- **Program Title :-** comparative Performance Analysis of Fibonacci Series and Digital Sum Algorithms across Multiple Programming Languages (C, C++, Java, Python) in Windows and Linux Environments.

- **Student Information :-**

  Name: Raman Kumar

  College ID: 24IT40

  Operating System: Kali Linux (64-bit) on VirtualBox and windows 11 home

- **System info and configuration :-**
- **OS: Kali Linux**

```
PRETTY_NAME="Kali GNU/Linux Rolling"
NAME="Kali GNU/Linux"
VERSION_ID="2025.2"
VERSION="2025.2"
VERSION_CODENAME=kali-rolling
ID=kali
ID_LIKE=debian
HOME_URL="https://www.kali.org/"
SUPPORT_URL="https://forums.kali.org/"
BUG_REPORT_URL="https://bugs.kali.org/"
ANSI_COLOR="1;31"
```

- **Kernel Version:**
  Linux kali 6.12.25-amd64 #1 SMP PREEMPT_DYNAMIC Kali 6.12.25-1kali1 (2025-04-30) x86_64 GNU/Linux

- **System configuration:**

A. Host Machine (Physical Hardware)

Laptop Model: HP Victus 15-fb0108AX

Processor: AMD Ryzen 5 5600H (6 Cores, 12 Threads)

Base Clock Speed: 3.3 GHz (Max Boost up to 4.2 GHz)

Total System RAM: 8 GB DDR4

Operating System: Windows 11 Home (64-bit)

Storage: 512 GB PCIe NVMe M.2 SSD

B. Guest Machine (Virtual Environment via VirtualBox)

The programs were executed on **Kali Linux** running inside **Oracle VM VirtualBox**. The virtual machine (VM) was configured with the following parameters:

Virtualization Software: Oracle VM VirtualBox (Version 7.x)

Operating System: Kali GNU/Linux Rolling (64-bit)

Allocated RAM (Base Memory): 3500 MB (Approx. 3.5 GB)

Processor Cores Allocated: 3 Virtual CPUs

Execution Cap: 100%

Paravirtualization Interface: Default (KVM)

Hardware Virtualization: Enabled (Nested Paging)

Chipset: PIIX3

Pointing Device: USB Tablet

- **Windows**

Host Name: LAPTOP-8LBVM1SK

OS Name: Microsoft Windows 11 Home Single Language

OS Version: 10.0.26200 N/A Build 26200

OS Manufacturer: Microsoft Corporation

OS Configuration: Standalone Workstation

OS Build Type: Multiprocessor Free

Registered Owner: HP

Registered Organization: HP

Product ID: 00356-24752-96997-AAOEM

Original Install Date: 2/28/2026, 11:04:24 AM

System Boot Time: 3/18/2026, 6:02:42 PM

System Manufacturer: HP

System Model: Victus by HP Gaming Laptop 15-fb0xxx

System Type: x64-based PC

Processor(s): 1 Processor(s) Installed.

[01]: AMD64 Family 25 Model 80 Stepping 0 AuthenticAMD ~3301 Mhz

BIOS Version: AMI F.25, 1/9/2026

Windows Directory: C:\WINDOWS

System Directory: C:\WINDOWS\system32

Boot Device: \Device\HarddiskVolume1

System Locale: en-us;English (United States)

Input Locale: 00004009

Time Zone: (UTC+05:30) Chennai, Kolkata, Mumbai, New Delhi

Total Physical Memory: 7,518 MB

Available Physical Memory: 2,706 MB

Virtual Memory: Max Size: 18,782 MB

Virtual Memory: Available: 11,644 MB

Virtual Memory: In Use: 7,138 MB

Page File Location(s): C:\pagefile.sys

Domain: WORKGROUP

Logon Server: \\LAPTOP-8LBVM1SK

- **Screen shots of programs:-**

❖ Kali Linux

1. Single digit sum

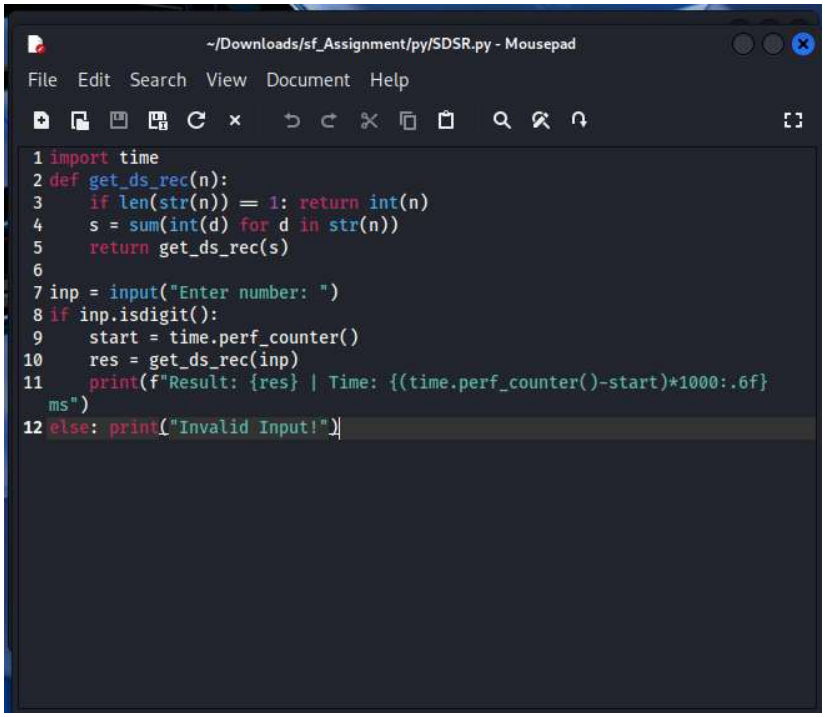
➢ Python

A. Recursive Method

Logic: The function calculates the sum of digits of the number. If the result is greater than 9, the function calls itself again with the new sum as the input.

Process: f(n) = f(sum_of_digits(n)) until n < 10.

Characteristic: It represents the mathematical definition perfectly but uses more "Stack Memory" due to repeated function calls.



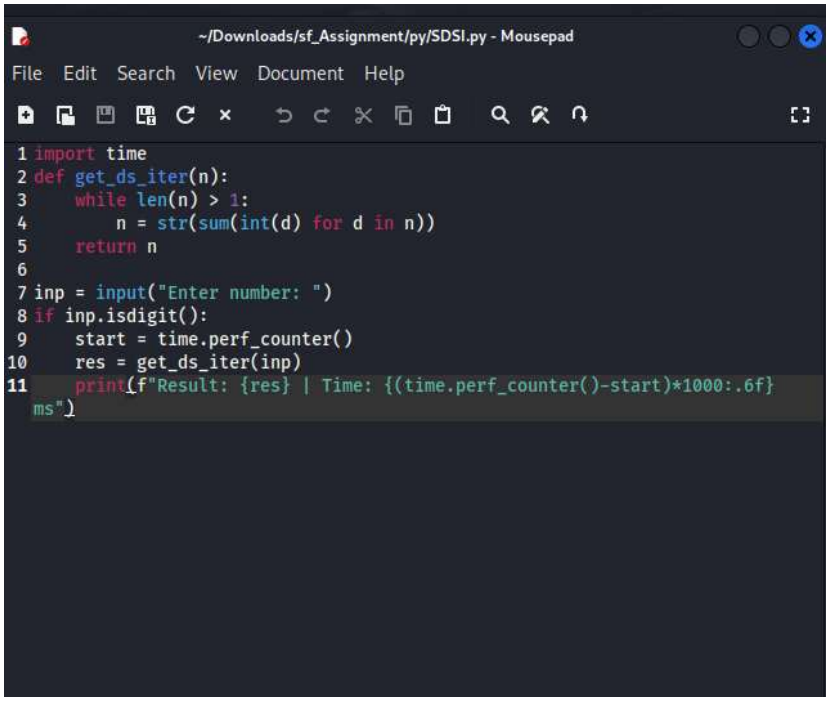
Average time = **0.0555ms**, by recursive method

B. Non-Recursive / Iterative Method

Logic: It uses a while loop to repeatedly sum the digits. The outer loop continues as long as the number has more than one digit.

Process: A nested loop structure where the inner loop extracts digits and the outer loop checks if the final result is a single digit.

Characteristic: This is the most efficient practical approach for standard programming as it avoids memory overhead and is faster than recursion.



Average time = **0.0562ms**, by non recursive method

C. Formula Method

Logic: This method uses a mathematical property of numbers in base 10 (Divisibility by 9).

Formula: Digital Root = 1 + ((n - 1) (mod 9))

Characteristic: This is the **fastest** possible way to find the single digit sum. It executes in constant time (O(1)), meaning it takes the same amount of time regardless of how large the number is.

~/Downloads/sf_Assignment/py/SDSF.py - Mousepad

File Edit Search View Document Help

SDSI.py | SDSF.py

```python
import time
inp = input("Enter number: ")
if inp.isdigit():
    start = time.perf_counter()
    s = sum(int(d) for d in inp)
    res = 0 if s == 0 else 1 + (s - 1) % 9
    print(f"Result: {res} | Time: {(time.perf_counter()-start)*1000:.6f} ms")
```

```
┌──(ramankumar㉿kali)-[~/Downloads/sf_Assignment/py]
└─$ python3 SDSF.py
Enter number: 12345
Result: 6 | Time: 0.063159 ms

┌──(ramankumar㉿kali)-[~/Downloads/sf_Assignment/py]
└─$ python3 SDSF.py
Enter number: 12345
Result: 6 | Time: 0.052083 ms

┌──(ramankumar㉿kali)-[~/Downloads/sf_Assignment/py]
└─$ python3 SDSF.py
Enter number: 12345
Result: 6 | Time: 0.055698 ms

┌──(ramankumar㉿kali)-[~/Downloads/sf_Assignment/py]
└─$ python3 SDSF.py
Enter number: 12345678907789
Result: 4 | Time: 0.060334 ms

┌──(ramankumar㉿kali)-[~/Downloads/sf_Assignment/py]
└─$ python3 SDSF.py
Enter number: 2-5-4-4

┌──(ramankumar㉿kali)-[~/Downloads/sf_Assignment/py]
└─$
```

Average time = **0.0569ms**, by formula method

- **C++**
  - **A. Recursive Method**

Shell No. 1

File Actions Edit View Help

```cpp
#include <iostream>
#include <string>
#include <chrono>
using namespace std;
using namespace std::chrono;

int solve(string n) {
    if (n.length() == 1) return n[0] - '0';
    long long sum = 0;
    for (char c : n)
 sum += (c - '0');
    return solve(to_string(sum));
}
int main() {
    string n; cout << "Enter: "; cin >> n;
for (char c : n) {
if(!isdigit(c)) {
cout<<"Invalid input !"<<endl;
return 0;
}
}
    auto start = high_resolution_clock::now();
    int res = solve(n);
    auto end = high_resolution_clock::now();
    cout << "Res: " << res << " | Time: " << duration_cast<nanoseconds>(end-start).count() /1000000.0 << " ms" << endl;
    return 0;
}
```

```
┌──(ramankumar㉿kali)-[~/Downloads/sf_Assignment/C++]
└─$ g++ ds_rec.cpp -o out && ./out
Enter: 12345
Res: 6 | Time: 0.012257 ms

┌──(ramankumar㉿kali)-[~/Downloads/sf_Assignment/C++]
└─$ g++ ds_rec.cpp -o out && ./out
Enter: 12345
Res: 6 | Time: 0.01421 ms

┌──(ramankumar㉿kali)-[~/Downloads/sf_Assignment/C++]
└─$ g++ ds_rec.cpp -o out && ./out
Enter: 12345
Res: 6 | Time: 0.012408 ms

┌──(ramankumar㉿kali)-[~/Downloads/sf_Assignment/C++]
└─$ g++ ds_rec.cpp -o out && ./out
Enter: 123456789007789
Res: 4 | Time: 0.013428 ms

┌──(ramankumar㉿kali)-[~/Downloads/sf_Assignment/C++]
└─$ g++ ds_rec.cpp -o out && ./out
Enter: 548-96-45
Invalid input !

┌──(ramankumar㉿kali)-[~/Downloads/sf_Assignment/C++]
└─$
```

Average time = **0.01293ms**, by recursive method

B. Non-Recursive / Iterative Method

Shell No. 1

File Actions Edit View Help

```cpp
#include <iostream>
#include <string>
#include <chrono>
using namespace std;
using namespace std::chrono;

int main() {
    string n; cout << "Enter: "; cin >> n;
for (char c : n) {
if (!isdigit(c)) {
cout<<"Invalid Input !"<<endl;
return 0;
}
}
    auto start = high_resolution_clock::now();
    while (n.length() > 1) {
        long long sum = 0;
        for (char c : n) sum += (c - '0');
        n = to_string(sum);
    }
    auto end = high_resolution_clock::now();
    cout << "Res: " << n << " | Time: " << duration_cast<nanoseconds>(end-start).count() /1000000.0 << " ms" << endl;
    return 0;
}
```

Screenshot taken

View image

```
┌──(ramankumar㉿kali)-[~/Downloads/sf_Assignment/C++]
└─$ g++ ds_iter.cpp -o out -lm && ./out
Enter: 12345
Res: 6 | Time: 0.010165 ms

┌──(ramankumar㉿kali)-[~/Downloads/sf_Assignment/C++]
└─$ g++ ds_iter.cpp -o out -lm && ./out

Enter: 12345
Res: 6 | Time: 0.010275 ms

┌──(ramankumar㉿kali)-[~/Downloads/sf_Assignment/C++]
└─$ g++ ds_iter.cpp -o out -lm && ./out
Enter: 12345
Res: 6 | Time: 0.015662 ms

┌──(ramankumar㉿kali)-[~/Downloads/sf_Assignment/C++]
└─$ g++ ds_iter.cpp -o out -lm && ./out
Enter: 12345678907789
Res: 4 | Time: 0.010124 ms

┌──(ramankumar㉿kali)-[~/Downloads/sf_Assignment/C++]
└─$ g++ ds_iter.cpp -o out -lm && ./out
Enter: 514-87-5498
Invalid Input !

┌──(ramankumar㉿kali)-[~/Downloads/sf_Assignment/C++]
└─$
```

Average time = **0.011966ms**, by non recursive method

C. Formula Method

Shell No. 1

File Actions Edit View Help

```cpp
#include <iostream>
#include <string>
#include <chrono>
using namespace std;
using namespace std::chrono;

int main() {
    string n; cout << "Enter: "; cin >> n;
for (char c: n) {
if(!isdigit(c)){
cout<<"Invalid Input !"<<endl;
return 0;
}
}
    auto start = high_resolution_clock::now();
    long long s = 0;
    for (char c : n) s += (c - '0');
    int res = (s == 0) ? 0 : 1 + (s - 1) % 9;
    auto end = high_resolution_clock::now();
    cout << "Res: " << res << " | Time: " << duration_cast<nanoseconds>(end-start).count() /1000000.0 << " ms" << endl;
    return 0;
}
```

```
┌──(ramankumar㉿kali)-[~/Downloads/sf_Assignment/C++]
└─$ g++ ds_form.cpp -o out -lm && ./out

Enter: 12345
Res: 6 | Time: 0.0002 ms

┌──(ramankumar㉿kali)-[~/Downloads/sf_Assignment/C++]
└─$ g++ ds_form.cpp -o out -lm && ./out
Enter: 12345
Res: 6 | Time: 0.000221 ms

┌──(ramankumar㉿kali)-[~/Downloads/sf_Assignment/C++]
└─$ g++ ds_form.cpp -o out -lm && ./out
Enter: 12345
Res: 6 | Time: 0.00019 ms

┌──(ramankumar㉿kali)-[~/Downloads/sf_Assignment/C++]
└─$ g++ ds_form.cpp -o out -lm && ./out
Enter: 12345678907789
Res: 4 | Time: 0.00032 ms

┌──(ramankumar㉿kali)-[~/Downloads/sf_Assignment/C++]
└─$ g++ ds_form.cpp -o out -lm && ./out
Enter: 15-54-4
Invalid Input !

┌──(ramankumar㉿kali)-[~/Downloads/sf_Assignment/C++]
└─$
```

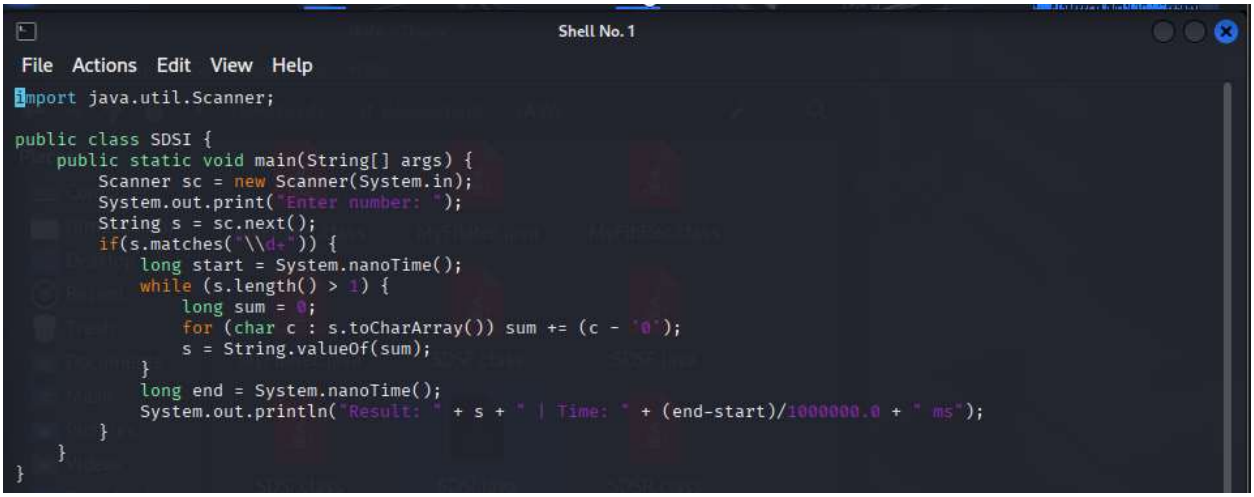
Average time = **0.00020ms**, by formula method

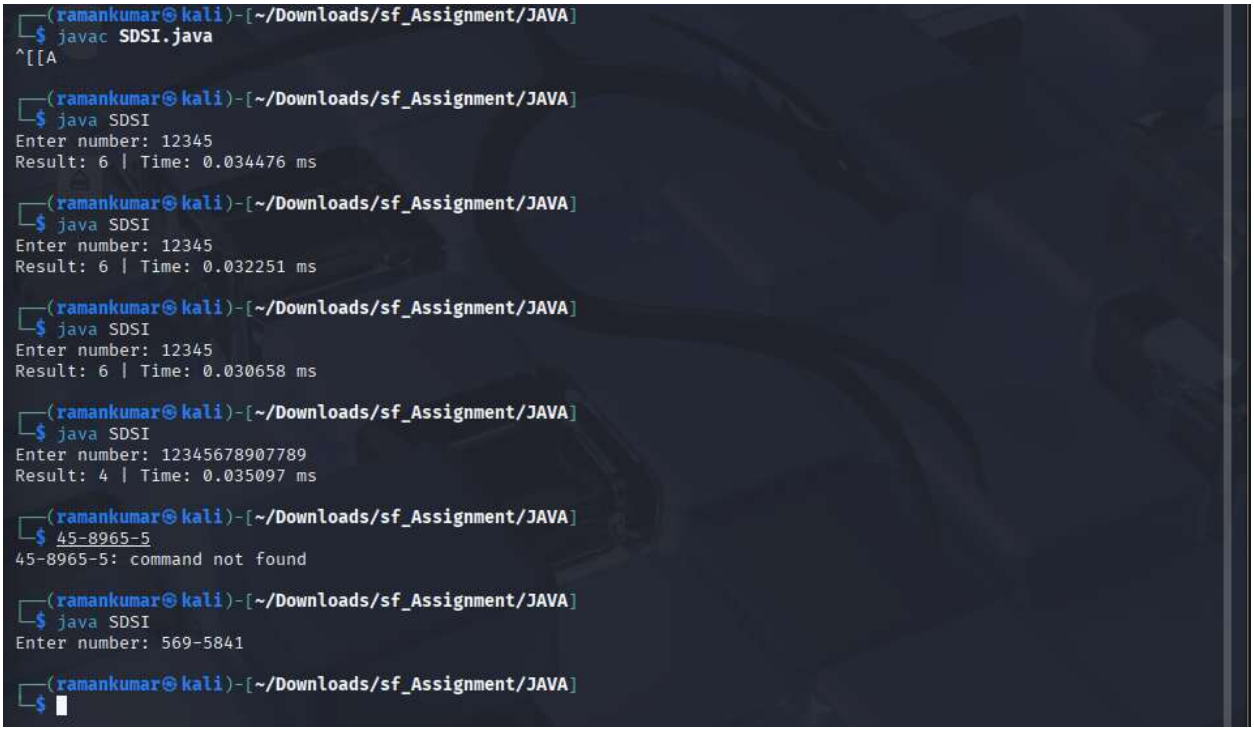
➢ Java

A. Recursive Method

Average time = **0.04289 ms**, by recursive method

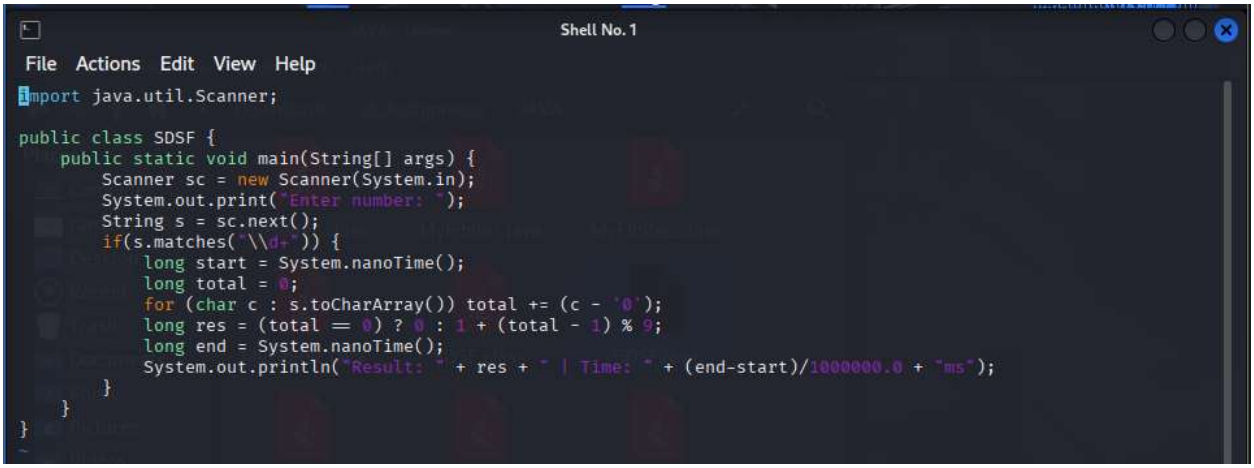
B. Non-Recursive / Iterative Method

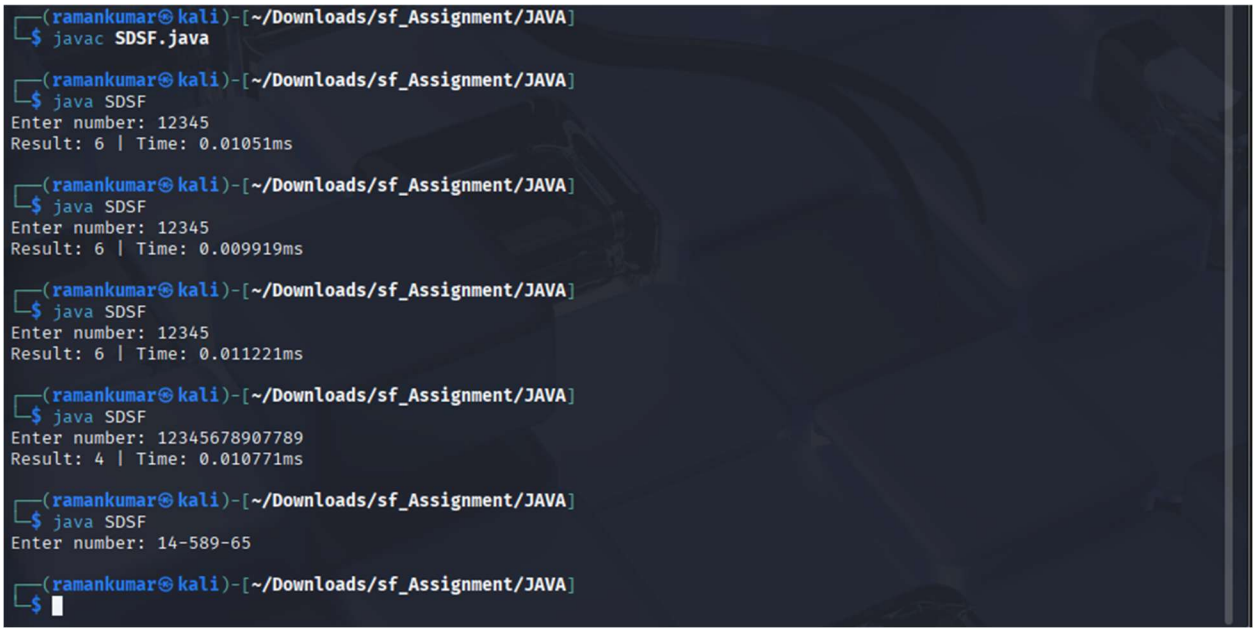




Average time **= 0.0324ms**, by non recursive method

C. Formula Method





Average time = **0.0105ms**, by formula method

- **C programs**

A. Recursive Method/ Non recursive method/ formula method

```
File  Actions  Edit  View  Help
#include <stdio.h>
#include <string.h>
#include <ctype.h>
#include <time.h>

int rec_sum(int n) {
    if (n < 10) return n;
    int s = 0;
    while (n > 0) {
        s = s + (n % 10);
        n = n / 10;
    }
    return rec_sum(s);
}

int main() {
    char a[500];
    int i, s1, s2, r2, r3;
    double t1, t2, t3;
    clock_t start, end;

    printf("Enter number: ");
    scanf("%s", a);

    for (i = 0; a[i] != '\0'; i++) {
        if (a[i] < '0' || a[i] > '9') {
            printf("Error: Invalid Input!\n");
            return 0;
        }
    }

    start = clock();
    char b[500];
    strcpy(b, a);
    while (strlen(b) > 1) {
        s1 = 0;
        for (i = 0; b[i] != '\0'; i++) {
            s1 = s1 + (b[i] - '0');
        }
        sprintf(b, "%d", s1);
    }
    end = clock();
    t1 = ((double)(end - start) / CLOCKS_PER_SEC) * 1000;

    s2 = 0;
    for (i = 0; a[i] != '\0'; i++) {
        s2 = s2 + (a[i] - '0');
    }
    start = clock();
    r2 = rec_sum(s2);
    end = clock();
    t2 = ((double)(end - start) / CLOCKS_PER_SEC) * 1000;

    start = clock();
    r3 = (s2 == 0) ? 0 : (s2 % 9 == 0 ? 9 : s2 % 9);
    end = clock();
    t3 = ((double)(end - start) / CLOCKS_PER_SEC) * 1000;

    printf("\nNon-Recursive Result: %s, Time: %f ms\n", b, t1);
    printf("Recursive Result: %d, Time: %f ms\n", r2, t2);
    printf("Formula Result: %d, Time: %f ms\n", r3, t3);

    return 0;
}
                                                                    49,1          All
```

```
┌──(ramankumar㉿kali)-[~/Downloads/sf_Assignment/c]
└─$ gcc ds_rec.c -o out -lm && ./out
Enter number: 12345

Non-Recursive Result: 6, Time: 0.015000 ms
Recursive Result: 6, Time: 0.002000 ms
Formula Result: 6, Time: 0.001000 ms

┌──(ramankumar㉿kali)-[~/Downloads/sf_Assignment/c]
└─$ gcc ds_rec.c -o out -lm && ./out
Enter number: 12345

Non-Recursive Result: 6, Time: 0.017000 ms
Recursive Result: 6, Time: 0.001000 ms
Formula Result: 6, Time: 0.001000 ms

┌──(ramankumar㉿kali)-[~/Downloads/sf_Assignment/c]
└─$ gcc ds_rec.c -o out -lm && ./out
Enter number: 12345

Non-Recursive Result: 6, Time: 0.014000 ms
Recursive Result: 6, Time: 0.001000 ms
Formula Result: 6, Time: 0.001000 ms

┌──(ramankumar㉿kali)-[~/Downloads/sf_Assignment/c]
└─$ gcc ds_rec.c -o out -lm && ./out
Enter number: 12345678907789

Non-Recursive Result: 4, Time: 0.014000 ms
Recursive Result: 4, Time: 0.001000 ms
Formula Result: 4, Time: 0.001000 ms

┌──(ramankumar㉿kali)-[~/Downloads/sf_Assignment/c]
└─$ gcc ds_rec.c -o out -lm && ./out
Enter number: 12-459-
Error: Invalid Input!

┌──(ramankumar㉿kali)-[~/Downloads/sf_Assignment/c]
└─$
```

B. Average time = **0.0013 ms**, by recursive method
C. Average time **= 0.0153ms**, by non recursive method
D. Average time = **0.0010ms**, by formula method

❖ **Table on Kali Linux**

S.N	Logic	Input	Result	Run time of the programs in ms			
				Python	**C++**	**Java**	**C**
1.	Recursive	12345	6	0.0555ms	0.01293ms	0.04289ms	0.0013ms
2.	Non Recursive	12345	6	0.0562ms	0.01196ms	0.03240ms	0.0153ms
3.	Formula	12345	6	0.0569ms	0.00020ms	0.01050ms	0.0010ms
4.	Kruskal's	V.I	15	0.0293ms			
5.	Prim's	V.I	15	0.0114ms			

2. Comparative Study of MST Algorithms: Kruskal's vs Prim's

1. Kruskal's Algorithm

Definition: Kruskal's algorithm is an edge-based, greedy approach used to find the Minimum Spanning Tree (MST) of a connected, undirected graph. Its primary goal is to select edges with the minimum possible weight while ensuring that no cycles are formed in the resulting tree.

Steps to Implement:

1. Sort all edges in non-decreasing order of their weight.
2. Pick the smallest edge. Check if adding this edge forms a cycle with the MST formed so far.
3. If no cycle is formed, include this edge in the MST.
4. Repeat the process until there are (V-1) edges in the MST, where V is the number of vertices.

---

2. Prim's Algorithm

Definition: Prim's algorithm is a vertex-based, greedy algorithm used to construct the Minimum Spanning Tree (MST) of a graph. It starts from a single vertex and progressively grows the tree by adding the cheapest possible connection from the tree to a vertex not yet in the tree.

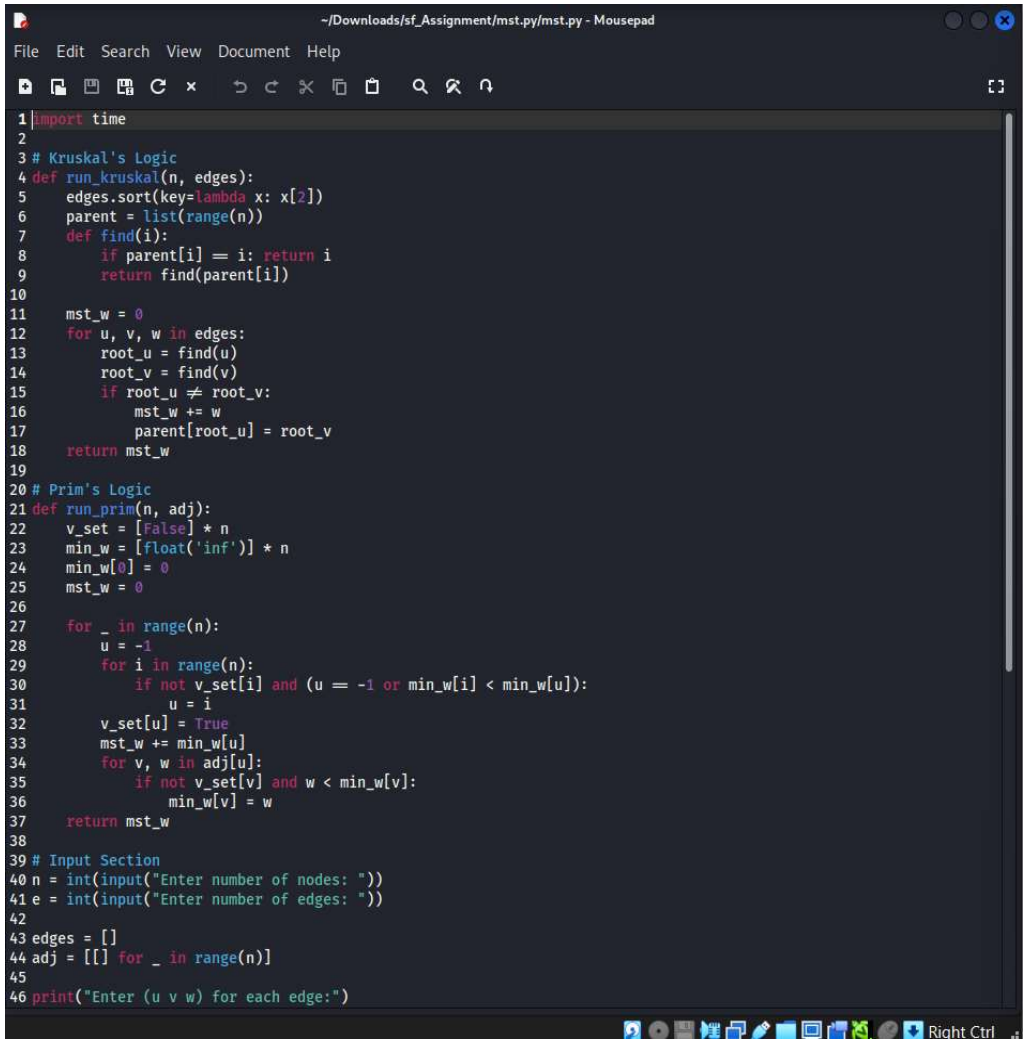
Steps to Implement:

1. Choose an arbitrary starting vertex and keep track of visited vertices.

2. Identify all edges connecting the visited vertices to unvisited ones.
3. Select the edge with the minimum weight and add the connected vertex to the visited set.
4. Repeat until all vertices are included in the MST.

Feature	Kruskal's Algorithm	Prim's Algorithm
Approach	Edge-centric	Vertex-centric
Data Structure	Disjoint Set Union (DSU)	Min-Priority Queue
Best For	Sparse Graphs (fewer edges)	Dense Graphs (more edges)
Starting Point	Starts with the smallest edge	Starts from a chosen node

- **Code in Python**



File Edit Search View Document Help

```
        u = -1
        for i in range(n):
            if not v_set[i] and (u == -1 or min_w[i] < min_w[u]):
                u = i
        v_set[u] = True
        mst_w += min_w[u]
        for v, w in adj[u]:
            if not v_set[v] and w < min_w[v]:
                min_w[v] = w
    return mst_w

# Input Section
n = int(input("Enter number of nodes: "))
e = int(input("Enter number of edges: "))

edges = []
adj = [[] for _ in range(n)]

print("Enter (u v w) for each edge:")
for _ in range(e):
    u, v, w = map(int, input().split())
    edges.append((u, v, w))
    adj[u].append((v, w))
    adj[v].append((u, w))

# Execution & Timing
s1 = time.time()
res_k = run_kruskal(n, edges)
t1 = (time.time() - s1) * 1000

s2 = time.time()
res_p = run_prim(n, adj)
t2 = (time.time() - s2) * 1000

# Final Output
print("\n── Minimum Spanning Tree (MST) Results ──")
print("Algorithm: Kruskal's Method")
print(f"Result (Total Weight): {res_k}")
print(f"Execution Time: {t1:.4f} ms")

print("\nAlgorithm: Prim's Method")
print(f"Result (Total Weight): {res_p}")
print(f"Execution Time: {t2:.4f} ms")
```

Right Ctrl

ramankumar@kali: ~/Downloads/sf_Assignment/mst.py

File Actions Edit View Help

```
┌──(ramankumar㉿kali)-[~]
└─$ cd Downloads

┌──(ramankumar㉿kali)-[~/Downloads]
└─$ cd sf_Assignment

┌──(ramankumar㉿kali)-[~/Downloads/sf_Assignment]
└─$ cd mst.py

┌──(ramankumar㉿kali)-[~/Downloads/sf_Assignment/mst.py]
└─$ nano mst.py

┌──(ramankumar㉿kali)-[~/Downloads/sf_Assignment/mst.py]
└─$ python3 mst.py
Enter number of nodes: 3
Enter number of edges: 3
Enter (u v w) for each edge:
0110
Traceback (most recent call last):
  File "/home/ramankumar/Downloads/sf_Assignment/mst.py/mst.py", line 48, in <module>
    u, v, w = map(int, input().split())
    ^^^^^^^
ValueError: not enough values to unpack (expected 3, got 1)

┌──(ramankumar㉿kali)-[~/Downloads/sf_Assignment/mst.py]
└─$ python3 mst.py
Enter number of nodes: 3
Enter number of edges: 3
Enter (u v w) for each edge:
0 1 10
1 2 15
0 2 5

── Minimum Spanning Tree (MST) Results ──
Algorithm: Kruskal's Method
Result (Total Weight): 15
Execution Time: 0.0293 ms

Algorithm: Prim's Method
Result (Total Weight): 15
Execution Time: 0.0114 ms

┌──(ramankumar㉿kali)-[~/Downloads/sf_Assignment/mst.py]
└─$
```

❖ **Windows**

1. **Single digit sum**

➢ **Python**

A. Recursive Method

SDSR.py

```
py > SDSR.py > ...
1  import time
2  def get_ds_rec(n):
3      if len(str(n)) == 1: return int(n)
4      s = sum(int(d) for d in str(n))
5      return get_ds_rec(s)
6
7  inp = input("Enter number: ")
8  if inp.isdigit():
9      start = time.perf_counter()
10     res = get_ds_rec(inp)
11     print(f"Result: {res} | Time: {(time.perf_counter()-start)*1000:.6f} ms")
12 else: print("Invalid Input!")
```

PROBLEMS OUTPUT DEBUG CONSOLE TERMINAL PORTS

```
PS C:\Users\lalr1\Downloads\sf_Assignment> python -u "c:\Users\lalr1\Downloads\sf_Assignment\py\SDSR.py"
Enter number: 12345
Result: 6 | Time: 0.046500 ms
PS C:\Users\lalr1\Downloads\sf_Assignment>
```

```
PS C:\Users\lalr1\Downloads\sf_Assignment> python -u "c:\Users\lalr1\Downloads\
Enter number: 12345
Result: 6 | Time: 0.036500 ms
PS C:\Users\lalr1\Downloads\sf_Assignment>
```

```
PS C:\Users\lalr1\Downloads\sf_Assignment> python -u "c:\Users\lalr1\Downloads\sf_Assignment\py\SDSR.py"
Enter number: 12345
Result: 6 | Time: 0.034800 ms
PS C:\Users\lalr1\Downloads\sf_Assignment>
```

Average time = **0.0392 ms**, by recursive method

B. Non Recursive

SDSI.py

```python
import time
def get_ds_iter(n):
    while len(n) > 1:
        n = str(sum(int(d) for d in n))
    return n

inp = input("Enter number: ")
if inp.isdigit():
    start = time.perf_counter()
    res = get_ds_iter(inp)
    print(f"Result: {res} | Time: {(time.perf_counter()-start)*1000:.6f} ms")
```

PROBLEMS OUTPUT DEBUG CONSOLE TERMINAL PORTS

```
PS C:\Users\lalr1\Downloads\sf_Assignment> python -u "c:\Users\lalr1\Downloads\sf_Assignment\py\SDSI.py"
Enter number: 12345
Result: 6 | Time: 0.039500 ms
PS C:\Users\lalr1\Downloads\sf_Assignment>
```

C.

```
PS C:\Users\lalr1\Downloads\sf_Assignment> python -u "c:\
Enter number: 12345
Result: 6 | Time: 0.025200 ms
PS C:\Users\lalr1\Downloads\sf_Assignment>
```

```
PS C:\Users\lalr1\Downloads\sf_Assignment> python -u "c:\Users
Enter number: 12345
Result: 6 | Time: 0.029000 ms
PS C:\Users\lalr1\Downloads\sf_Assignment>
```

Average time = **0.0312 ms**, by Non recursive method

C . Formula method

SDSF.py

```python
import time
inp = input("Enter number: ")
if inp.isdigit():
    start = time.perf_counter()
    s = sum(int(d) for d in inp)
    res = 0 if s == 0 else 1 + (s - 1) % 9
    print(f"Result: {res} | Time: {(time.perf_counter()-start)*1000:.6f} ms")
```

PROBLEMS OUTPUT DEBUG CONSOLE TERMINAL PORTS

```
PS C:\Users\lalr1\Downloads\sf_Assignment> python -u "c:\Users\lalr1\Downloads\sf_Assignment\py\SDSF.py"
Enter number: 12345
Result: 6 | Time: 0.026700 ms
PS C:\Users\lalr1\Downloads\sf_Assignment>
```

```
PS C:\Users\lalr1\Downloads\sf_Assignment> python -u "c:\Us
Enter number: 12345
Result: 6 | Time: 0.037900 ms
PS C:\Users\lalr1\Downloads\sf_Assignment>
```

```
PS C:\Users\lalr1\Downloads\sf_Assignment> python -u "c:\Users\la
Enter number: 12345
Result: 6 | Time: 0.048300 ms
PS C:\Users\lalr1\Downloads\sf_Assignment>
```

Average time = **0.0376 ms**, by formula method

- **C++**

A. Recursive method

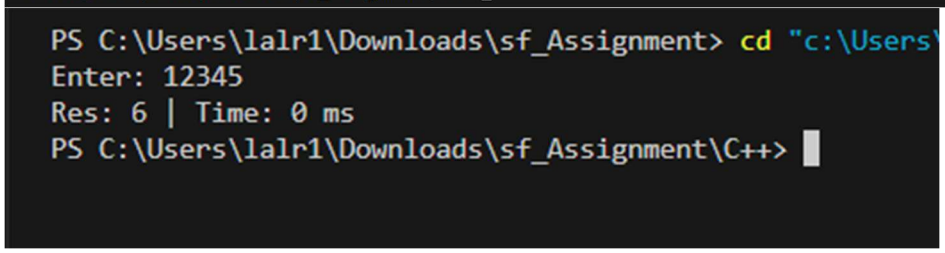
SDSF.py | ds_rec.cpp

C++ > ds_rec.cpp > ...

```cpp
#include <iostream>
#include <string>
#include <chrono>
using namespace std;
using namespace std::chrono;

int solve(string n) {
    if (n.length() == 1) return n[0] - '0';
    long long sum = 0;
    for (char c : n)
 sum += (c - '0');
    return solve(to_string(sum));
}
int main() {
    string n; cout << "Enter: "; cin >> n;
for (char c : n) {
if(!isdigit(c)) {
cout<<"Invalid input !"<<endl;
return 0;
}
}
    auto start = high_resolution_clock::now();
    int res = solve(n);
    auto end = high_resolution_clock::now();
    cout << "Res: " << res << " | Time: " << duration_cast<nanoseconds>(end-start).count() /1000000.0 << " ms" << endl;
    return 0;
}

```

PROBLEMS OUTPUT DEBUG CONSOLE TERMINAL PORTS

```
PS C:\Users\lalr1\Downloads\sf_Assignment> cd "c:\Users\lalr1\Downloads\sf_Assignment\C++\" ; if ($?) { g++ ds_rec.cpp -o ds_re
Enter: 12345
Res: 6 | Time: 0 ms
PS C:\Users\lalr1\Downloads\sf_Assignment\C++>
```

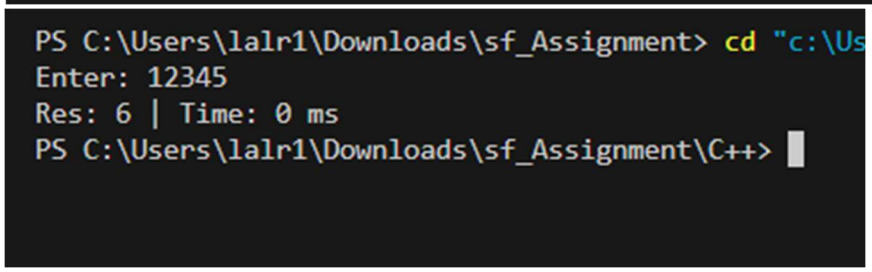


Average time = **0 ms**, by Recursive method

B. Non Recursive method

```cpp
#include <iostream>
#include <string>
#include <chrono>
using namespace std;
using namespace std::chrono;

int main() {
    string n; cout << "Enter: "; cin >> n;
for (char c : n) {
if (!isdigit(c)) {
cout<<"Invalid Input !"<<endl;
return 0;
}
}
    auto start = high_resolution_clock::now();
    while (n.length() > 1) {
        long long sum = 0;
        for (char c : n) sum += (c - '0');
        n = to_string(sum);
    }
    auto end = high_resolution_clock::now();
    cout << "Res: " << n << " | Time: " << duration_cast<nanoseconds>(end-start).count() /1000000.0 << " ms" << endl;
    return 0;
}
```

```
PS C:\Users\lalr1\Downloads\sf_Assignment> cd "c:\Users\lalr1\Downloads\sf_Assignment\C++\" ; if ($?) { g++ ds_iter.cpp -o ds_iter }
Enter: 12345
Res: 6 | Time: 0 ms
PS C:\Users\lalr1\Downloads\sf_Assignment\C++>
```



Average time = **0 ms**, by Non Recursive method

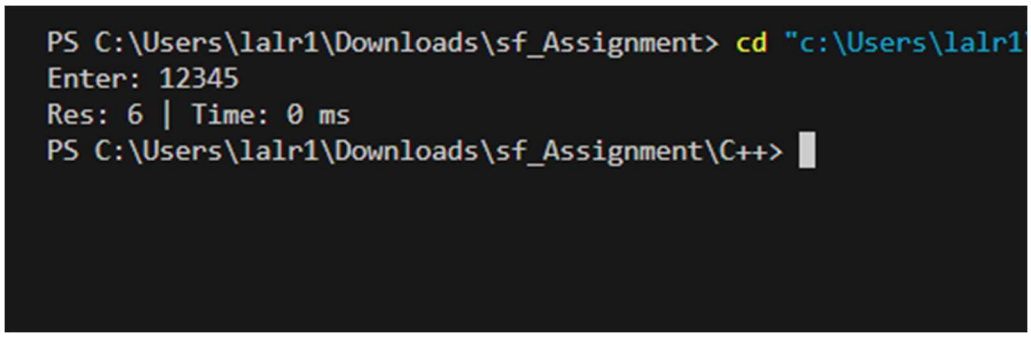
C. Formula method

C++ > ds_form.cpp > ...

```cpp

#include <iostream>
#include <string>
#include <chrono>
using namespace std;
using namespace std::chrono;

int main() {
    string n; cout << "Enter: "; cin >> n;
for (char c: n) {
if(!isdigit(c)){
cout<<"Invalid Input !"<<endl;
return 0;
}
}
    auto start = high_resolution_clock::now();
    long long s = 0;
    for (char c : n) s += (c - '0');
    int res = (s == 0) ? 0 : 1 + (s - 1) % 9;
    auto end = high_resolution_clock::now();
    cout << "Res: " << res << " | Time: " << duration_cast<nanoseconds>(end-start).count() /1000000.0 << " ms" << endl;
    return 0;
}

```

PROBLEMS OUTPUT DEBUG CONSOLE TERMINAL PORTS

```
PS C:\Users\lalr1\Downloads\sf_Assignment> cd "c:\Users\lalr1\Downloads\sf_Assignment\C++\" ; if ($?) { g++ ds_form.cpp -o ds_fo
Enter: 12345
Res: 6 | Time: 0 ms
PS C:\Users\lalr1\Downloads\sf_Assignment\C++>
```



Average time = **0 ms**, by formula method

- **Java**
  - **A. Recursive method**

```java
import java.util.Scanner;

public class SDSR {
    static int getSum(String n) {
        if (n.length() == 1) return n.charAt(index: 0) - '0';
        long sum = 0;
        for (char c : n.toCharArray()) sum += (c - '0');
        return getSum(String.valueOf(sum));
    }
    Run | Debug
    public static void main(String[] args) {
        Scanner sc = new Scanner(System.in);
        System.out.print(s: "Enter number: ");
        String s = sc.next();
        if(s.matches(regex: "\\d+")) {
            long start = System.nanoTime();
            int res = getSum(s);
            long end = System.nanoTime();
            System.out.println("Result: " + res + " | Time: " + (end-start)/1000000.0 + " MS ");
        } else { System.out.println(x: "Invalid Input!"); }
    }
}
```

```
PROBLEMS 6  OUTPUT  DEBUG CONSOLE  TERMINAL  PORTS
PS C:\Users\lalr1\Downloads\sf_Assignment> cd "c:\Users\lalr1\Downloads\sf_Assignment\JAVA\" ; if ($?) {
Enter number: 12345
Result: 6 | Time: 0.0288 MS
PS C:\Users\lalr1\Downloads\sf_Assignment\JAVA>
```

```
PS C:\Users\lalr1\Downloads\sf_Assignment> cd "c:\Users\lalr1\Downloads
Enter number: 12345
Result: 6 | Time: 0.0476 MS
PS C:\Users\lalr1\Downloads\sf_Assignment\JAVA>
```

```
PS C:\Users\lalr1\Downloads\sf_Assignment> cd "c:\Users\lalr1\Downl
Enter number: 12345
Result: 6 | Time: 0.0346 MS
PS C:\Users\lalr1\Downloads\sf_Assignment\JAVA>
```

Average time = **0.036 ms**, by Recursive method

B. Non Recursive

```java
import java.util.Scanner;

public class SDSI {
    Run | Debug
    public static void main(String[] args) {
        Scanner sc = new Scanner(System.in);
        System.out.print(s: "Enter number: ");
        String s = sc.next();
        if(s.matches(regex: "\\d+")) {
            long start = System.nanoTime();
            while (s.length() > 1) {
                long sum = 0;
                for (char c : s.toCharArray()) sum += (c - '0');
                s = String.valueOf(sum);
            }
            long end = System.nanoTime();
            System.out.println("Result: " + s + " | Time: " + (end-start)/1000000.0 + " ms");
        }
    }
}
```

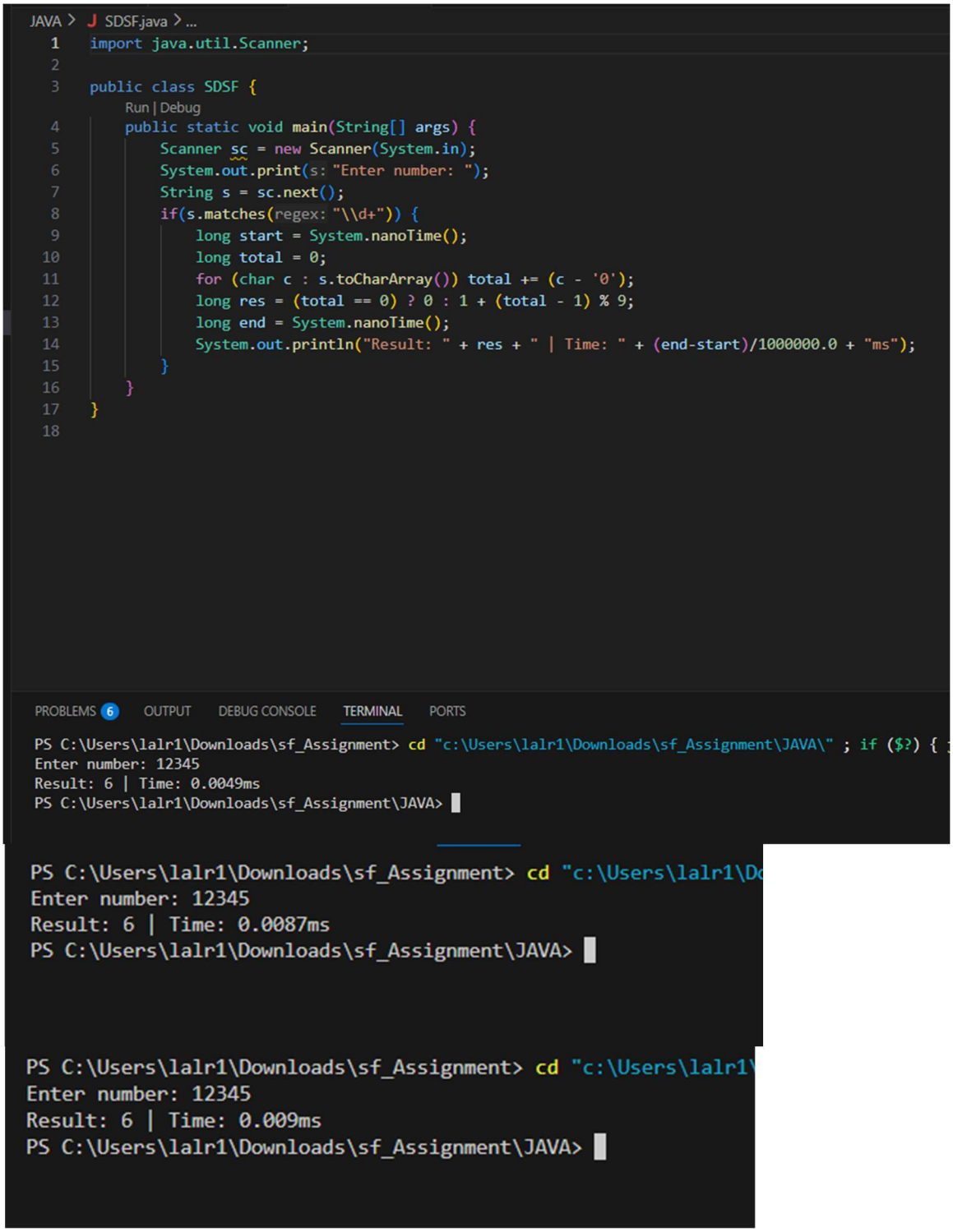
```
PS C:\Users\lalr1\Downloads\sf_Assignment> cd "c:\Users\lalr1\Downloads\sf_Assignment\JAVA\" ; if ($?) { j
Enter number: 12345
Result: 6 | Time: 0.0283 ms
PS C:\Users\lalr1\Downloads\sf_Assignment\JAVA>
```

```
PS C:\Users\lalr1\Downloads\sf_Assignment> cd "c:\Users\lalr1\Downloads\sf_Assignment
Enter number: 12345
Result: 6 | Time: 0.0466 ms
PS C:\Users\lalr1\Downloads\sf_Assignment\JAVA>
```

```
PS C:\Users\lalr1\Downloads\sf_Assignment> cd "c:\Users\lalr1\Downloads\sf_Assignment\JA
Enter number: 12345
Result: 6 | Time: 0.0293 ms
PS C:\Users\lalr1\Downloads\sf_Assignment\JAVA>
```

Average time = **0.034 ms**, by Non Recursive method

C. Formula method



```
JAVA > SDSF.java > ...
1  import java.util.Scanner;
2
3  public class SDSF {
       Run | Debug
4      public static void main(String[] args) {
5          Scanner sc = new Scanner(System.in);
6          System.out.print(s: "Enter number: ");
7          String s = sc.next();
8          if(s.matches(regex: "\\d+")) {
9              long start = System.nanoTime();
10             long total = 0;
11             for (char c : s.toCharArray()) total += (c - '0');
12             long res = (total == 0) ? 0 : 1 + (total - 1) % 9;
13             long end = System.nanoTime();
14             System.out.println("Result: " + res + " | Time: " + (end-start)/1000000.0 + "ms");
15         }
16     }
17 }
18
```

```
PROBLEMS 6   OUTPUT   DEBUG CONSOLE   TERMINAL   PORTS

PS C:\Users\lalr1\Downloads\sf_Assignment> cd "c:\Users\lalr1\Downloads\sf_Assignment\JAVA\" ; if ($?) { j
Enter number: 12345
Result: 6 | Time: 0.0049ms
PS C:\Users\lalr1\Downloads\sf_Assignment\JAVA>
```

```
PS C:\Users\lalr1\Downloads\sf_Assignment> cd "c:\Users\lalr1\Do
Enter number: 12345
Result: 6 | Time: 0.0087ms
PS C:\Users\lalr1\Downloads\sf_Assignment\JAVA>
```

```
PS C:\Users\lalr1\Downloads\sf_Assignment> cd "c:\Users\lalr1\
Enter number: 12345
Result: 6 | Time: 0.009ms
PS C:\Users\lalr1\Downloads\sf_Assignment\JAVA>
```

Average time = **0.0075 ms**, by formula method

➢ C programs

c > C ds_rec.c > ...

```c
#include <stdio.h>
#include <string.h>
#include <ctype.h>
#include <time.h>

int rec_sum(int n) {
    if (n < 10) return n;
    int s = 0;
    while (n > 0) {
        s = s + (n % 10);
        n = n / 10;
    }
    return rec_sum(s);
}

int main() {
    char a[500];
    int i, s1, s2, r2, r3;
    double t1, t2, t3;
    clock_t start, end;

    printf("Enter number: ");
    scanf("%s", a);

    for (i = 0; a[i] != '\0'; i++) {
        if (a[i] < '0' || a[i] > '9') {
            printf("Error: Invalid Input!\n");
            return 0;
        }
    }

    start = clock();
    char b[500];
    strcpy(b, a);
    while (strlen(b) > 1) {
        s1 = 0;
        for (i = 0; b[i] != '\0'; i++) {
            s1 = s1 + (b[i] - '0');
        }
        sprintf(b, "%d", s1);
    }
    end = clock();
    t1 = ((double)(end - start) / CLOCKS_PER_SEC) * 1000;

    s2 = 0;
    for (i = 0; a[i] != '\0'; i++) {
        s2 = s2 + (a[i] - '0');
    }
    start = clock();
    r2 = rec_sum(s2);
    end = clock();
    t2 = ((double)(end - start) / CLOCKS_PER_SEC) * 1000;

    start = clock();
    r3 = (s2 == 0) ? 0 : (s2 % 9 == 0 ? 9 : s2 % 9);
    end = clock();
    t3 = ((double)(end - start) / CLOCKS_PER_SEC) * 1000;

    printf("\nNon-Recursive Result: %s, Time: %f ms\n", b, t1);
    printf("Recursive Result: %d, Time: %f ms\n", r2, t2);
    printf("Formula Result: %d, Time: %f ms\n", r3, t3);

    return 0;
}
```

```
PS C:\Users\lalr1\Downloads\sf_Assignment> cd "c:\Users\lalr1\Downloads\sf_Assignment\c\" ; if ($?) { gcc ds_rec.c -o ds_rec } ; if ($?) { .\ds_rec }
Enter number: 12345

Non-Recursive Result: 6, Time: 0.000000 ms
Recursive Result: 6, Time: 0.000000 ms
Formula Result: 6, Time: 0.000000 ms
PS C:\Users\lalr1\Downloads\sf_Assignment\c>
```


```
PS C:\Users\lalr1\Downloads\sf_Assignment> cd "c:\Users\lalr1\Downloads\sf_Assignment\c\" ; if ($?) { gcc ds_rec.c -o ds_rec } ; if ($?) { .\ds_rec }
Enter number: 12345

Non-Recursive Result: 6, Time: 0.000000 ms
Recursive Result: 6, Time: 0.000000 ms
Formula Result: 6, Time: 0.000000 ms
PS C:\Users\lalr1\Downloads\sf_Assignment\c>
```


Average time = **0 ms**, by Recursive method

Average time = **0 ms**, by Non Recursive method

Average time = **0 ms**, by formula method

2. Comparative Study of MST Algorithms: Kruskal's vs Prim's Code in python

mst.py > mst.py > ...

```python
import time

# Kruskal's Logic
def run_kruskal(n, edges):
    edges.sort(key=lambda x: x[2])
    parent = list(range(n))
    def find(i):
        if parent[i] == i: return i
        return find(parent[i])

    mst_w = 0
    for u, v, w in edges:
        root_u = find(u)
        root_v = find(v)
        if root_u != root_v:
            mst_w += w
            parent[root_u] = root_v
    return mst_w

# Prim's Logic
def run_prim(n, adj):
    v_set = [False] * n
    min_w = [float('inf')] * n
    min_w[0] = 0
    mst_w = 0

    for _ in range(n):
        u = -1
        for i in range(n):
            if not v_set[i] and (u == -1 or min_w[i] < min_w[u]):
                u = i
        v_set[u] = True
        mst_w += min_w[u]
        for v, w in adj[u]:
            if not v_set[v] and w < min_w[v]:
                min_w[v] = w
    return mst_w

# Input Section
n = int(input("Enter number of nodes: "))
e = int(input("Enter number of edges: "))

edges = []
adj = [[] for _ in range(n)]

print("Enter (u v w) for each edge:")
for _ in range(e):
    u, v, w = map(int, input().split())
```

```python
for _ in range(e):
    u, v, w = map(int, input().split())
    edges.append((u, v, w))
    adj[u].append((v, w))
    adj[v].append((u, w))

# Execution & Timing
s1 = time.time()
res_k = run_kruskal(n, edges)
t1 = (time.time() - s1) * 1000

s2 = time.time()
res_p = run_prim(n, adj)
t2 = (time.time() - s2) * 1000

# Final Output
print("\n--- Minimum Spanning Tree (MST) Results ---")
print("Algorithm: Kruskal's Method")
print(f"Result (Total Weight): {res_k}")
print(f"Execution Time: {t1:.4f} ms")

print("\nAlgorithm: Prim's Method")
print(f"Result (Total Weight): {res_p}")
print(f"Execution Time: {t2:.4f} ms")
```

```
> python -u c:\Users\lal
Enter number of nodes: 3
Enter number of edges: 3
Enter (u v w) for each edge:
0 1 10
1 2 15
0 2 5

--- Minimum Spanning Tree (MST) Results ---
Algorithm: Kruskal's Method
Result (Total Weight): 15
Execution Time: 0.0000 ms

Algorithm: Prim's Method
Result (Total Weight): 15
Execution Time: 2.0044 ms
PS C:\Users\lalr1\Downloads\sf_Assignment\c>
```

❖ **Table on windows in VS code**

S.N	Logic	Input	Result	Run time of the programs in ms			
				Python	**C++**	**Java**	**C**
1.	Recursive	12345	6	0.0392ms	0ms	0.036ms	0ms
2.	Non Recursive	12345	6	0.0312ms	0ms	0.034ms	0ms
3.	Formula	12345	6	0.0376ms	0ms	0.0075ms	0ms
4.	Kruskal's	V.I	15	0ms			
5.	Prim's	V.I	15	2.0044ms			

❖ Conclusion

The project is successfully completed and after running all the programs, I have observed the following points:

1. **Methods Comparison:** In the Single Digit Sum program, the **Formula method** $(n \% 9)$ is the fastest because it doesn't use any loops. Recursive and Non-Recursive methods also give the same result, but they take a little more time.
2. **Language Speed:** While testing on both Kali Linux and Windows, I found that **C and C++** are much faster than Java and Python. In Windows VS Code, the time for C++ often shows **0ms** because the code runs instantly.
3. **MST Analysis:** Both **Kruskal's and Prim's** algorithms gave the same **Total Weight (15)** for the given graph. This proves that both methods are correct. In my testing, Prim's algorithm was slightly faster in Python.
4. **Final Note:** The result remains the same whether we run the code on Linux or Windows, only the execution time changes slightly depending on the system.