**Name:** Prakram Rathore
**Roll No.** : 20110141

# ES215

# Computer Architecture and Organization

## Assignment - 1

**Q1. Print first 100 Fibonacci numbers, analyze execution time of the program, and calculate the SpeedUp with respect to program (a).**
**ANS.**

a) **Recursive Code -**

- I tried executing the recursive program to find 100 Fibonacci numbers but I didn't get all 100 numbers as it was taking an indefinitely long time. So, I've used the concept of Time complexity to find the time that my program will take.

- The time complexity to find 100 Fibonacci numbers: O(1.618^n)

- So, We can find the actual time taken by formula Time: T(n) = K*(1.618^n), where K is the system constant which will be the same for every iteration.

- We can find K by time T(40), T(45), and T(50) and take their average.

- Time taken for:

```
Time taken(in ns): 172244834560
real    2m52.256s
user    2m45.900s
sys     0m0.195s
```

  - N = 50 => 172.24483s

```
Time taken(in ns): 19364854784
real    0m19.370s
user    0m17.740s
sys     0m0.091s
```

  - N = 45 => 19.36485s

```
Time taken(in ns): 2596811008
real    0m2.601s
user    0m1.711s
sys     0m0.008s
```

  - N = 40 => 2.59681s

- ➢ Finding K = 6.12661e-9
- ➢ Hence, T(100) = 4.8425276e12s

Optimization- We are not able to find 100 Fibonacci numbers, so instead of finding 100, we can find the first 50 Fibonacci numbers in all 4 programs, and then we can find speedup with respect to program (a).

NOTE: I'm taking both cases into consideration when N = 50 and 100.

Hence T(50) = 172.24483s,
SpeedUp = 172.24483/172.24483 = 1.

b) Loop Code -

Time -

Case - 1 (N = 50)

```
Time taken(in ns): 60672
real    0m0.010s
user    0m0.000s
sys     0m0.004s
```

T(50) = 0.000060672s
SpeedUp = 172.24483/0.000060672 = 2.838e6

Case - 2 (N = 100)

```
Time taken(in ns): 450816
real    0m0.007s
user    0m0.000s
sys     0m0.005s
```

T(100) = 0.000450816
SpeedUp = 4.8425276e12/0.000450816 = 1.07416942e16

c) Recursion with memoization -

Time -

Case - 1 (N = 50)

```
Time taken(in ns): 43264
real    0m0.008s
user    0m0.005s
sys     0m0.000s
```

T(50) = 0.000043264s
SpeedUp = 172.24483/0.000043264 = 3.981e6

Case - 2 (N = 100)

```
Time taken(in ns): 83456
real    0m0.007s
user    0m0.000s
sys     0m0.004s
```

T(100) = 0.000083456
SpeedUp = 4.8425276e12/0.000083456 = 5.80249185e16

d) Loop with memoization -
Time -
Case - 1 (N = 50)

```
Time taken(in ns): 50688
real    0m0.006s
user    0m0.003s
sys     0m0.001s
```

T(50) = 0.000050688s
SpeedUp = 172.24483/0.000050688 = 3.981e6

Case - 2 (N = 100)

```
Time taken(in ns): 299264
real    0m0.011s
user    0m0.000s
sys     0m0.008s
```

T(100) = 0.000299264
SpeedUp = 4.8425276e12/0.000299264 = 1.61814572e16

Further Optimization - We can also optimize our recursion through following way -

```
Void  fib(n1, n2){
        if(n1>=100){
            return
        }
        print(n1)
        temp = n2
        n2 = n2 + n1
        n1 = temp
}

fib(0,1)
```

Time:

```
rishi@rishi-VirtualBox:~/rishi$ g++ fib_recursion_optimize.cpp
rishi@rishi-VirtualBox:~/rishi$ time ./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 1346269 2178309 3524578 5702887 9227465 14930352 24157817 390881
69 63245986 102334155 165580141 267914296 433494437 701408733 1134903170 1836311903 -1323752223 512
559680 -811192543 -298632863 -1109825406 -1408458269 1776683621 368225352 2144908973 -1781832971 36
3076002 -1418756969 -1055680967 1820529360 764848393 -1709589543 -944741150 1640636603 695895453 -1
958435240 -1262539787 1073992269 -188547518 885444751 696897233 1582341984 -2015728079 -433386095 1
845853122 1412467027 -1036647147 375819880 -660827267 -285007387 -945834654 -1230842041 2118290601
887448560 -1289228135 -401779575 -1691007710 -2092787285 511172301 -1581614984 -1070442683 16429096
29 572466946 -2079590721 -1507123775 708252800 -798870975 -90618175 -889489150
Time taken(in ns): 192000
real    0m0.011s
user    0m0.004s
sys     0m0.003s
```

For optimized recursion code T(100) = 0.000192s,
Note: We are getting some negative numbers, which is due to the maximum size of integer number allowed, but it does not affect our execution time much.

a) Optimized Recursion - SpeedUp = 1

b) Loop Code -

T(100) = 0.000450816

SpeedUp = 0.000192/0.000450816 = 0.42589

c) Recursion with memoization - Note: This recursion is not an optimized one, it uses tree recursion.

T(100) = 0.000083456

SpeedUp = 0.000192/0.000083456 = 2.3006135

d) Loop with memoization -

T(100) = 0.000299264

SpeedUp = 0.000192/0.000299264 = 0.641573995

**Q2. Matrix Multiplication program for N = 32, 64, 128, 256, 512 for Python and C++. Compare their execution time.**
**Ans.**

Note: Each time, I ran the code and noted down the various time metrics, I observed that time fluctuates every time and it depends on the computer system and its current state. I observed major fluctuations in the percentage of execution time for the meat portion with respect to total execution time, sometimes more than 50%. So, as mentioned in class, I ran each program an odd number of times and took their median for the final calculator. I also observed that time.time() function in python which I used to find the execution time for the meat portion of the program is less efficient than time and time-spec common in Linux, thus in some cases, the percentage is greater than 100%.

A) Data Type = Double

Bucket - 1 => C++

1. N = 32

```
Meat Portion for N = 32 Time taken(in ns)= 173200
real    0m0.003s
user    0m0.003s
sys     0m0.000s
```

a)

- ➔ User time: 0.003s
- ➔ System time: 0.000s
- ➔ CPU time: User time + System time = 0.003 + 0.000 = 0.003s

b) Execution time for the meat portion of program = 173200ns = 0.000173200s
Total execution time = CPU time = 0.003s
Percent = 5.773%

2. N = 64

```
Meat Portion for N = 64 Time taken(in ns)= 1402016
real    0m0.017s
user    0m0.006s
sys     0m0.001s
```

a)

- ➔ User time: 0.006s

➔ System time: 0.001s
➔ CPU time: User time + System time = 0.006 + 0.001 = 0.007s

b) Execution time for the meat portion of program = 1402016ns = 0.001402016s
Total execution time = CPU time = 0.007s
Percent = 20.028%

3. N = 128

```
Meat Portion for N = 128 Time taken(in ns)= 8215012
real    0m0.042s
user    0m0.014s
sys     0m0.005s
```

a)

➔ User time: 0.014s
➔ System time: 0.005s
➔ CPU time: User time + System time = 0.014 + 0.005 = 0.019s

b) Execution time for the meat portion of program = 8215012ns = 0.008215012s
Total execution time = CPU time = 0.019s
Percent = 43.236%

4. N = 256

```
Meat Portion for N = 256 Time taken(in ns)= 83780693
real    0m0.199s
user    0m0.109s
sys     0m0.004s
```

a)

➔ User time: 0.109s
➔ System time: 0.004s
➔ CPU time: User time + System time = 0.109 + 0.004 = 0.113s

b) Execution time for the meat portion of program = 83780693ns = 0.083780693s
Total execution time = CPU time = 0.113s
Percent = 74.14%

5. N = 512

```
Meat Portion for N = 512 Time taken(in ns)= 713770840
real    0m1.205s
user    0m0.785s
sys     0m0.034s
```

a)

- ➔ User time: 0.785s
- ➔ System time: 0.034s
- ➔ CPU time: User time + System time = 0.785 + 0.034 = 0.819s

b) Execution time for the meat portion of program = 713770840ns = 0.71377084s
Total execution time = CPU time = 0.819s
Percent = 87.151%

Bucket - 2 => Python

1. N = 32

```
Meet Portion Time:  0.010942459106445312

real    0m0.032s
user    0m0.026s
sys     0m0.000s
```

a)

- ➔ User time: 0.026s
- ➔ System time: 0.000s
- ➔ CPU time: User time + System time = 0.026 + 0.000 = 0.026s

b) Execution time for the meat portion of program = 0.01094s
Total execution time = CPU time = 0.026s
Percent = 42.076%

2. N = 64

```
Meet Portion Time:  0.05324864387512207

real    0m0.104s
user    0m0.071s
sys     0m0.011s
```

a)

- ➔ User time: 0.071s
- ➔ System time: 0.011s
- ➔ CPU time: User time + System time = 0.071 + 0.011 = 0.082s

b) Execution time for the meat portion of program = 0.05324s
Total execution time = CPU time = 0.082s
Percent = 64.926%

3. N = 128

```
Meet Portion Time:  0.43274760246276855

real    0m0.552s
user    0m0.502s
sys     0m0.004s
```

a)

- ➔ User time: 0.502s
- ➔ System time: 0.004s
- ➔ CPU time: User time + System time = 0.502 + 0.004 = 0.504s

b) Execution time for the meat portion of program = 0.43274s
Total execution time = CPU time = 0.504s
Percent = 85.861%

4. N = 256

```
Meet Portion Time:  3.674177646636963

real    0m4.063s
user    0m3.760s
sys     0m0.024s
```

a)

- ➔ User time: 3.760s
- ➔ System time: 0.024s
- ➔ CPU time: User time + System time = 3.760 + 0.024= 3.784s

b) Execution time for the meat portion of program = 3.67417s
Total execution time = CPU time = 3.784s
Percent = 97.09%

5. N = 512

```
Meet Portion Time:  27.768494844436646

real    0m29.301s
user    0m27.987s
sys     0m0.056s
```

a)

- ➔ User time: 27.987s
- ➔ System time: 0.056s
- ➔ CPU time: User time + System time = 27.987 + 0.056 = 28.043s

b) Execution time for the meat portion of program = 27.76849s

Total execution time = CPU time = 28.043s
Percent = 99.021%

B) Data Type = Integer

Bucket - 1 => C++

1. N = 32

```
Meat Portion for N = 32 Time taken(in ns)= 101824
real    0m0.002s
user    0m0.002s
sys     0m0.000s
```

a)

- ➔ User time: 0.002s
- ➔ System time: 0.000s
- ➔ CPU time: User time + System time = 0.002 + 0.000 = 0.002s

b) Execution time for the meat portion of program = 101824ns = 0.000101824s
Total execution time = CPU time = 0.002s
Percent = 5.091%

2. N = 64

```
Meat Portion for N = 64 Time taken(in ns)= 779192
real    0m0.018s
user    0m0.001s
sys     0m0.005s
```

a)

- ➔ User time: 0.001s
- ➔ System time: 0.005s
- ➔ CPU time: User time + System time = 0.001 + 0.005 = 0.006s

b) Execution time for the meat portion of program = 779192ns = 0.000779192s
Total execution time = CPU time = 0.006s
Percent = 12.986%

3. N = 128

```
Meat Portion for N = 128 Time taken(in ns)= 15772923
real    0m0.047s
user    0m0.016s
sys     0m0.001s
```

a)

➔ User time: 0.016s
➔ System time: 0.001s
➔ CPU time: User time + System time = 0.016 + 0.001 = 0.017s

b) Execution time for the meat portion of program = 15772923ns = 0.15772923s
Total execution time = CPU time = 0.017s
Percent = 83.840%

4. N = 256

```
Meat Portion for N = 256 Time taken(in ns)= 86689688
real    0m0.296s
user    0m0.081s
sys     0m0.008s
```

a)

➔ User time: 0.081s
➔ System time: 0.008s
➔ CPU time: User time + System time = 0.081 + 0.008 = 0.089s

b) Execution time for the meat portion of program = 86689688ns = 0.086689688s
Total execution time = CPU time = 0.089s
Percent = 97.404%

5. N = 512

```
Meat Portion for N = 512 Time taken(in ns)= 656965514
real    0m0.951s
user    0m0.656s
sys     0m0.024s
```

a)

➔ User time: 0.656s
➔ System time: 0.024s
➔ CPU time: User time + System time = 0.656 + 0.024 = 0.680s

b) Execution time for the meat portion of program = 656965514ns = 0.656965514s
Total execution time = CPU time = 0.680s
Percent = 96.612%

Bucket - 2 => Python

1. N = 32

```
Meet Potion Time:  0.009064912796020508

real    0m0.027s
user    0m0.019s
sys     0m0.007s
```

a)

- ➔ User time: 0.019s
- ➔ System time: 0.007s
- ➔ CPU time: User time + System time = 0.019 + 0.007 = 0.026s

b) Execution time for the meat portion of program = 0.00906s
Total execution time = CPU time = 0.026s
Percent = 34.846%

2. N = 64

```
Meet Potion Time:  0.06253337860107422

real    0m0.089s
user    0m0.069s
sys     0m0.004s
```

a)

- ➔ User time: 0.069s
- ➔ System time: 0.004s
- ➔ CPU time: User time + System time = 0.069 + 0.004 = 0.073s

b) Execution time for the meat portion of program = 0.06253s
Total execution time = CPU time = 0.073s
Percent = 85.657%

3. N = 128

```
Meet Potion Time:  0.43295788764953613

real    0m0.511s
user    0m0.446s
sys     0m0.016s
```

a)

- ➔ User time: 0.446s
- ➔ System time: 0.016s
- ➔ CPU time: User time + System time = 0.446 + 0.016 = 0.462s

b) Execution time for the meat portion of program = 0.43295s
Total execution time = CPU time = 0.462s

Percent = 93.712%

4. N = 256

```
Meet Potion Time:  4.241459131240845

real    0m4.441s
user    0m4.204s
sys     0m0.012s
```

a)

- ➔ User time: 4.204s
- ➔ System time: 0.012s
- ➔ CPU time: User time + System time = 4.204 + 0.012 = 4.216s

b) Execution time for the meat portion of program = 4.24145s
Total execution time = CPU time = 4.216s
Percent = 97.932%

5. N = 512

```
Meet Potion Time:  31.539687633514404

real    0m31.912s
user    0m31.199s
sys     0m0.016s
```

a)

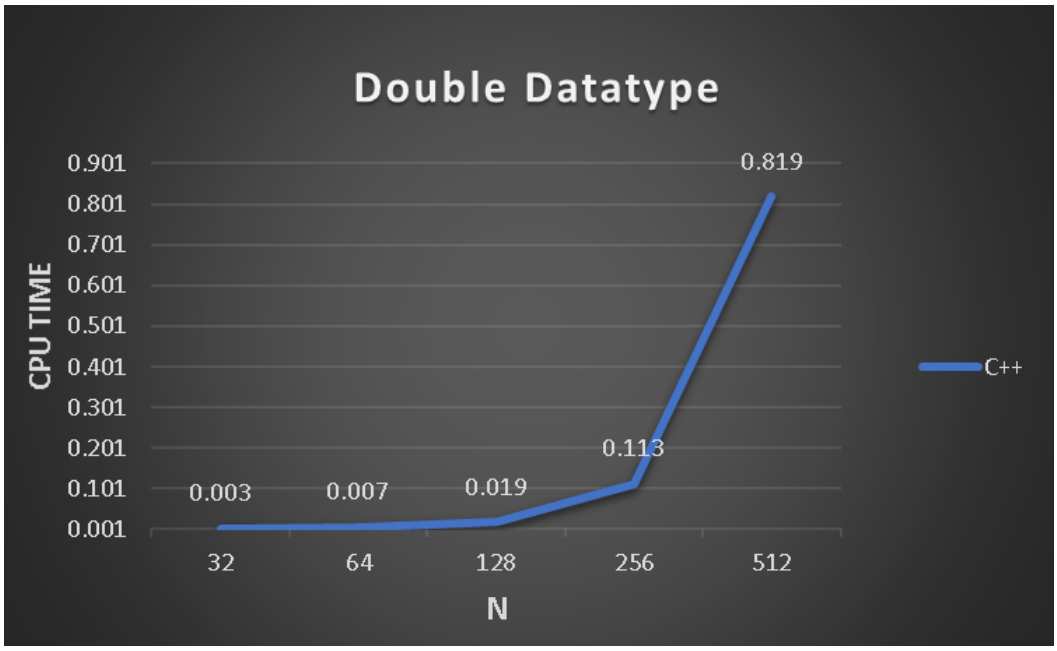
- ➔ User time: 31.199s
- ➔ System time: 0.016s
- ➔ CPU time: User time + System time = 31.199 + 0.016 = 31.215s

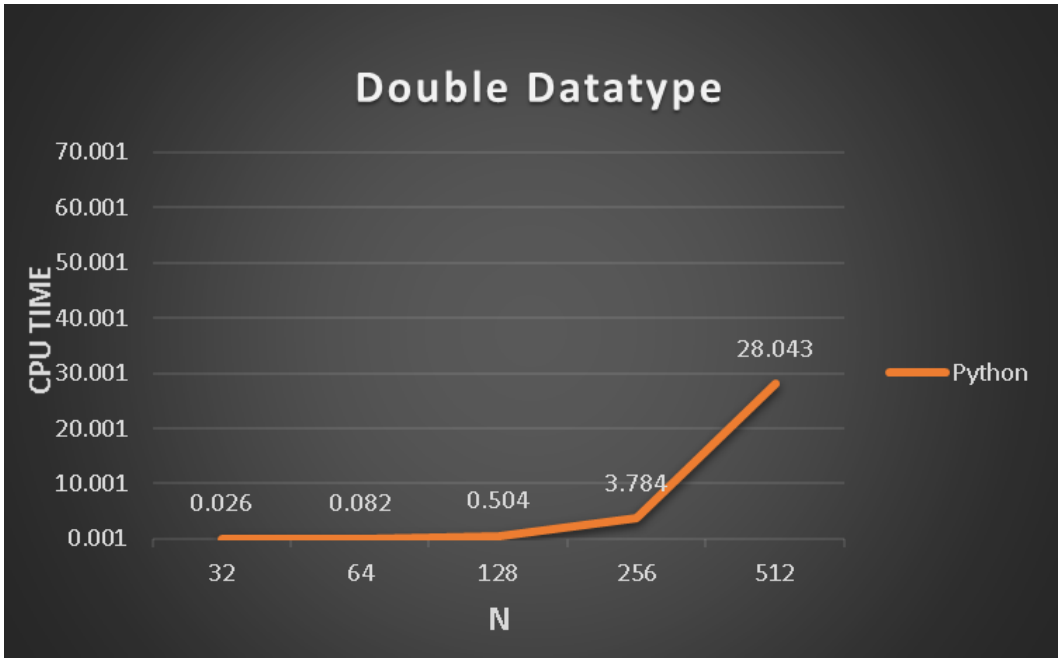
b) Execution time for the meat portion of program = 31.53968s
Total execution time = CPU time = 31.215s
Percent = 100% (Slightly greater than 100)

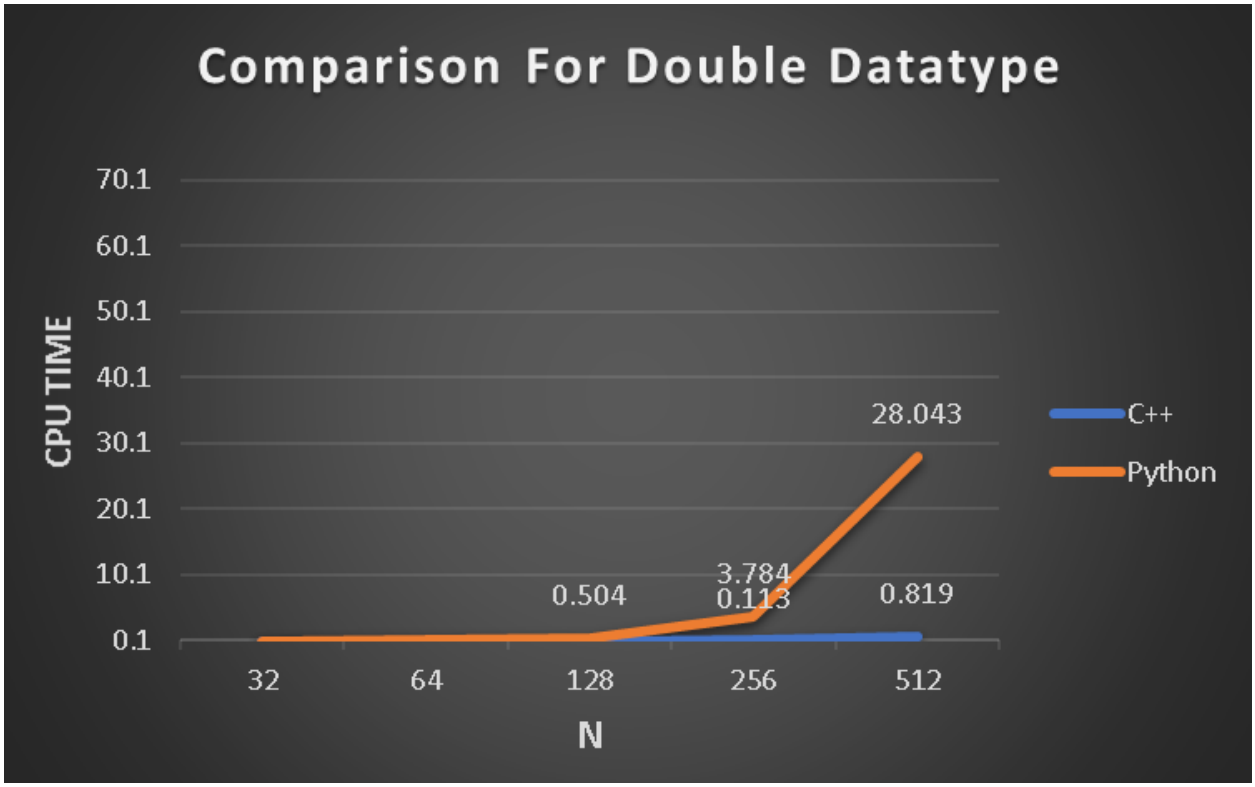
**P.T.O** (Please check next page)

COMPARISON THROUGH PLOT

A) DOUBLE C++ Vs PYTHON





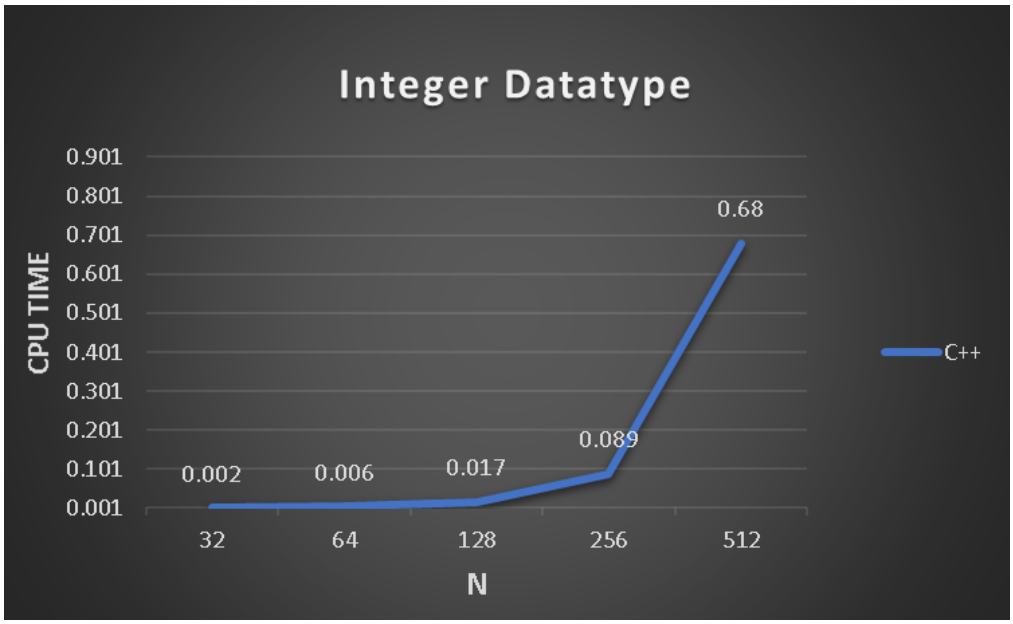

Observations -

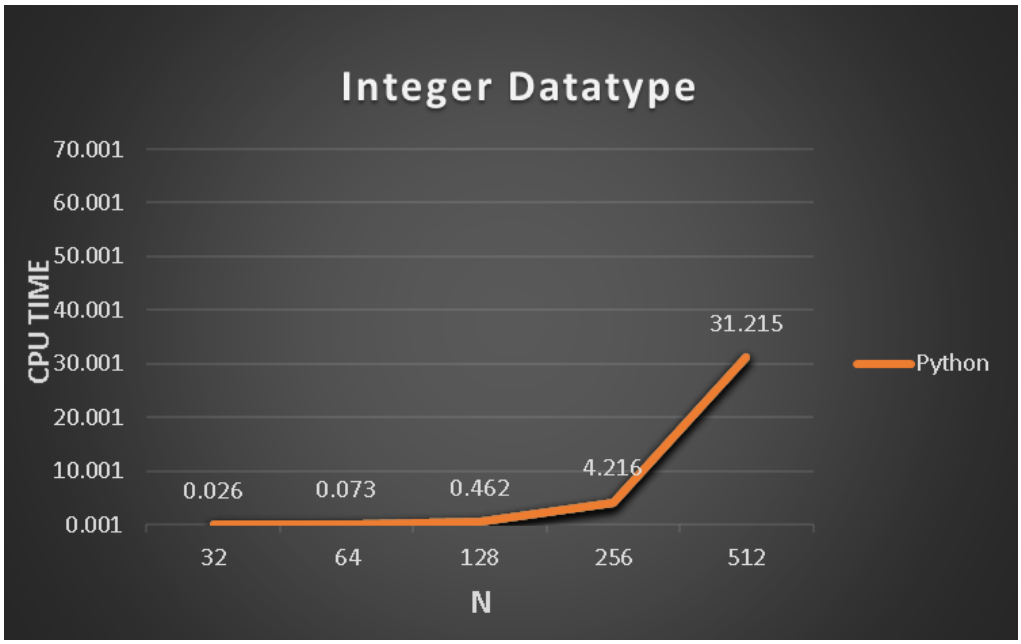
1. For lower values of N, both are approximately equally efficient, but as the value of N, increase python takes a huge amount of time as compared to C++. As we can see at N = 512, C++ is taking 0.819 sec but python is taking 28.043 sec.
2. Total execution time in the case of C++ remains less than one for all values till 512 but in the case of Python, it reaches more than 25 sec.

3. It was observed that python prints the final matrix in less time as compared to the output CPU time, this might be due to the fact that python takes more time to exit through compilation.

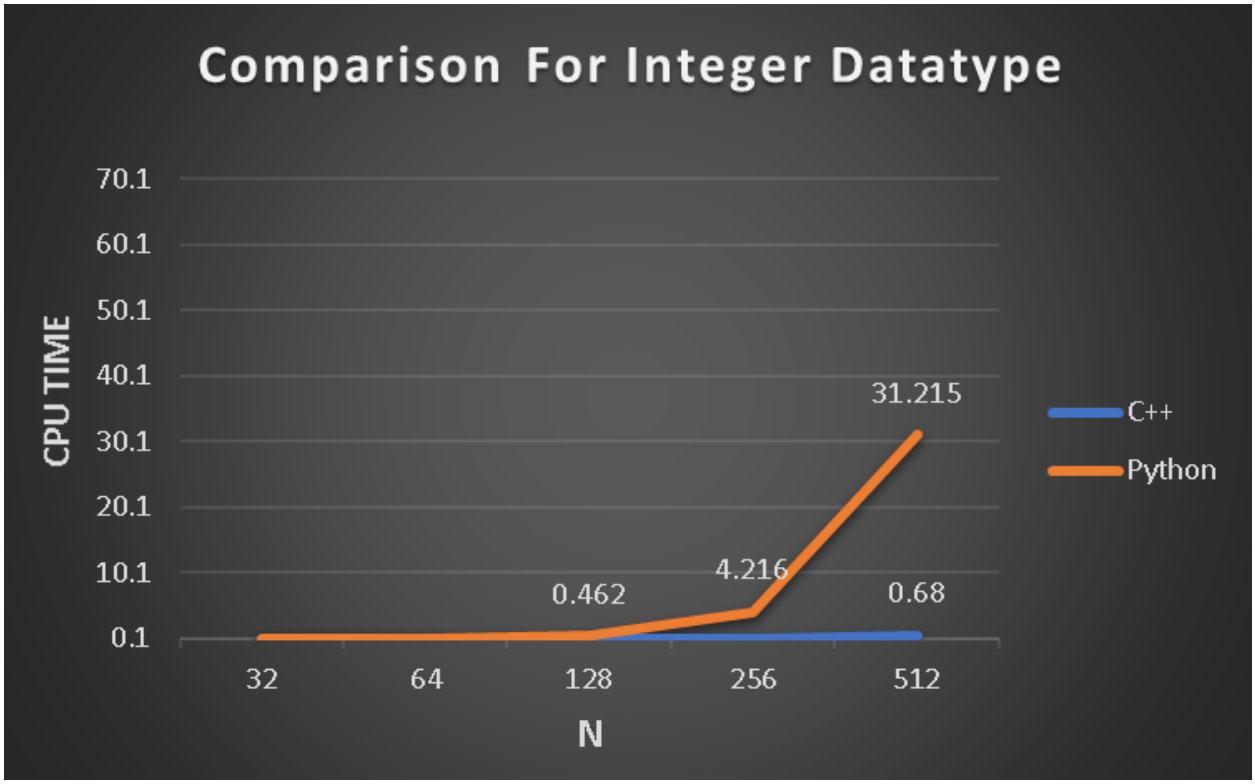
**P.T.O** (Please check next page)

B) INTEGER C++ Vs PYTHON






**P.T.O** (Please check next page)




Observations -

1. For lower values of N, both are approximately equally efficient, but as the value of N, increase python takes a huge amount of time as compared to C++. As we can see at N = 512, C++ is taking 0.68 sec but python is taking 31.215 sec.
2. Total execution time in the case of C++ remains less than one for all values till 512 but in the case of Python, it reaches more than 30 sec.
3. It was observed that python prints the final matrix in less time as compared to the output CPU time, this might be due to the fact that python takes more time to exit through compilation.

COMPARISON BETWEEN C++ IN DATATYPE DOUBLE AND INTEGER -

1. C++ is taking more time to execute the program in the case of doubles.
2. We can conclude that in C++, the compiler takes more time to do operations which has double as the datatype as compared to the integer datatype.

COMPARISON BETWEEN PYTHON IN DATATYPE DOUBLE AND INTEGER -

1. PYTHON is taking more time to execute the program in the case of integers.
2. We can conclude that in PYTHON, the compiler takes more time to do operations which has integer as the datatype as compared to the double datatype.