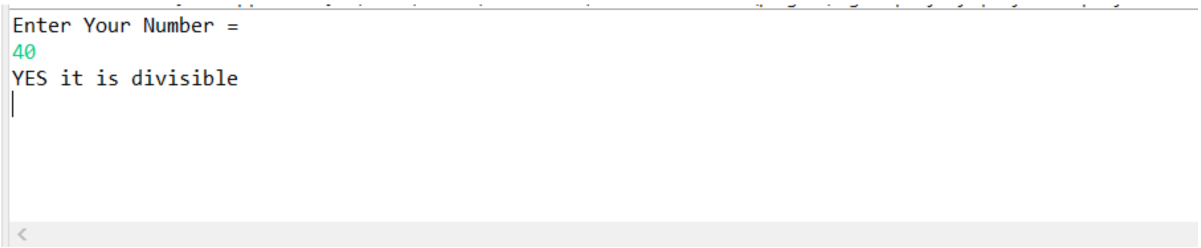
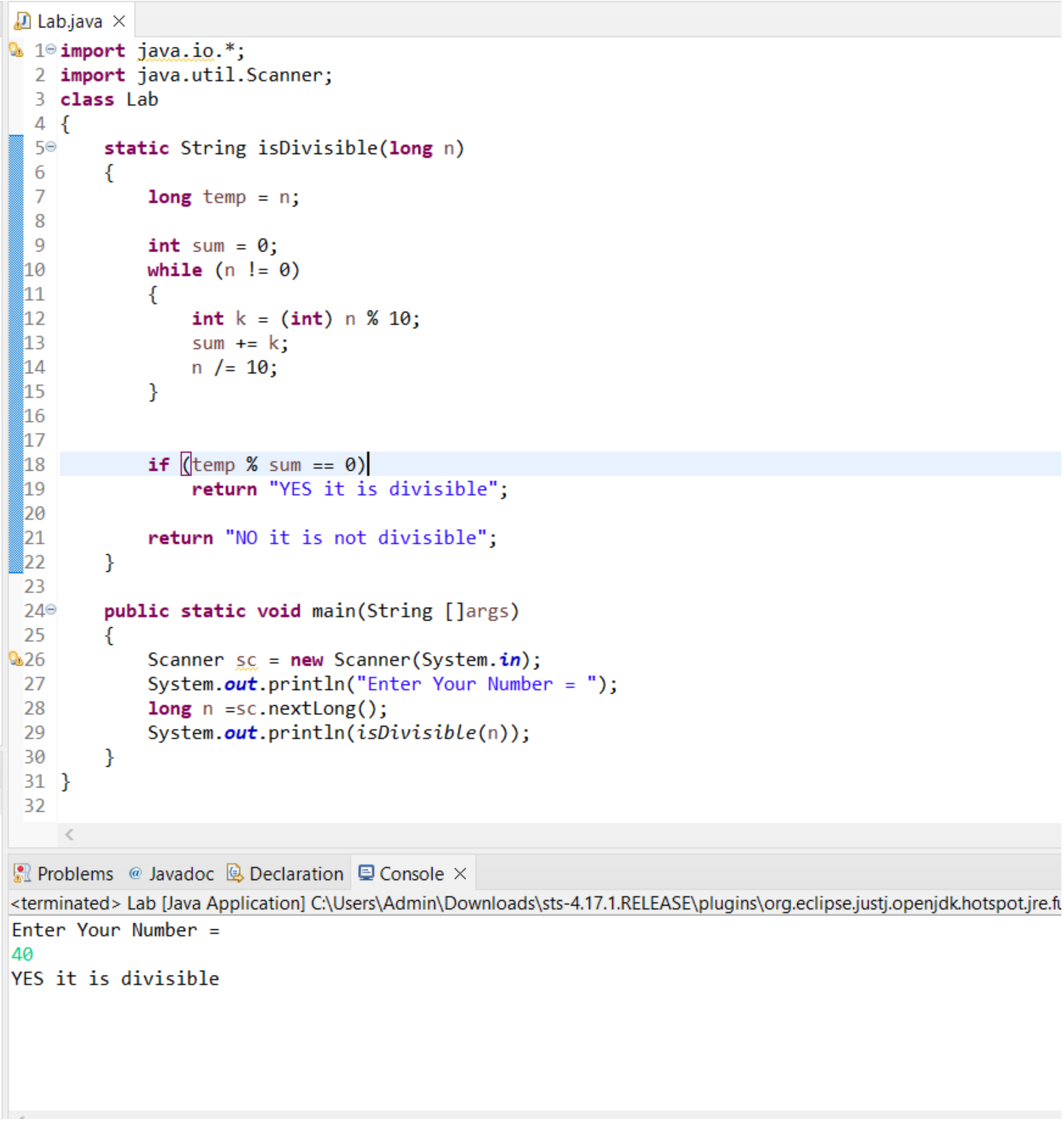
NAME – GAGAN VIJAY BHANGALE

1)

```
import java.io.*;

import java.util.Scanner;

class Lab

{

static String isDivisible(long n)

{

long temp = n;

int sum = 0;

while (n != 0)

{

int k = (int) n % 10;

sum += k;

n /= 10;

}

if (temp % sum == 0)

return "YES it is divisible";

return "NO it is not divisible";
}
public static void main(String []args)
{
Scanner sc = new Scanner(System.in);
System.out.println("Enter Your Number = ");
long n =sc.nextLong();
System.out.println(isDivisible(n));
}
}
```



```
Enter Your Number =
40
YES it is divisible
```



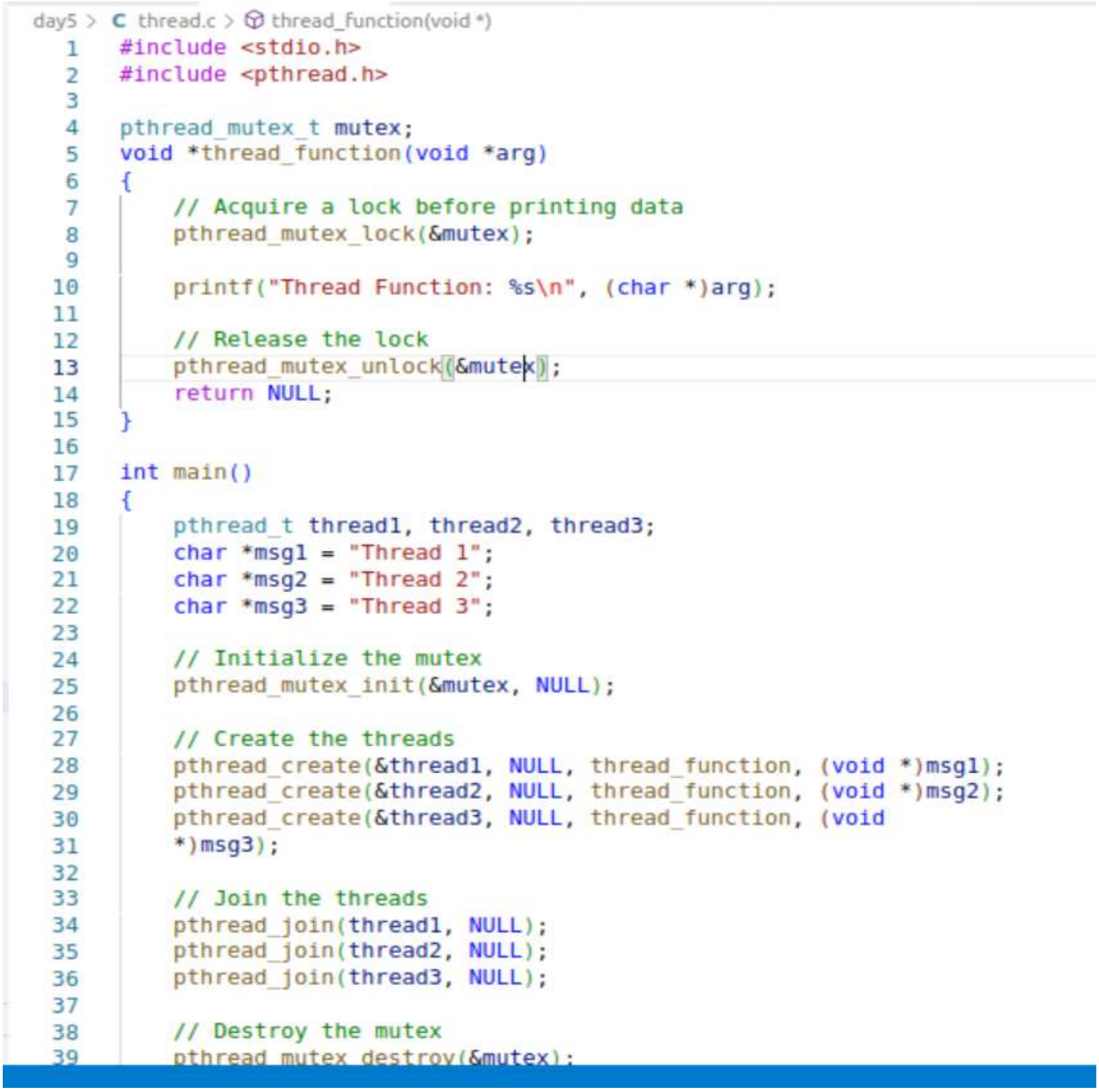
```java
import java.io.*;
import java.util.Scanner;
class Lab
{
    static String isDivisible(long n)
    {
        long temp = n;

        int sum = 0;
        while (n != 0)
        {
            int k = (int) n % 10;
            sum += k;
            n /= 10;
        }


        if (temp % sum == 0)
            return "YES it is divisible";

        return "NO it is not divisible";
    }

    public static void main(String []args)
    {
        Scanner sc = new Scanner(System.in);
        System.out.println("Enter Your Number = ");
        long n =sc.nextLong();
        System.out.println(isDivisible(n));
    }
}

```

<terminated> Lab [Java Application] C:\Users\Admin\Downloads\sts-4.17.1.RELEASE\plugins\org.eclipse.justj.openjdk.hotspot.jre.fu

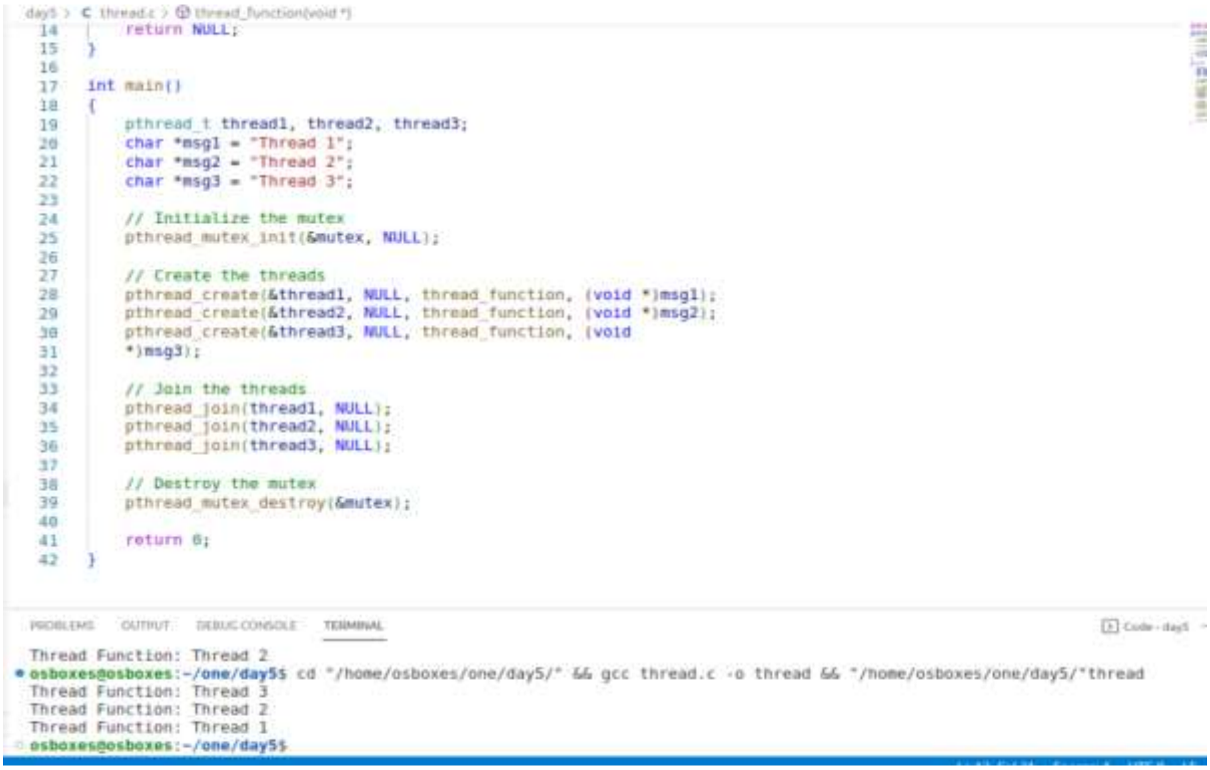
```
Enter Your Number =
40
YES it is divisible
```

2)

```c
#include <stdio.h>
#include <pthread.h>

pthread_mutex_t mutex;

void *thread_function(void *arg)
{
    // Acquire a lock before printing data
    pthread_mutex_lock(&mutex);

    printf("Thread Function: %s\n", (char *)arg);

    // Release the lock
    pthread_mutex_unlock(&mutex);

    return NULL;
}

int main()
{
    pthread_t thread1, thread2, thread3;
    char *msg1 = "Thread 1";
    char *msg2 = "Thread 2";
    char *msg3 = "Thread 3";

    // Initialize the mutex
    pthread_mutex_init(&mutex, NULL);

    // Create the threads
    pthread_create(&thread1, NULL, thread_function, (void *)msg1);
    pthread_create(&thread2, NULL, thread_function, (void *)msg2);
    pthread_create(&thread3, NULL, thread_function, (void
    *)msg3);

    // Join the threads
    pthread_join(thread1, NULL);
    pthread_join(thread2, NULL);
    pthread_join(thread3, NULL);

    // Destroy the mutex
    pthread_mutex_destroy(&mutex);

    return 0;
}
```

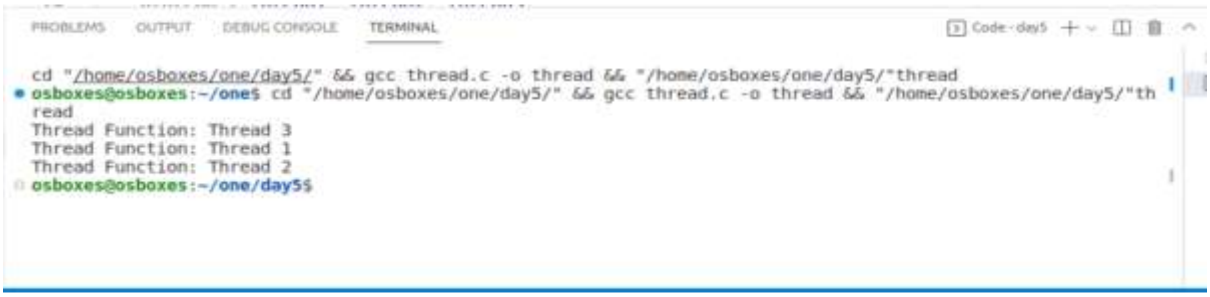


```c
#include <stdio.h>
#include <pthread.h>

pthread_mutex_t mutex;
void *thread_function(void *arg)
{
    // Acquire a lock before printing data
    pthread_mutex_lock(&mutex);

    printf("Thread Function: %s\n", (char *)arg);

    // Release the lock
    pthread_mutex_unlock(&mutex);
    return NULL;
}

int main()
{
    pthread_t thread1, thread2, thread3;
    char *msg1 = "Thread 1";
    char *msg2 = "Thread 2";
    char *msg3 = "Thread 3";

    // Initialize the mutex
    pthread_mutex_init(&mutex, NULL);

    // Create the threads
    pthread_create(&thread1, NULL, thread_function, (void *)msg1);
    pthread_create(&thread2, NULL, thread_function, (void *)msg2);
    pthread_create(&thread3, NULL, thread_function, (void
    *)msg3);

    // Join the threads
    pthread_join(thread1, NULL);
    pthread_join(thread2, NULL);
    pthread_join(thread3, NULL);

    // Destroy the mutex
    pthread_mutex_destroy(&mutex);
```



```c
    return NULL;
}

int main()
{
    pthread_t thread1, thread2, thread3;
    char *msg1 = "Thread 1";
    char *msg2 = "Thread 2";
    char *msg3 = "Thread 3";

    // Initialize the mutex
    pthread_mutex_init(&mutex, NULL);

    // Create the threads
    pthread_create(&thread1, NULL, thread_function, (void *)msg1);
    pthread_create(&thread2, NULL, thread_function, (void *)msg2);
    pthread_create(&thread3, NULL, thread_function, (void
    *)msg3);

    // Join the threads
    pthread_join(thread1, NULL);
    pthread_join(thread2, NULL);
    pthread_join(thread3, NULL);

    // Destroy the mutex
    pthread_mutex_destroy(&mutex);

    return 0;
}
```

```
Thread Function: Thread 2
osboxes@osboxes:~/one/day5$ cd "/home/osboxes/one/day5/" && gcc thread.c -o thread && "/home/osboxes/one/day5/"thread
Thread Function: Thread 3
Thread Function: Thread 2
Thread Function: Thread 1
osboxes@osboxes:~/one/day5$
```



```
cd "/home/osboxes/one/day5/" && gcc thread.c -o thread && "/home/osboxes/one/day5/"thread
osboxes@osboxes:~/one$ cd "/home/osboxes/one/day5/" && gcc thread.c -o thread && "/home/osboxes/one/day5/"th
read
Thread Function: Thread 3
Thread Function: Thread 1
Thread Function: Thread 2
osboxes@osboxes:~/one/day5$
```