Name : Tejas Parmar
Roll No. : 20110125

ES215

Computer Architecture and Organization

Assignment - 1

Q1. Fibonacci series upto 100th element:

→ I used timespec to get program execution time in ubuntu linux.

NOTE : Keeping in mind that we have to list 100 fibonacci series elements and find speedups compared to the program(reference program) which used recursion to find a solution. In that program, we can't go beyond the 50th element and so we can't compare other programs executing the list of 100 fibonacci elements with the reference one. So, to get a fair and verging solution I have run all programs for 50 elements to find the speedup of all programs.

I have also considered 100 elements in case the program is able to run it. I have shown it along with.

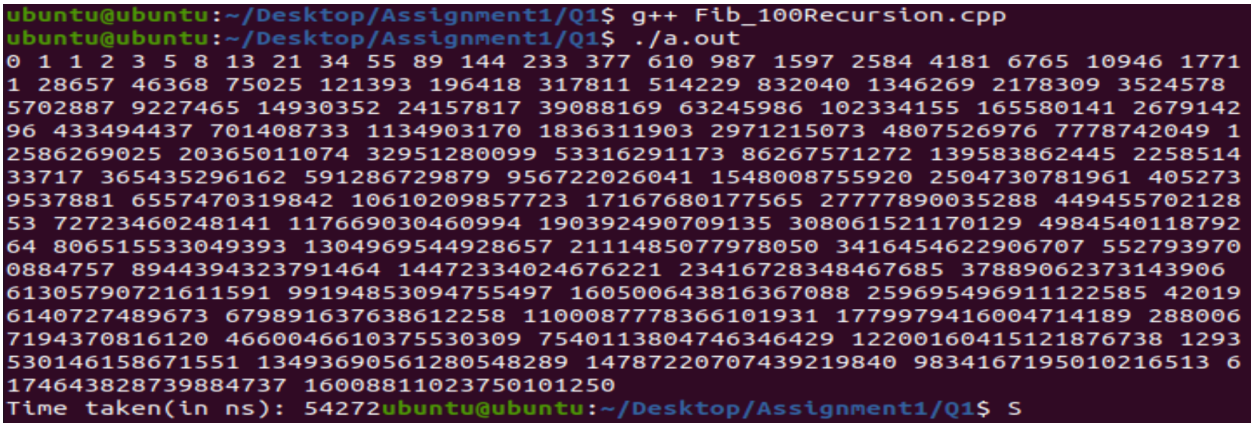
a) **Using Recursion(Upto 50th Element)**

```
Time taken(in ns): 116839390208ubuntu@ubuntu:~/Desktop/COA$ g++ Fibonacci_using
_Recursion.cpp
ubuntu@ubuntu:~/Desktop/COA$ ./a.out

0 1 1 2 3 5 8 13 21 34 55 89 144 233 377 610 987 1597 2584 4181 6765 10946 1771
1 28657 46368 75025 121393 196418 317811 514229 832040 1346269 2178309 3524578
5702887 9227465 14930352 24157817 39088169 63245986 102334155 165580141 2679142
96 433494437 701408733 1134903170 1836311903 2971215073 4807526976 7778742049
Time taken(in ns): 127589296128ubuntu@ubuntu:~/Desktop/COA$
```

- Time Taken (in ns): 127589296128 = 127.59 s
- Speedup : (127.59/127.59) = 1

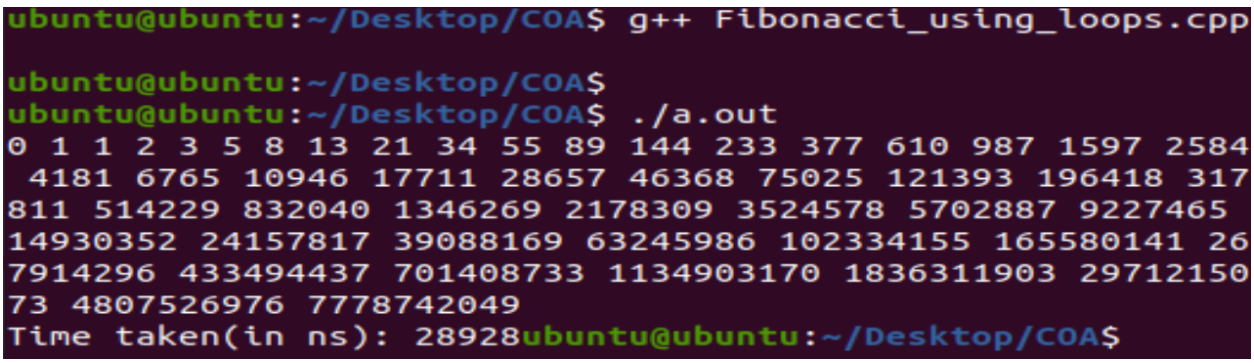
NOTE : To print 100 elements using Recursion, I've written another program for it which is named as (Fib_100Recursion.cpp).

***(Upto 100th Element)**



- Time Taken (in ns): 54272 = 0.000054 s

b) **Using Loops(Upto 50th Element)**



- Time Taken (in ns): 28928 = 0.000029 s
- Speedup : 4410581.31

***(Upto 100th Element)**

```
ubuntu@ubuntu:~/Desktop/COA$ g++ Fibonacci_using_loops.cpp
ubuntu@ubuntu:~/Desktop/COA$ ./a.out
0 1 1 2 3 5 8 13 21 34 55 89 144 233 377 610 987 1597 2584 4181 6765
 10946 17711 28657 46368 75025 121393 196418 317811 514229 832040 13
46269 2178309 3524578 5702887 9227465 14930352 24157817 39088169 632
45986 102334155 165580141 267914296 433494437 701408733 1134903170 1
836311903 2971215073 4807526976 7778742049 12586269025 20365011074 3
2951280099 53316291173 86267571272 139583862445 225851433717 3654352
96162 591286729879 956722026041 1548008755920 2504730781961 40527395
37881 6557470319842 10610209857723 17167680177565 27777890035288 449
45570212853 72723460248141 117669030460994 190392490709135 308061521
170129 498454011879264 806515533049393 1304969544928657 211148507797
8050 3416454622906707 5527939700884757 8944394323791464 144723340246
76221 23416728348467685 37889062373143906 61305790721611591 99194853
094755497 160500643816367088 259695496911122585 420196140727489673 6
79891637638612258 1100087778366101931 1779979416004714189 2880067194
370816120 4660046610375530309 7540113804746346429 122001604151218767
38 1293530146158671551 13493690561280548289 14787220707439219840 983
4167195010216513 6174643828739884737 16008811023750101250
Time taken(in ns): 81920ubuntu@ubuntu:~/Desktop/COA$
```

- Time Taken (in ns): 81920 = 0.000082 s

c) **Using Recursion with Memoization(Upto 50th Element)**

```
ubuntu@ubuntu:~/Desktop/COA$ g++ Fib_Recursionmemo.cpp
ubuntu@ubuntu:~/Desktop/COA$ ./a.out
0 1 1 2 3 5 8 13 21 34 55 89 144 233 377 610 987 1597 2584
 4181 6765 10946 17711 28657 46368 75025 121393 196418 317
811 514229 832040 1346269 2178309 3524578 5702887 9227465
14930352 24157817 39088169 63245986 102334155 165580141 26
7914296 433494437 701408733 1134903170 1836311903 29712150
73 4807526976 7778742049
Time taken(in ns): 26112ubuntu@ubuntu:~/Desktop/COA$
```

- Time taken (in ns): 26112 = 0.000026 s
- Speedup : 4889232.23

***(Upto 100th Element)**

```
ubuntu@ubuntu:~/Desktop/COA$ g++ Fib_Recursionmemo.cpp
ubuntu@ubuntu:~/Desktop/COA$ ./a.out
0 1 1 2 3 5 8 13 21 34 55 89 144 233 377 610 987 1597 2584
 4181 6765 10946 17711 28657 46368 75025 121393 196418 317
811 514229 832040 1346269 2178309 3524578 5702887 9227465
14930352 24157817 39088169 63245986 102334155 165580141 26
7914296 433494437 701408733 1134903170 1836311903 29712150
73 4807526976 7778742049 12586269025 20365011074 329512800
99 53316291173 86267571272 139583862445 225851433717 36543
5296162 591286729879 956722026041 1548008755920 2504730781
961 4052739537881 6557470319842 10610209857723 17167680177
565 27777890035288 44945570212853 72723460248141 117669030
460994 190392490709135 308061521170129 498454011879264 806
515533049393 1304969544928657 2111485077978050 34164546229
06707 5527939700884757 8944394323791464 14472334024676221
23416728348467685 37889062373143906 61305790721611591 9919
4853094755497 160500643816367088 259695496911122585 420196
140727489673 679891637638612258 1100087778366101931 177997
9416004714189 2880067194370816120 4660046610375530309 7540
113804746346429 12200160415121876738 19735301461586715511
3493690561280548289 14787220707439219840 98341671950102165
13 617464382873988473 7 160088110237501012 50
Time taken(in ns): 39168ubuntu@ubuntu:~/Desktop/COA$
```

- Time Taken (in ns): 39168 = 0.000039 s

d) **Using loops with Memoization(Upto 50th Element)**

```
ubuntu@ubuntu:~/Desktop/COA$ ./a.out
0 1 1 2 3 5 8 13 21 34 55 89 144 233 377 610 987 1597 2584
 4181 6765 10946 17711 28657 46368 75025 121393 196418 317
811 514229 832040 1346269 2178309 3524578 5702887 9227465
14930352 24157817 39088169 63245986 102334155 165580141 26
7914296 433494437 701408733 1134903170 1836311903 29712150
73 4807526976 7778742049
Time taken(in ns): 25856ubuntu@ubuntu:~/Desktop/COA$
```

- Time taken (in ns): 25856 = 0.000026 s
- Speedup : 4934610.77

***(Upto 100th Element)**

```
ubuntu@ubuntu:~/Desktop/COA$ g++ Fib_loopmemo.cpp
ubuntu@ubuntu:~/Desktop/COA$ ./a.out
0 1 1 2 3 5 8 13 21 34 55 89 144 233 377 610 987 1597 2584 4181 6765
 10946 17711 28657 46368 75025 121393 196418 317811 514229 832040 13
46269 2178309 3524578 5702887 9227465 14930352 24157817 39088169 632
45986 102334155 165580141 267914296 433494437 701408733 1134903170 1
836311903 2971215073 4807526976 7778742049 12586269025 20365011074 3
2951280099 53316291173 86267571272 139583862445 225851433717 3654352
96162 591286729879 956722026041 1548008755920 2504730781961 40527395
37881 6557470319842 10610209857723 17167680177565 27777890035288 449
45570212853 72723460248141 117669030460994 190392490709135 308061521
170129 498454011879264 806515533049393 1304969544928657 211148507797
8050 3416454622906707 5527939700884757 8944394323791464 144723340246
76221 23416728348467685 37889062373143906 61305790721611591 99194853
094755497 160500643816367088 259695496911122585 420196140727489673 6
79891637638612258 1100087778366101931 1779979416004714189 2880067194
370816120 4660046610375530309 7540113804746346429 122001604151218767
38 1293530146158671551 13493690561280548289 14787220707439219840 983
4167195010216513 61746438287398847 16008811023750101250
Time taken(in ns): 36608ubuntu@ubuntu:~/Desktop/COA$
```

- Time Taken (in ns): 36608 = 0.000037 s

Q2. Matrix Multiplication

Bucket 1: C++ Language

i) Data type : double

- **Size of Matrix :** 32 x 32

```
Meat Portion for N = 32,Time taken(in ns)= 126959
real    0m0.003s
user    0m0.003s
sys     0m0.000s
```

a) → **User Time :** 0.003 s
→ **System Time :** 0.000 s
→ **CPU Time :** System Time + User Time = 0.003 s

b) → **Odds of Meat Portion w.r.t Total Execution Time(CPU Time) :**(0.000126959/0.003) = 4.23 %

- **Size of Matrix :** 64 x 64

```
Meat Portion for N = 64,Time taken(in ns)= 1021235
real    0m0.015s
user    0m0.006s
sys     0m0.000s
```

a) → **User Time :** 0.006 s
→ **System Time :** 0.000 s
→ **CPU Time :** System Time + User Time = 0.006 s

b) → **Odds of Meat Portion w.r.t Total Execution Time(CPU Time) :**(0.001021235/0.006) = 17.02 %

- **Size of Matrix :** 128 x 128

```
Meat Portion for N = 128,Time taken(in ns)= 8322469
real    0m0.068s
user    0m0.029s
sys     0m0.006s
```

a) → **User Time :** 0.029 s
→ **System Time :** 0.006 s
→ **CPU Time :** System Time + User Time = 0.035 s

b) → **Odds of Meat Portion w.r.t Total Execution Time(CPU Time) :**(0.008322469/0.035) = 23.78 %

- **Size of Matrix :** 256 x 256

```
Meat Portion for N = 256,Time taken(in ns)= 67349471
real    0m0.184s
user    0m0.124s
sys     0m0.000s
```

a) → **User Time :** 0.124 s
→ **System Time :** 0.000 s
→ **CPU Time :** System Time + User Time = 0.124 s

b) → **Odds of Meat Portion w.r.t Total Execution Time(CPU Time) :**(0.067349471/0.124) = 54.31 %

- **Size of Matrix :** 512 x 512

```
Meat Portion for N = 512,Time taken(in ns)= 586438942
real    0m1.055s
user    0m0.800s
sys     0m0.013s
```

a) → **User Time :** 0.800 s
→ **System Time :** 0.013 s
→ **CPU Time :** System Time + User Time = 0.813 s

b) → **Odds of Meat Portion w.r.t Total Execution Time(CPU Time) :**(0.586438942/0.813) = 72.13 %

ii) <u>Data type</u> : int

- **Size of Matrix :** 32 x 32

```
Meat Portion for N = 32 Time taken(in ns)= 101891
real    0m0.002s
user    0m0.000s
sys     0m0.002s
```

a) → **User Time :** 0.000 s
→ **System Time :** 0.002 s
→ **CPU Time :** System Time + User Time = 0.002 s

b) → **Odds of Meat Portion w.r.t Total Execution Time(CPU Time) :**(0.000101891/0.002) = 5.09 %

- **Size of Matrix :** 64 x 64

```
Meat Portion for N = 64 Time taken(in ns)= 939162
real    0m0.011s
user    0m0.004s
sys     0m0.000s
```

a) → **User Time :** 0.004 s
→ **System Time :** 0.000 s
→ **CPU Time :** System Time + User Time = 0.004 s

b) → **Odds of Meat Portion w.r.t Total Execution Time(CPU Time) :**(0.000939162/0.004) = 23.48 %

- **Size of Matrix :** 128 x 128

```
Meat Portion for N = 128,Time taken(in ns)= 8322469
real    0m0.068s
user    0m0.029s
sys     0m0.006s
```

a) → **User Time :** 0.029 s
→ **System Time :** 0.006 s
→ **CPU Time :** System Time + User Time = 0.035 s

b) → **Odds of Meat Portion w.r.t Total Execution Time(CPU Time) :**(0.008322469/0.035) = 23.78 %

- **Size of Matrix :** 256 x 256

```
Meat Portion for N = 256,Time taken(in ns)= 67349471
real    0m0.184s
user    0m0.124s
sys     0m0.000s
```

a) → **User Time :** 0.124 s
→ **System Time :** 0.000 s
→ **CPU Time :** System Time + User Time = 0.124 s

b) → **Odds of Meat Portion w.r.t Total Execution Time(CPU Time) :**(0.067349471/0.124) = 54.31 %

- **Size of Matrix :** 512 x 512

```
Meat Portion for N = 512 Time taken(in ns)= 483170654
real    0m0.802s
user    0m0.553s
sys     0m0.015s
```

a) → **User Time :** 0.553 s
→ **System Time :** 0.015 s
→ **CPU Time :** System Time + User Time = 0.568 s

b) → **Odds of Meat Portion w.r.t Total Execution Time(CPU Time) :**(0.483170654/0.568) = 85.06 %

Bucket 2: Python Language

i) Data type : double

- **Size of Matrix :** 32 x 32

```
ubuntu@ubuntu:~/Desktop/COA/Q2$ time python3 matmuldbl.py
Meet Portion Time:  0.0181734561920166

real    0m0.042s
user    0m0.034s
sys     0m0.008s
```

a) → **User Time :** 0.034 s
→ **System Time :** 0.008 s
→ **CPU Time :** System Time + User Time = 0.042 s

b) → **Odds of Meat Portion w.r.t Total Execution Time(CPU Time) :**(0.018173456/0.042) = 43.27 %

- **Size of Matrix :** 64 x 64

```
ubuntu@ubuntu:~/Desktop/COA/Q2$ time python3 matmuldbl.py
Meet Portion Time:  0.14359259605407715

real    0m0.185s
user    0m0.172s
sys     0m0.013s
```

a) → **User Time :** 0.172 s
→ **System Time :** 0.013 s
→ **CPU Time :** System Time + User Time = 0.185 s

b) → **Odds of Meat Portion w.r.t Total Execution Time(CPU Time) :**(0.143592596/0.185) = 77.61 %

- **Size of Matrix :** 128 x 128

```
ubuntu@ubuntu:~/Desktop/COA/Q2$ time python3 matmuldbl.py
Meet Portion Time:  1.2131543159484863

real    0m1.285s
user    0m1.284s
sys     0m0.000s
```

a) → **User Time :** 1.284 s
→ **System Time :** 0.000 s
→ **CPU Time :** System Time + User Time = 1.284 s

b) → **Odds of Meat Portion w.r.t Total Execution Time(CPU Time) :**(1.213154316/1.284) = 94.48 %

- **Size of Matrix :** 256 x 256

```
ubuntu@ubuntu:~/Desktop/COA/Q2$ time python3 matmuldbl.py
Meet Portion Time:  8.793598175048828

real    0m9.029s
user    0m9.024s
sys     0m0.005s
```

a) → **User Time :** 9.024 s
→ **System Time :** 0.005 s
→ **CPU Time :** System Time + User Time = 9.029 s

b) → **Odds of Meat Portion w.r.t Total Execution Time(CPU Time) :**(8.793598175/9.029) = 97.39 %

- **Size of Matrix :** 512 x 512

```
ubuntu@ubuntu:~/Desktop/COA/Q2$ time python3 matmuldbl.py
Meet Portion Time:   72.25014901161194

real    1m13.100s
user    1m13.092s
sys     0m0.000s
```

a) → **User Time :** 1m 13.092 s
→ **System Time :** 0.000 s
→ **CPU Time :** System Time + User Time = 73.092 s

b) → **Odds of Meat Portion w.r.t Total Execution Time(CPU Time) :**(72.25014902/73.092) = 98.85 %

ii) Data type : int

- **Size of Matrix :** 32 x 32

```
ubuntu@ubuntu:~/Desktop/COA/Q2$ time python3 matmulint.py
Meet Potion Time:   0.02436089515686035

real    0m0.047s
user    0m0.039s
sys     0m0.008s
```

a) → **User Time :** 0.039 s
→ **System Time :** 0.008 s
→ **CPU Time :** System Time + User Time = 0.047 s

b) → **Odds of Meat Portion w.r.t Total Execution Time(CPU Time) :**(0.024360895/0.047) = 51.83 %

- **Size of Matrix :** 64 x 64

```
ubuntu@ubuntu:~/Desktop/COA/Q2$ time python3 matmulint.py
Meet Potion Time:   0.1445293426513672

real    0m0.175s
user    0m0.174s
sys     0m0.000s
```

a) → **User Time :** 0.174 s
→ **System Time :** 0.000 s
→ **CPU Time :** System Time + User Time = 0.174 s

b) → **Odds of Meat Portion w.r.t Total Execution Time(CPU Time) :**(0.144529343/0.174) = 83.06 %

- **Size of Matrix :** 128 x 128

```
ubuntu@ubuntu:~/Desktop/COA/Q2$ time python3 matmulint.py
Meet Potion Time:  1.1371369361877441

real    0m1.209s
user    0m1.203s
sys     0m0.004s
```

a) → **User Time :** 1.203 s
→ **System Time :** 0.000 s
→ **CPU Time :** System Time + User Time = 1.203 s

b) → **Odds of Meat Portion w.r.t Total Execution Time(CPU Time) :**(1.137136936/1.203) = 94.52 %

- **Size of Matrix :** 256 x 256

```
ubuntu@ubuntu:~/Desktop/COA/Q2$ time python3 matmulint.py
Meet Potion Time:  8.948956727981567

real    0m9.170s
user    0m9.162s
sys     0m0.004s
```

a) → **User Time :** 9.162 s
→ **System Time :** 0.004 s
→ **CPU Time :** System Time + User Time = 9.166 s

b) → **Odds of Meat Portion w.r.t Total Execution Time(CPU Time) :**(8.948956728/9.166) = 97.63 %

- **Size of Matrix :** 512 x 512

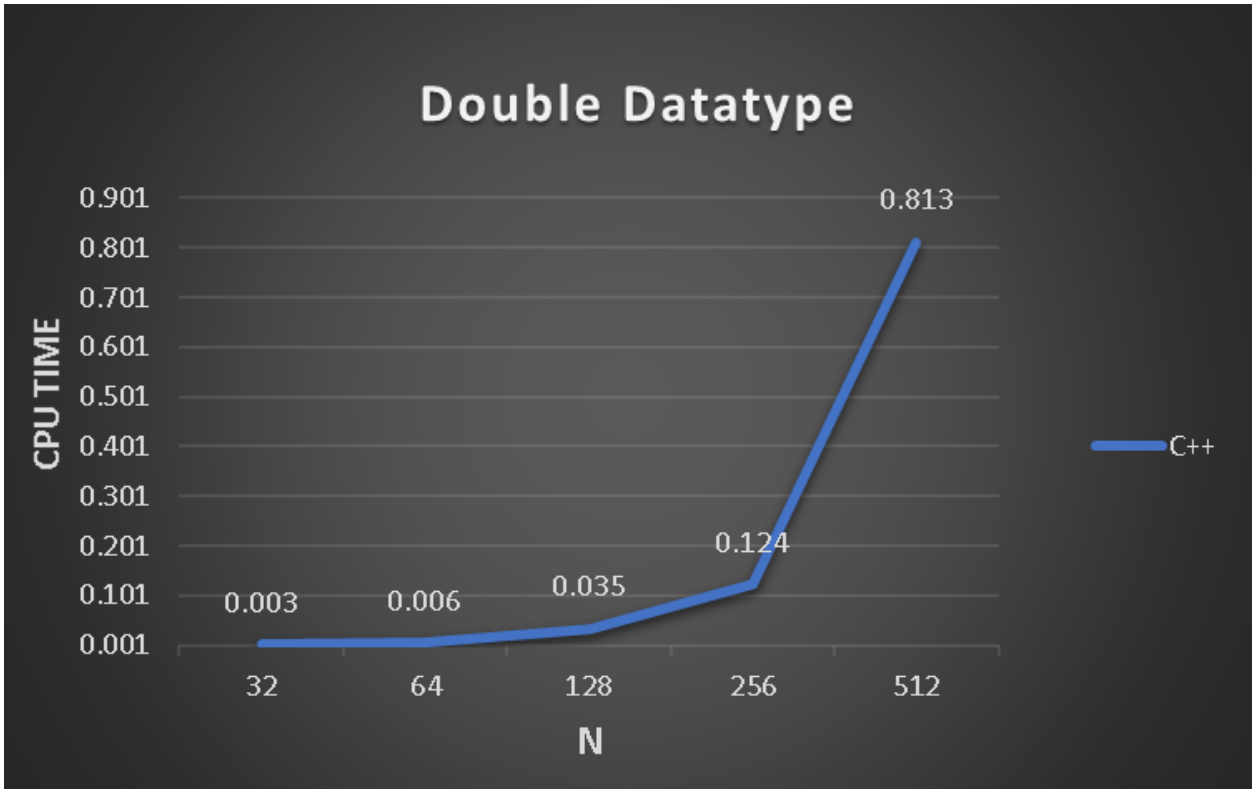
```
ubuntu@ubuntu:~/Desktop/COA/Q2$ time python3 matmulint.py
Meet Potion Time:  73.97407293319702

real    1m14.842s
user    1m14.830s
sys     0m0.008s
```

a) → **User Time :** 1m 14.830 s
→ **System Time :** 0.008 s
→ **CPU Time :** System Time + User Time = 74.838 s

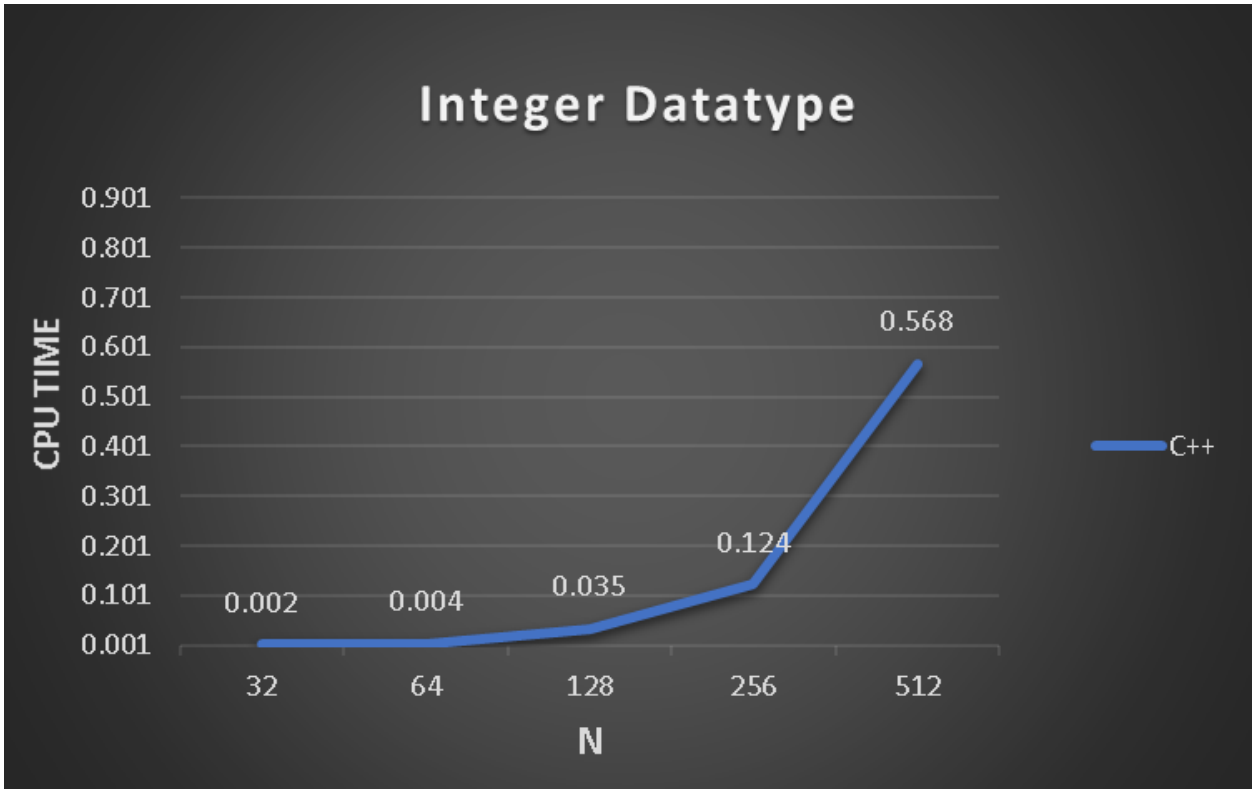
b) → **Odds of Meat Portion w.r.t Total Execution Time(CPU Time) :**(73.97407293/74.838) = 98.84 %

C) **<u>Bucket 1</u> : C++**

~ The below graph depicts the variation of Program Execution Time with N(Size of Matrix) for **Double** data types.

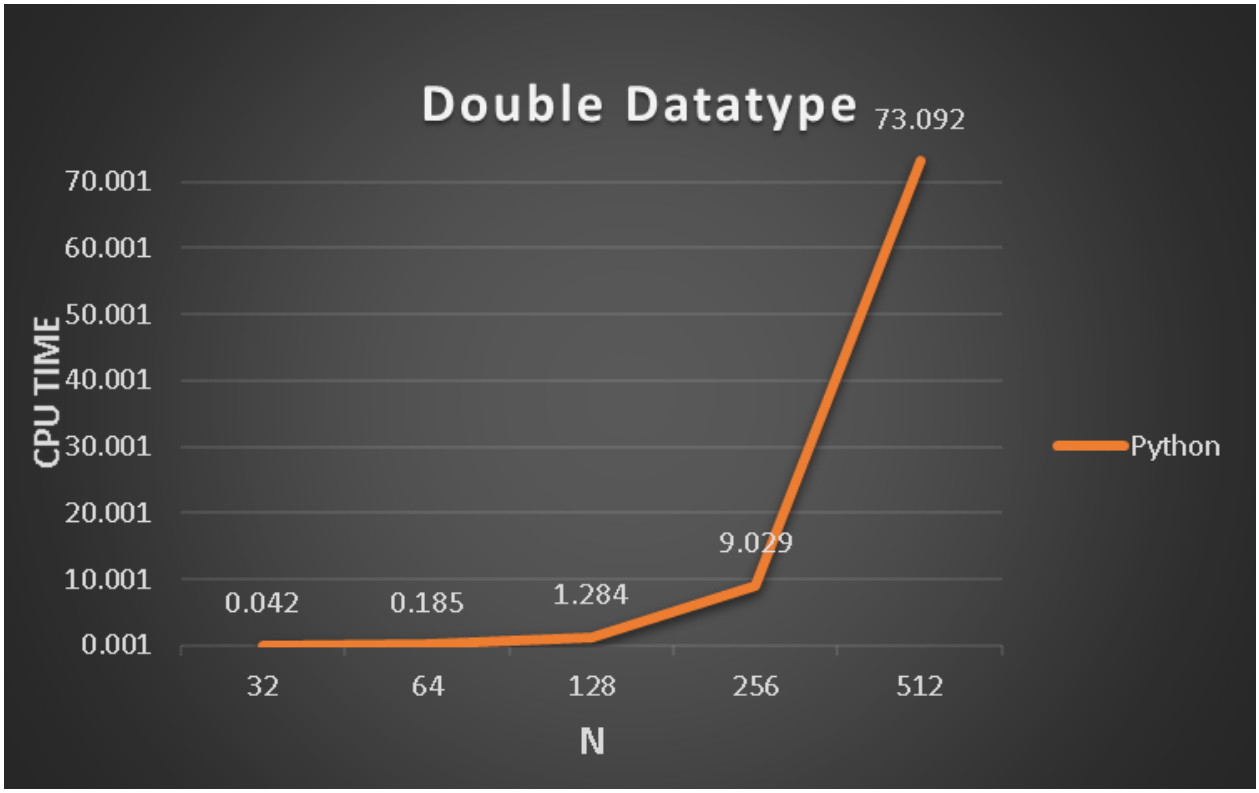


~ The below graph depicts the variation of Program Execution Time with N(Size of Matrix) for **Integer** data types.

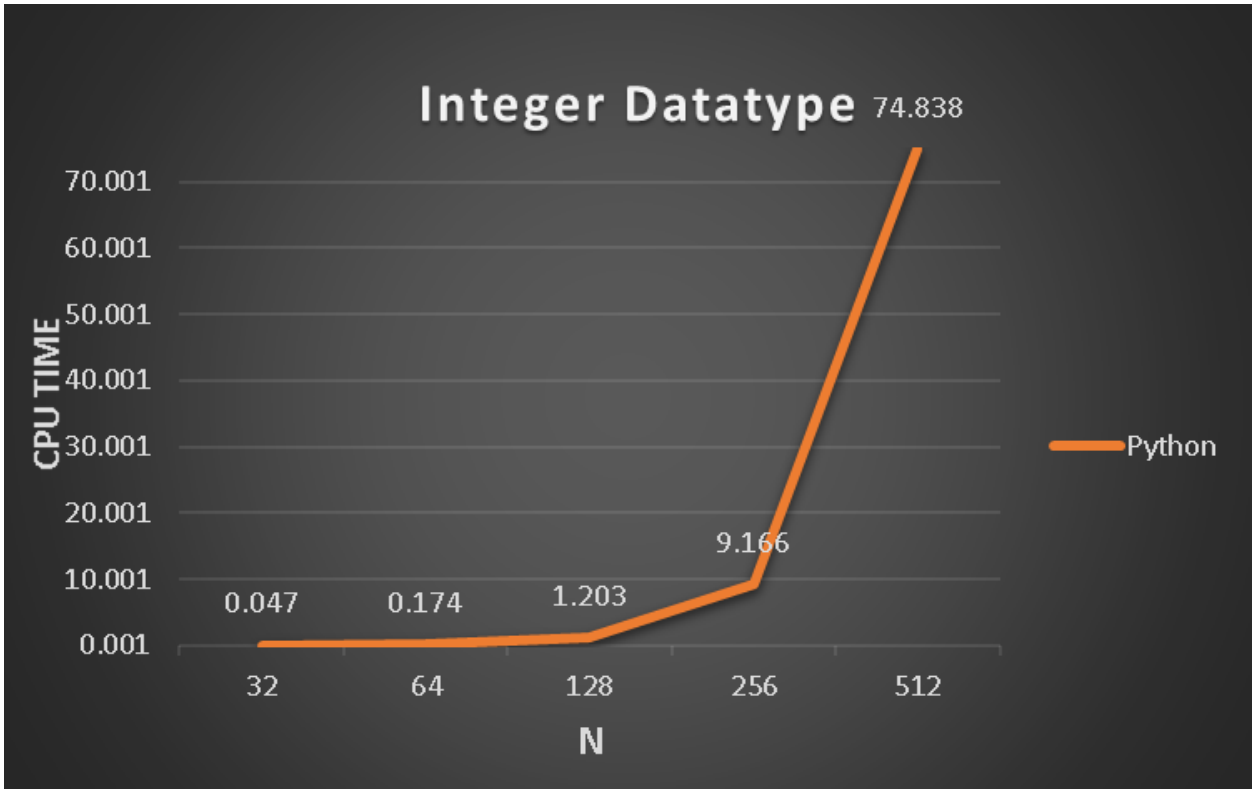


<u>Bucket 2</u> : Python

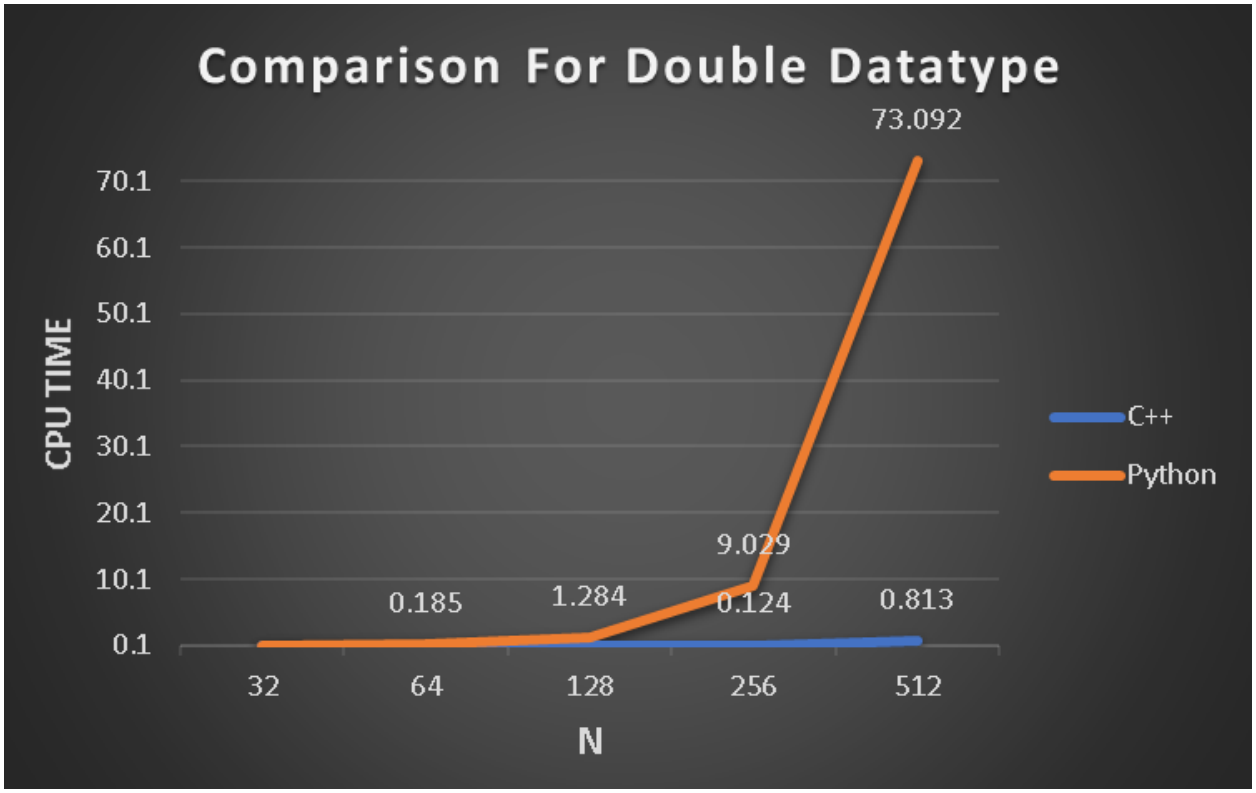
~ The below graph depicts the variation of Program Execution Time with N(Size of Matrix) for **Double** data types.



~ The below graph depicts the variation of Program Execution Time with N(Size of Matrix) for **Integer** data types.

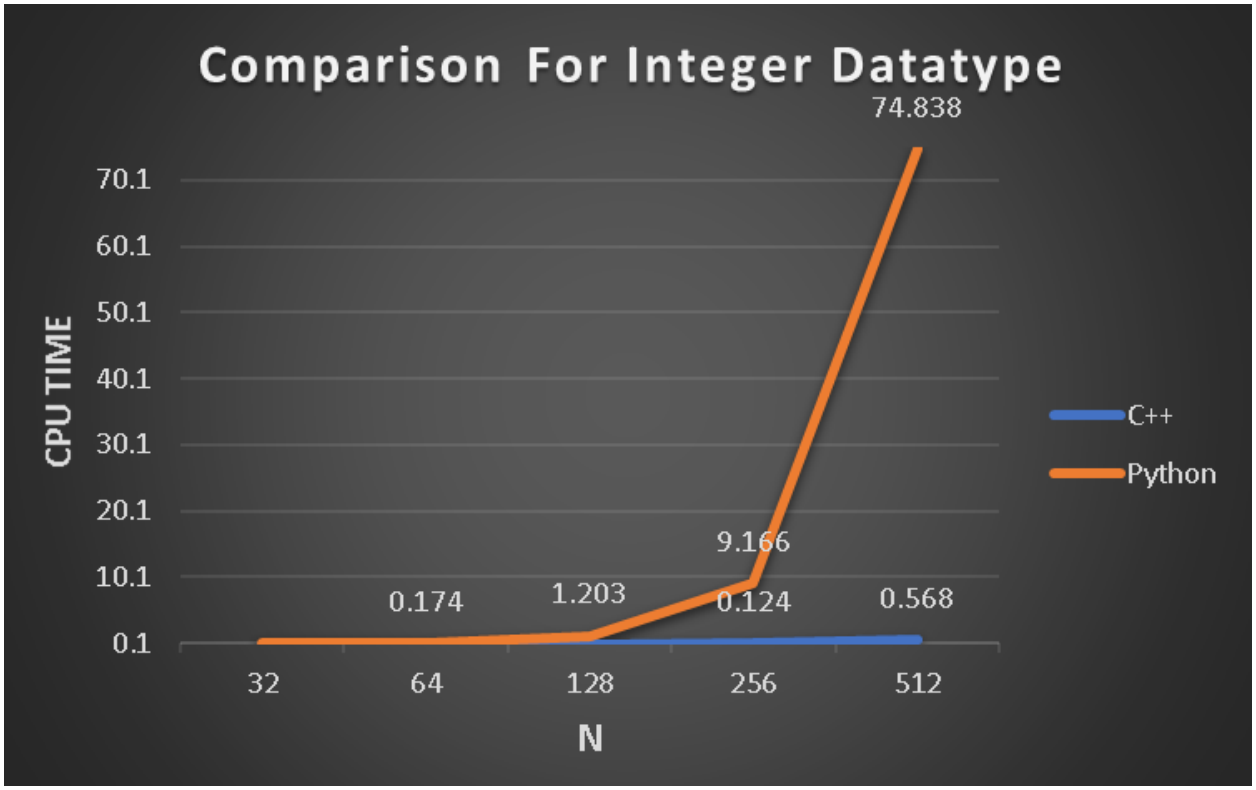


Comparison :



→ Here, I have compared the graph between two languages for a matrix of the same data type(**DOUBLE**). In which, we can see that Python is horribly taking a huge time on increasing the size of the matrix whereas C++ is taking almost same time within 1 second.

→ It was observed that in a python program, the display time of the output was greater than that in C++ program for the same data type.



→ In this graph, I have compared the graph between two languages for a matrix of the same data type(**INTEGER**). In the above graph, the same observation which was made in the above one proved correct.

→ During execution of both programs and comparing the time between two same languages program for two different data type then, In case of

i) Bucket 1(C++): It was observed that in C++ language, it was taking more time to execute for double data type than integer data type as we increased the size of the matrix.

ii) Bucket 2(PYTHON): It was observed that in Python language, it was just opposite to C++ language that is Integer data type is taking more time than double data type which confused me a little, because it is logical/rational to think that double data type contains decimal points also behind the number so it would might be taking more time in calculating the matrix multiplication, But the logic wasn't worked. It is a different problem which might come under the microstructure of computer architecture.

The End