	0.003785135	0.133817702
64	2040200	0.2450856	0m0.119s	0m0.000	0m0.031	0m0.609	0m1.047	0m1.094	0.031	2.141	0.119	0.609	0.017144538	0.402439409
128	14074400	1.6318608	0m0.208	0m0.047	0m0.031	0m2.085	0m2.547	0m1.125	0.078	3.672	0.208	2.085	0.067665385	0.78266705
256	87756400	26.5567879	0m0.547	0m0.156	0m0.109	0m28.43	0m27.68	0m1.328	0.265	29.016	0.547	28.438	0.160432176	0.93384865
512	582217500	240.161876	0m1.408	0m0.688	0m0.141s	4m4.896	4m0.719	0m2.844	0.829	243.563	1.408	244.896	0.413506747	0.980668839

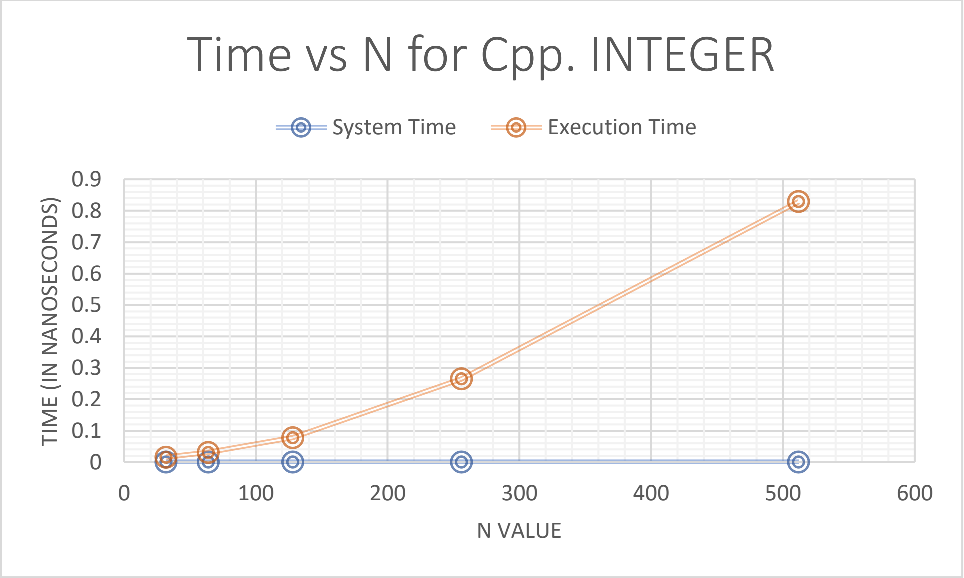


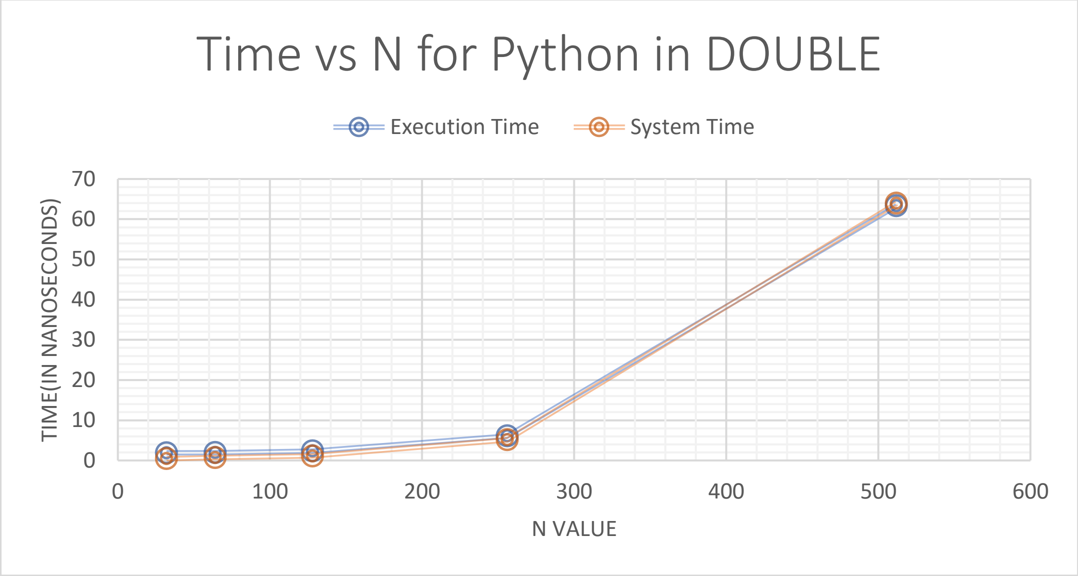
For Double

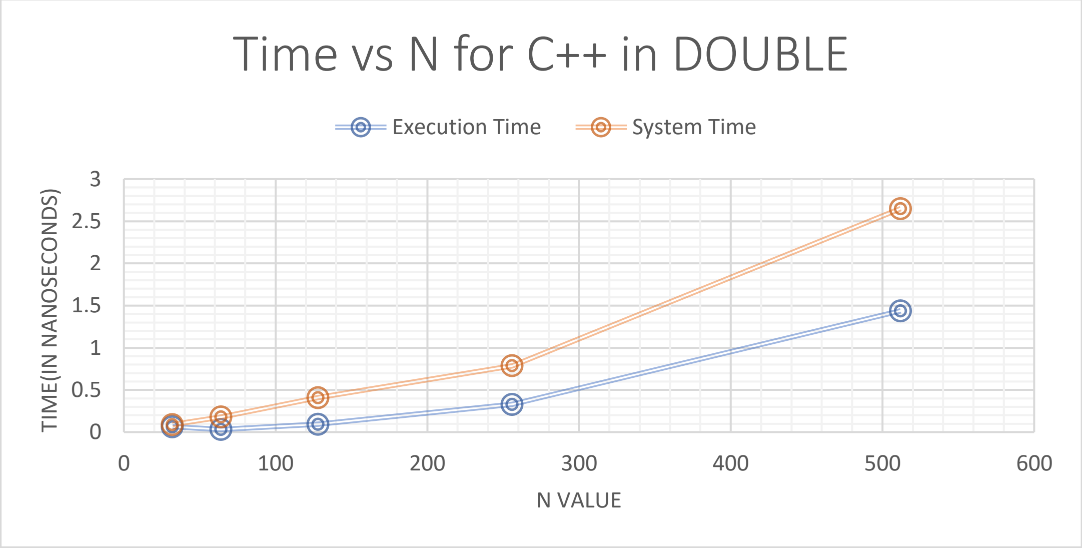
	Execution time(Using Lang. Hooks)		C++			Python			CPU TIME = System + User (in Seconds)		System Time(in Seconds)		Proportion- Execution Time/ Total System Time	
N	C++(Nano)	Python(Seconds)	Real	User	System	Real	User	System	C++(in s)	Python	C++	Python	C++	Python
32	269100	0.0093464	0m0.090s	0m0.000s	0m0.063s	0m0.425s	0m0.703s	0m1.172s	0.063	1.902	0.09	0.425	0.00299	0.021991529
64	2914300	0.0715208	0m0.178s	0m0.000s	0m0.031s	0m0.689s	0m0.953s	0m1.000s	0.031	1.953	0.178	0.689	0.016372472	0.103803774
128	16351000	0.434097	0m0.409s	0m0.063s	0m0.031s	0m1.114s	0m1.359s	0m0.969s	0.094	2.328	0.409	1.114	0.039977995	0.389674147
256	81900300	3.8310476	0m0.787s	0m0.266s	0m0.063s	0m5.108s	0m4.953s	0m1.125s	0.329	6.078	0.787	5.108	0.104066455	0.750009319
512	549712600	59.7214721	0m2.650s	0m1.094s	0m0.344s	1m3.941s	1m2.250s	0m1.063s	1.438	63.313	2.65	63.941	0.207438717	0.934009041

Here I have considered Execution Time as CPU time and System Time as Real Time.









In python, we can observe that the system and the execution time are varying nearly same.

In the case of C++, we can observe that in the case of integer, the System time is very less as compared to the program execution time.

In case of double for C++, we can observe that execution time is less than system time but the value of execution time is considerable.