1. **Names of the Students:** Priyanshu Jain, Sayali Patil, Shadi Davari
2. **Usage:**

```
[sayali.patil@linuxlab010 test_programs]$ make
gcc -Wall -o dense_mm dense_mm.c
gcc -Wall -o parallel_dense_mm parallel_dense_mm.c -fopenmp
gcc -Wall -o sort sort.c
gcc -Wall -o sing sing.c


[sayali.patil@linuxlab010 test_programs]$ gcc dense_mm.c -o dense_mm
[sayali.patil@linuxlab010 test_programs]$ ./dense_mm
Usage: ./dense_mm <size of matrices>

[sayali.patil@linuxlab010 test_programs]$ gcc parallel_dense_mm.c -o parallel_dense_mm
[sayali.patil@linuxlab010 test_programs]$ ./parallel_dense_mm
Usage: ./dense_mm <size of matrices>

[sayali.patil@linuxlab010 test_programs]$ gcc sing.c -o sing
[sayali.patil@linuxlab010 test_programs]$ ./sing
Usage: ./sing <number of verses>

[sayali.patil@linuxlab010 test_programs]$ gcc sort.c -o sort
[sayali.patil@linuxlab010 test_programs]$ ./sort
Usage: ./sort <size of array to sort>
```

- **dense_mm.c:** Given a number for size of matrix. Generates two square matrices of that size and matrix multiplies them.
- **parallel_dense_mm.**c: This program generates and multiplies the generated matrices parallelly by using multiple threads or multiple processor cores.
- **sing.c:** Given a number for number of verses. Sings a verse from Lord of the Rings that many times.
- **sort.c:** Given a number for size of an array. Generates the array of that size with random numbers and sorts it.

3. **Results:**

```
[sayali.patil@linuxlab010 test_programs]$ time ./dense_mm 2
Generating matrices...
Multiplying matrices...
Multiplication done!

real    0m0.002s
user    0m0.002s
sys     0m0.000s

[sayali.patil@linuxlab010 test_programs]$ time ./parallel_dense_mm 2
Generating matrices...
Multiplying matrices...
Multiplication done!

real    0m0.002s
user    0m0.001s
sys     0m0.001s
```

```
[sayali.patil@linuxlab010 test_programs]$ time ./sing 2
The Road goes ever on and on
Down from the door where it began.
Now far ahead the Road has gone,
And I must follow, if I can,
Pursuing it with eager feet,
Until it joins some larger way,
Where many paths and errands meet.
And whither then? I cannot say.
-Bilbo, The Lord of the Rings
The Road goes ever on and on
Down from the door where it began.
Now far ahead the Road has gone,
And I must follow, if I can,
Pursuing it with eager feet,
Until it joins some larger way,
Where many paths and errands meet.
And whither then? I cannot say.
-Bilbo, The Lord of the Rings

real    0m0.002s
user    0m0.000s
sys     0m0.002s

[sayali.patil@linuxlab010 test_programs]$ time ./sort 2
Generating array...
Sorting array...
Sort done!

real    0m0.002s
user    0m0.000s
sys     0m0.002s

[sayali.patil@linuxlab010 test_programs]$ time ./dense_mm 3
Generating matrices...
Multiplying matrices...
Multiplication done!

real    0m0.002s
user    0m0.000s
sys     0m0.002s
[sayali.patil@linuxlab010 test_programs]$ time ./parallel_dense_mm 3
Generating matrices...
Multiplying matrices...
Multiplication done!

real    0m0.002s
user    0m0.000s
sys     0m0.001s

[sayali.patil@linuxlab010 test_programs]$ time ./sing 3
```

The Road goes ever on and on
Down from the door where it began.
Now far ahead the Road has gone,
And I must follow, if I can,
Pursuing it with eager feet,
Until it joins some larger way,
Where many paths and errands meet.
And whither then? I cannot say.
-Bilbo, The Lord of the Rings
The Road goes ever on and on
Down from the door where it began.
Now far ahead the Road has gone,
And I must follow, if I can,
Pursuing it with eager feet,
Until it joins some larger way,
Where many paths and errands meet.
And whither then? I cannot say.
-Bilbo, The Lord of the Rings
The Road goes ever on and on
Down from the door where it began.
Now far ahead the Road has gone,
And I must follow, if I can,
Pursuing it with eager feet,
Until it joins some larger way,
Where many paths and errands meet.
And whither then? I cannot say.
-Bilbo, The Lord of the Rings

real 0m0.002s
user 0m0.000s
sys 0m0.002s

[sayali.patil@linuxlab010 test_programs]$ time ./sort 3
Generating array...
Sorting array...
Sort done!

real 0m0.002s
user 0m0.000s
sys 0m0.002s

[sayali.patil@linuxlab010 test_programs]$ time ./sort 8
Generating array...
Sorting array...
Sort done!

real 0m0.002s
user 0m0.002s
sys 0m0.000s

[sayali.patil@linuxlab010 test_programs]$ time ./sing 8
The Road goes ever on and on

Down from the door where it began.
Now far ahead the Road has gone,
And I must follow, if I can,
Pursuing it with eager feet,
Until it joins some larger way,
Where many paths and errands meet.
And whither then? I cannot say.
-Bilbo, The Lord of the Rings
The Road goes ever on and on
Down from the door where it began.
Now far ahead the Road has gone,
And I must follow, if I can,
Pursuing it with eager feet,
Until it joins some larger way,
Where many paths and errands meet.
And whither then? I cannot say.
-Bilbo, The Lord of the Rings
The Road goes ever on and on
Down from the door where it began.
Now far ahead the Road has gone,
And I must follow, if I can,
Pursuing it with eager feet,
Until it joins some larger way,
Where many paths and errands meet.
And whither then? I cannot say.
-Bilbo, The Lord of the Rings
The Road goes ever on and on
Down from the door where it began.
Now far ahead the Road has gone,
And I must follow, if I can,
Pursuing it with eager feet,
Until it joins some larger way,
Where many paths and errands meet.
And whither then? I cannot say.
-Bilbo, The Lord of the Rings
The Road goes ever on and on
Down from the door where it began.
Now far ahead the Road has gone,
And I must follow, if I can,
Pursuing it with eager feet,
Until it joins some larger way,
Where many paths and errands meet.
And whither then? I cannot say.
-Bilbo, The Lord of the Rings
The Road goes ever on and on
Down from the door where it began.
Now far ahead the Road has gone,
And I must follow, if I can,
Pursuing it with eager feet,
Until it joins some larger way,
Where many paths and errands meet.
And whither then? I cannot say.

-Bilbo, The Lord of the Rings
The Road goes ever on and on
Down from the door where it began.
Now far ahead the Road has gone,
And I must follow, if I can,
Pursuing it with eager feet,
Until it joins some larger way,
Where many paths and errands meet.
And whither then? I cannot say.
-Bilbo, The Lord of the Rings
The Road goes ever on and on
Down from the door where it began.
Now far ahead the Road has gone,
And I must follow, if I can,
Pursuing it with eager feet,
Until it joins some larger way,
Where many paths and errands meet.
And whither then? I cannot say.
-Bilbo, The Lord of the Rings

real 0m0.003s
user 0m0.000s
sys 0m0.002s

[sayali.patil@linuxlab010 test_programs]$ time ./parallel_dense_mm 8
Generating matrices...
Multiplying matrices...
Multiplication done!

real 0m0.002s
user 0m0.000s
sys 0m0.002s

[sayali.patil@linuxlab010 test_programs]$ time ./dense_mm 8
Generating matrices...
Multiplying matrices...
Multiplication done!

real 0m0.002s
user 0m0.000s
sys 0m0.002s

4. **Three different pieces of timing information:**

- **Real:** It is value of a wall clock time – total time elapsed from the start to finish of the process call. It includes the time slices used by other processes and time the process spends being blocked by other processes (for example if it is waiting for I/O to complete). That means it is the elapsed real time between invocation and termination
- **User:** It is the amount of CPU time spent in user-mode code within the process (outside the kernel). It stands for the actual CPU time used in executing the process. Time slices used by other processes and time the process spends blocked does not count towards this. User CPU time is the sum of the tms_utime and tms_cutime values in a struct tms as returned by times(2).

- **Sys:** It is the amount of CPU time spent in the kernel within the process. This means the time spent processing the system calls. System time is the sum of the tms_stime and tms_cstime values in a struct tms as returned by times(2).

5. **Comparison:**

   [sayali.patil@linuxlab010 test_programs]$ time ./dense_mm 1000
   Generating matrices...
   Multiplying matrices...
   Multiplication done!

   real 0m5.146s
   user 0m5.134s
   sys 0m0.012s

   [sayali.patil@linuxlab010 test_programs]$ time ./parallel_dense_mm 1000
   Generating matrices...
   Multiplying matrices...
   Multiplication done!

   real 0m4.832s
   user 0m4.821s
   sys 0m0.011s

   In case of parallel execution there is a possibility of user time exceeding the value of real time because if the process uses multiple threads or CPU cores then the user time is the total of all the figures collected from all the child processes. As a result, even though more CPU time is spent, it is spread across multiple processors. For the non-parallel one, we don't see that happening since it is unable to make use of the multiple processors.

6. **Output:** The poem gets printed 1000 times with time values as:
   real 0m0.112s
   user 0m0.007s
   sys 0m0.041s

   For sing.c, the lines of code are way less compared to the parallel_dense_mm.c. Additionally, while the sing.c involves printing the same verse multiple times, parallel_dense_mm.c does the same computation of multiplication and addition but with different inputs. Therefore, a lot of time is also spent in moving around data in the variables and the registers in the user-code mode.

7. **Different clocks:**

- Good for userspace benchmarking:
  CLOCK_PROCESS_CPUTIME_ID because it provides high-resolution per-process timer from the CPU. This one is process-specific and high-resolution. This fits our purpose perfectly.
- Bad for userspace benchmarking:
  CLOCK_REALTIME_COARSE (since Linux 2.6.32; Linux-specific) because it is a faster but less precise version of CLOCK_REALTIME. Use when you need very fast, but not fine-grained timestamps.

It is bad because it measures system-wide time instead of process-specific time. Due to speed, it sacrifices accuracy. Also, it can be affected by discontinuous jumps in time which might give us the wrong result.

8. **Getres.c Output:**

[sayali.patil@linuxlab010 test_programs]$ ./getres
Timer: 0, Seconds: 0, Nanos: 1
Timer: 1, Seconds: 0, Nanos: 1000000
Timer: 2, Seconds: 0, Nanos: 1
Timer: 3, Seconds: 0, Nanos: 1000000
Timer: 4, Seconds: 0, Nanos: 1
Timer: 5, Seconds: 0, Nanos: 1
Timer: 6, Seconds: 0, Nanos: 1
Timer: 7, Seconds: 0, Nanos: 1

In our program, the clock are in the following order: CLOCK_REALTIME, CLOCK_REALTIME_COARSE, CLOCK_MONOTONIC, CLOCK_MONOTONIC_COARSE, CLOCK_MONOTONIC_RAW, CLOCK_BOOTTIME, CLOCK_PROCESS_CPUTIME_ID, CLOCK_THREAD_CPUTIME_ID. The results tell us that all clocks except the ones ending with _COARSE have a precision of 1nsec. However, the clocks ending with _COARSE have low precision of 1000000 nsec. They have high speed at the cost of low precision.

9. **Difference between CLOCK_MONOTONIC and CLOCK_MONOTONIC_RAW:**

CLOCK_MONOTONIC never goes through discontinuities because of time adjustments by NTP but it also doesn't change its frequency of oscillation based NTP's error learning mechanism. On the other hand, CLOCK_MONOTONIC_RAW is not controlled by NTP time adjustments but rather it stands for a simple local oscillator. If we need to build a time synchronization algorithm that doesn't get affected by NTP then we can use CLOCK_MONOTONIC_RAW.

10. **Clock_gettime:**

This is how we obtained the value:

```
#include <stdio.h>
#include <unistd.h>
#include <time.h>
#include <sys/time.h>
#include <sys/resource.h>

int main()
{
	struct timespec start, finish;
	clock_gettime(CLOCK_MONOTONIC, &start);
	clock_gettime(CLOCK_MONOTONIC, &finish);

	long sec = finish.tv_sec - start.tv_sec;
	long nanosec = finish.tv_nsec - start.tv_nsec;

	if (start.tv_nsec > finish.tv_nsec)
```

```
{
        --sec;
        nanosec += 1000000000;
}
printf("Value of sec: %ld\n", sec);
printf("Value of nanosec: %ld\n", nanosec);
printf("Total seconds: %e\n", (double)sec + (double)nanosec/ (double) 1000000000);


}
```

Output:

```
[sayali.patil@linuxlab010 test_programs]$ ./clock_gettime
Value of sec: 0
Value of nanosec: 195
Total seconds: 1.950000e-07
```

We just called the function twice in a row. The difference between the two, should be time taken for one call to complete.

11. Time_parallel_dense_mm.c Output:

```
Generating matrices...
Multiplying matrices...
Minimum time: 0 seconds, 11396343 nanoseconds
Maximum time: 0 seconds, 11396343 nanoseconds
Average time: 0 seconds, 11396343 nanoseconds
Multiplication done!

real    0m0.014s
user    0m0.013s
sys     0m0.002s
```

We think that the average time is a reasonable estimation of a common case running time as we always define the best case and worst case scenarios for our program's running time as we cannot predict the running time accurately because it's not only based on the size of the input but also the type of the input. As average time is defined to be the average running time of a program considering all types of inputs it can be thought of as a more realistic way of measuring running times.