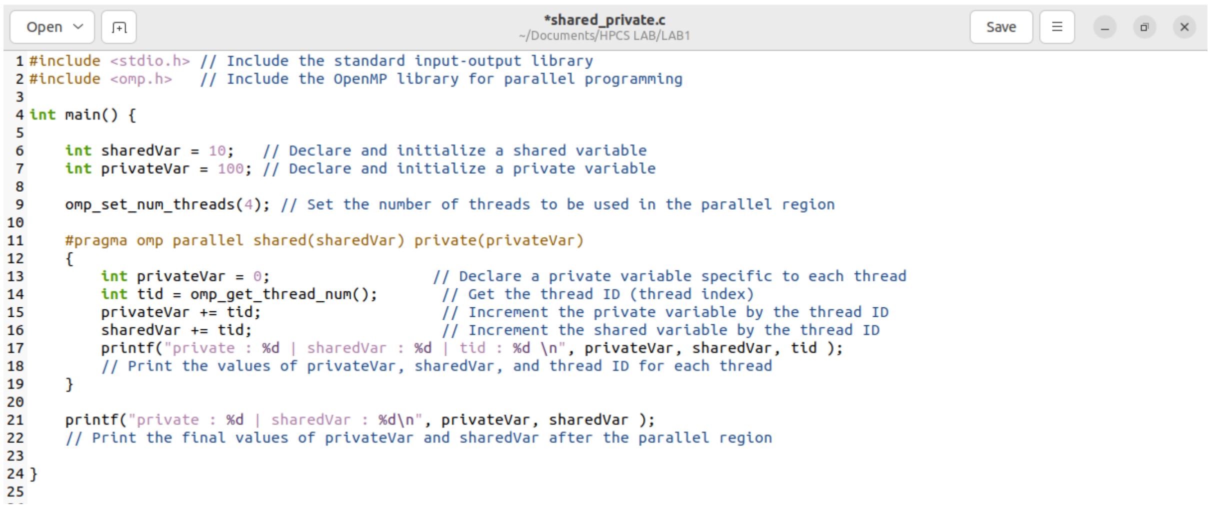
HPCS

LAB 1 WEEK 1

OpenMP PROGRAMMING

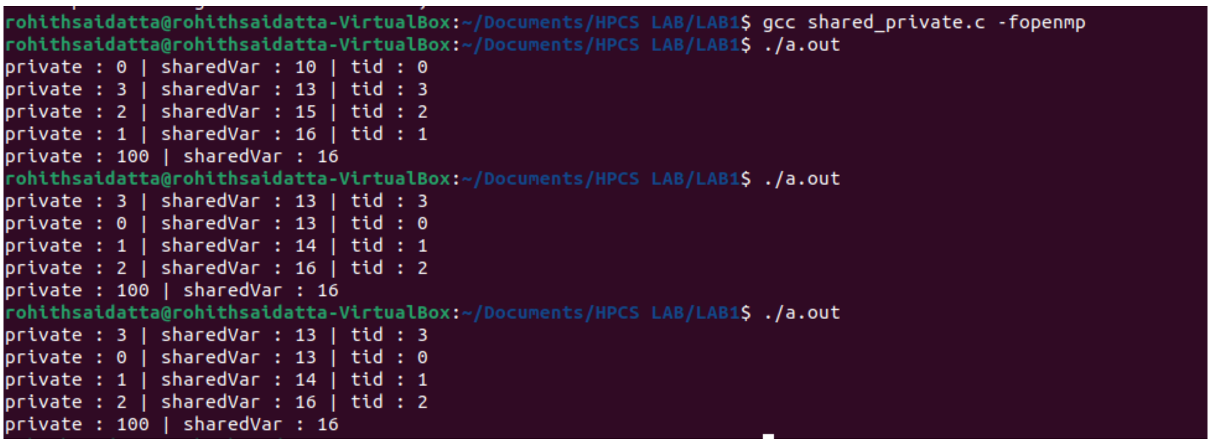
1. Write an OpenMP program to use variables as shared or private

   Code:



```c
#include <stdio.h> // Include the standard input-output library
#include <omp.h>   // Include the OpenMP library for parallel programming

int main() {

    int sharedVar = 10;   // Declare and initialize a shared variable
    int privateVar = 100; // Declare and initialize a private variable

    omp_set_num_threads(4); // Set the number of threads to be used in the parallel region

    #pragma omp parallel shared(sharedVar) private(privateVar)
    {
        int privateVar = 0;                   // Declare a private variable specific to each thread
        int tid = omp_get_thread_num();        // Get the thread ID (thread index)
        privateVar += tid;                     // Increment the private variable by the thread ID
        sharedVar += tid;                      // Increment the shared variable by the thread ID
        printf("private : %d | sharedVar : %d | tid : %d \n", privateVar, sharedVar, tid );
        // Print the values of privateVar, sharedVar, and thread ID for each thread
    }

    printf("private : %d | sharedVar : %d\n", privateVar, sharedVar );
    // Print the final values of privateVar and sharedVar after the parallel region

}
```

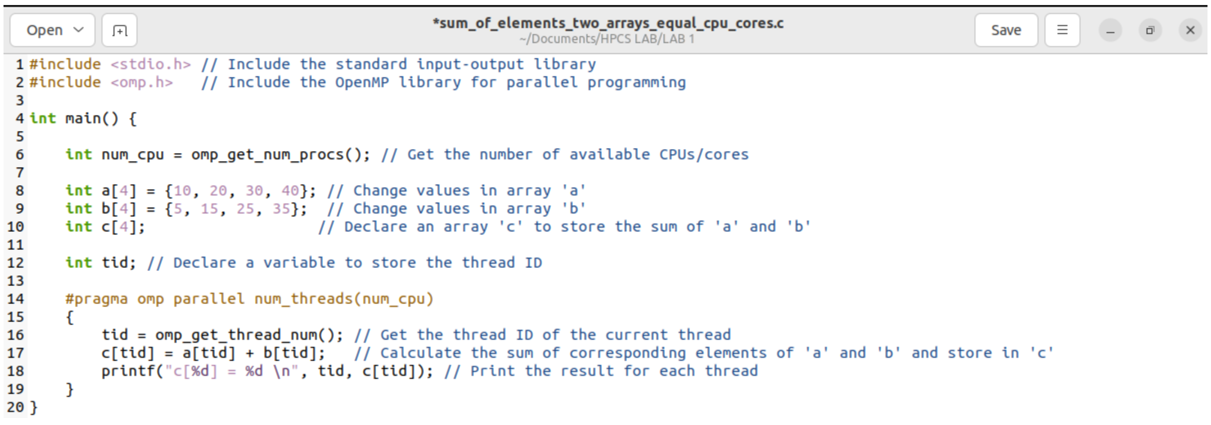
Results:



```
rohithsaidatta@rohithsaidatta-VirtualBox:~/Documents/HPCS LAB/LAB1$ gcc shared_private.c -fopenmp
rohithsaidatta@rohithsaidatta-VirtualBox:~/Documents/HPCS LAB/LAB1$ ./a.out
private : 0 | sharedVar : 10 | tid : 0
private : 3 | sharedVar : 13 | tid : 3
private : 2 | sharedVar : 15 | tid : 2
private : 1 | sharedVar : 16 | tid : 1
private : 100 | sharedVar : 16
rohithsaidatta@rohithsaidatta-VirtualBox:~/Documents/HPCS LAB/LAB1$ ./a.out
private : 3 | sharedVar : 13 | tid : 3
private : 0 | sharedVar : 13 | tid : 0
private : 1 | sharedVar : 14 | tid : 1
private : 2 | sharedVar : 16 | tid : 2
private : 100 | sharedVar : 16
rohithsaidatta@rohithsaidatta-VirtualBox:~/Documents/HPCS LAB/LAB1$ ./a.out
private : 3 | sharedVar : 13 | tid : 3
private : 0 | sharedVar : 13 | tid : 0
private : 1 | sharedVar : 14 | tid : 1
private : 2 | sharedVar : 16 | tid : 2
private : 100 | sharedVar : 16
```

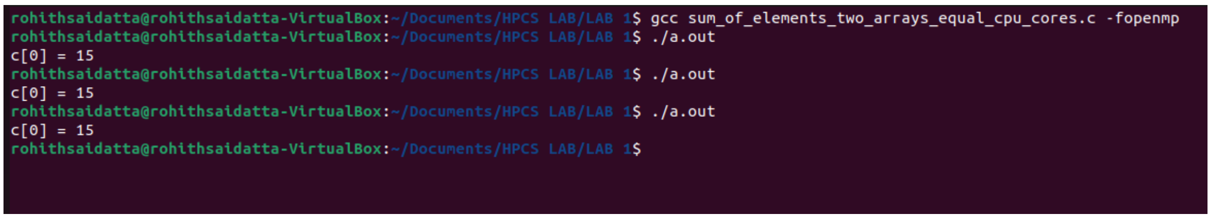
2 Write an OpenMP program to sum the respective elements of the two arrays

i. using a number of threads equal to the number of CPU cores.
   Code:


Open
Save
*sum_of_elements_two_arrays_equal_cpu_cores.c
~/Documents/HPCS LAB/LAB 1

```c
#include <stdio.h> // Include the standard input-output library
#include <omp.h>   // Include the OpenMP library for parallel programming

int main() {

    int num_cpu = omp_get_num_procs(); // Get the number of available CPUs/cores

    int a[4] = {10, 20, 30, 40}; // Change values in array 'a'
    int b[4] = {5, 15, 25, 35};  // Change values in array 'b'
    int c[4];                    // Declare an array 'c' to store the sum of 'a' and 'b'

    int tid; // Declare a variable to store the thread ID

    #pragma omp parallel num_threads(num_cpu)
    {
        tid = omp_get_thread_num(); // Get the thread ID of the current thread
        c[tid] = a[tid] + b[tid];   // Calculate the sum of corresponding elements of 'a' and 'b' and store in 'c'
        printf("c[%d] = %d \n", tid, c[tid]); // Print the result for each thread
    }
}
```

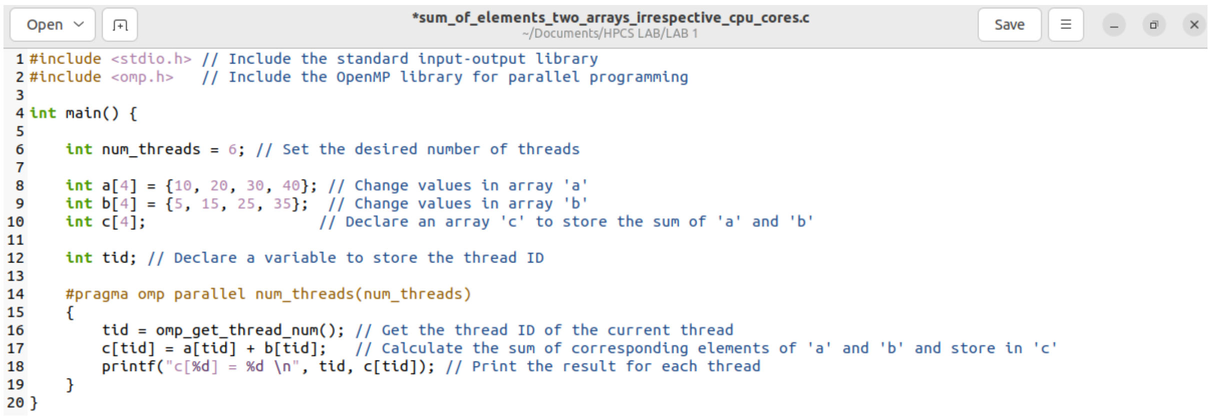
Results:



```
rohithsaidatta@rohithsaidatta-VirtualBox:~/Documents/HPCS LAB/LAB 1$ gcc sum_of_elements_two_arrays_equal_cpu_cores.c -fopenmp
rohithsaidatta@rohithsaidatta-VirtualBox:~/Documents/HPCS LAB/LAB 1$ ./a.out
c[0] = 15
rohithsaidatta@rohithsaidatta-VirtualBox:~/Documents/HPCS LAB/LAB 1$ ./a.out
c[0] = 15
rohithsaidatta@rohithsaidatta-VirtualBox:~/Documents/HPCS LAB/LAB 1$ ./a.out
c[0] = 15
rohithsaidatta@rohithsaidatta-VirtualBox:~/Documents/HPCS LAB/LAB 1$
```

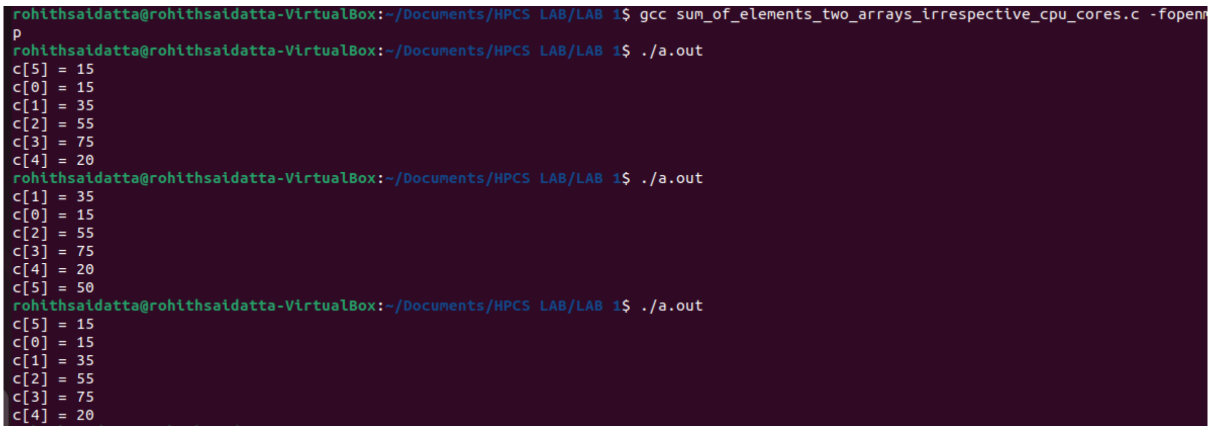
ii. using number of threads irrespective of number of CPU cores

Code:


Open
Save
*sum_of_elements_two_arrays_irrespective_cpu_cores.c
~/Documents/HPCS LAB/LAB 1

```c
#include <stdio.h> // Include the standard input-output library
#include <omp.h>   // Include the OpenMP library for parallel programming

int main() {

    int num_threads = 6; // Set the desired number of threads

    int a[4] = {10, 20, 30, 40}; // Change values in array 'a'
    int b[4] = {5, 15, 25, 35};  // Change values in array 'b'
    int c[4];                     // Declare an array 'c' to store the sum of 'a' and 'b'

    int tid; // Declare a variable to store the thread ID

    #pragma omp parallel num_threads(num_threads)
    {
        tid = omp_get_thread_num(); // Get the thread ID of the current thread
        c[tid] = a[tid] + b[tid];   // Calculate the sum of corresponding elements of 'a' and 'b' and store in 'c'
        printf("c[%d] = %d \n", tid, c[tid]); // Print the result for each thread
    }
}
```

Results:

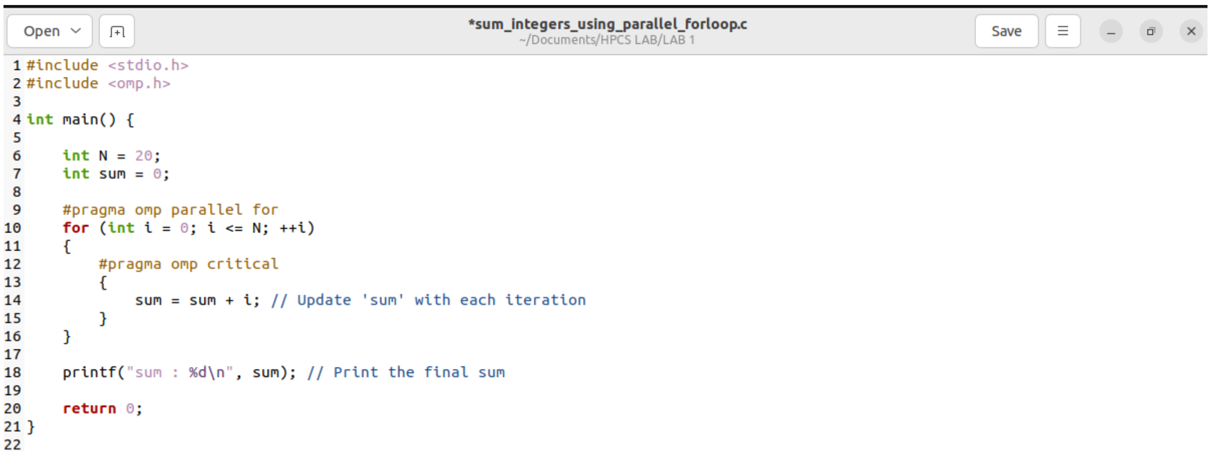


```
rohithsaidatta@rohithsaidatta-VirtualBox:~/Documents/HPCS LAB/LAB 1$ gcc sum_of_elements_two_arrays_irrespective_cpu_cores.c -fopenm
p
rohithsaidatta@rohithsaidatta-VirtualBox:~/Documents/HPCS LAB/LAB 1$ ./a.out
c[5] = 15
c[0] = 15
c[1] = 35
c[2] = 55
c[3] = 75
c[4] = 20
rohithsaidatta@rohithsaidatta-VirtualBox:~/Documents/HPCS LAB/LAB 1$ ./a.out
c[1] = 35
c[0] = 15
c[2] = 55
c[3] = 75
c[4] = 20
c[5] = 50
rohithsaidatta@rohithsaidatta-VirtualBox:~/Documents/HPCS LAB/LAB 1$ ./a.out
c[5] = 15
c[0] = 15
c[1] = 35
c[2] = 55
c[3] = 75
c[4] = 20
```

3 Write an OpenMP program to find the sum of integers from 1 to N

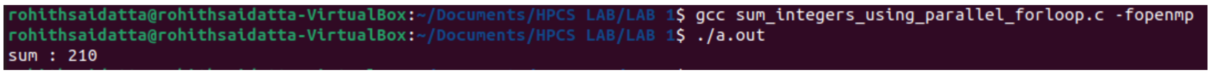
i. using parallel for loop

Code:


Open
*sum_integers_using_parallel_forloop.c
~/Documents/HPCS LAB/LAB 1
Save

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

int main() {

    int N = 20;
    int sum = 0;

    #pragma omp parallel for
    for (int i = 0; i <= N; ++i)
    {
        #pragma omp critical
        {
            sum = sum + i; // Update 'sum' with each iteration
        }
    }

    printf("sum : %d\n", sum); // Print the final sum

    return 0;
}

```

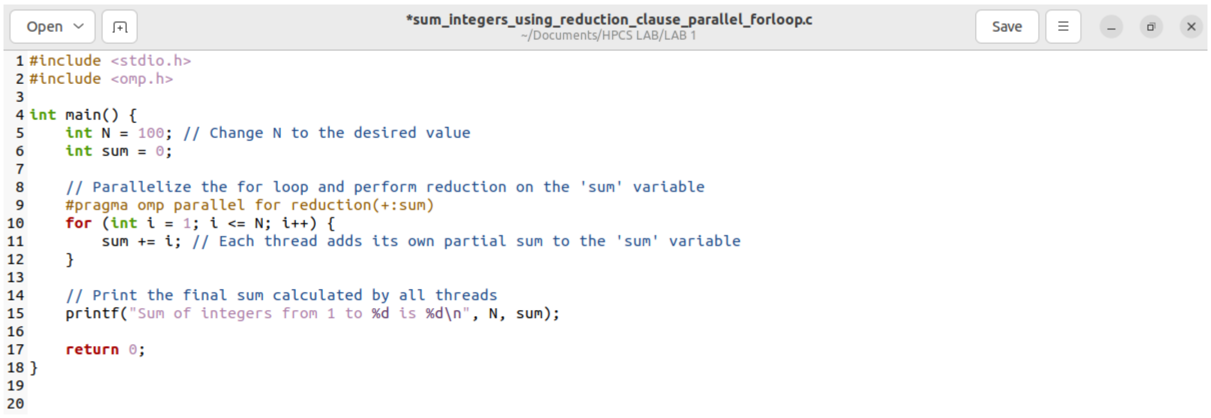
Results:



```
rohithsaidatta@rohithsaidatta-VirtualBox:~/Documents/HPCS LAB/LAB 1$ gcc sum_integers_using_parallel_forloop.c -fopenmp
rohithsaidatta@rohithsaidatta-VirtualBox:~/Documents/HPCS LAB/LAB 1$ ./a.out
sum : 210
```

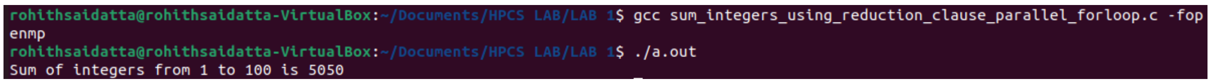
ii. using reduction clause in parallel for loop

Code:


Open
*sum_integers_using_reduction_clause_parallel_forloop.c
~/Documents/HPCS LAB/LAB 1
Save

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

int main() {
    int N = 100; // Change N to the desired value
    int sum = 0;

    // Parallelize the for loop and perform reduction on the 'sum' variable
    #pragma omp parallel for reduction(+:sum)
    for (int i = 1; i <= N; i++) {
        sum += i; // Each thread adds its own partial sum to the 'sum' variable
    }

    // Print the final sum calculated by all threads
    printf("Sum of integers from 1 to %d is %d\n", N, sum);

    return 0;
}


```

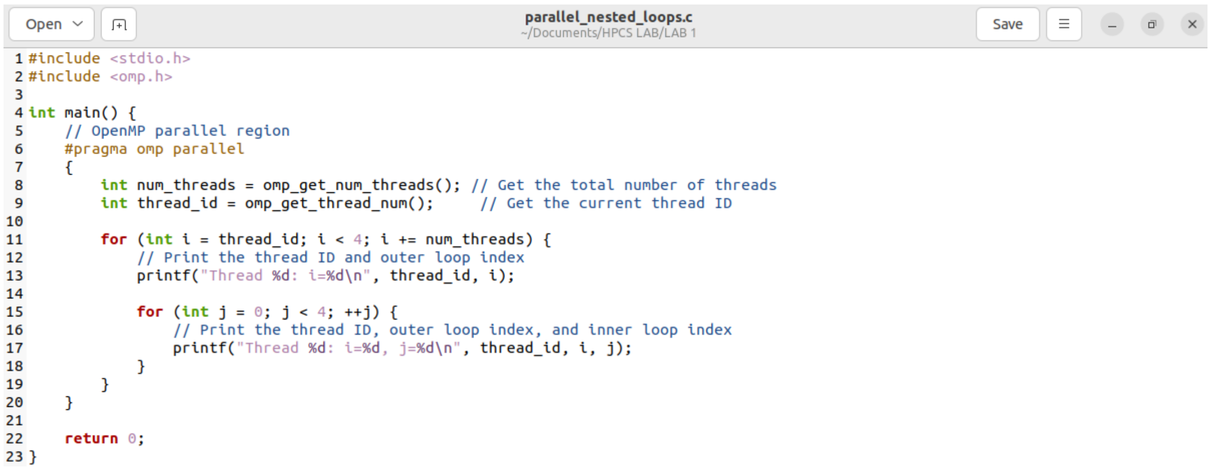
Results:



```
rohithsaidatta@rohithsaidatta-VirtualBox:~/Documents/HPCS LAB/LAB 1$ gcc sum_integers_using_reduction_clause_parallel_forloop.c -fop
enmp
rohithsaidatta@rohithsaidatta-VirtualBox:~/Documents/HPCS LAB/LAB 1$ ./a.out
Sum of integers from 1 to 100 is 5050
```

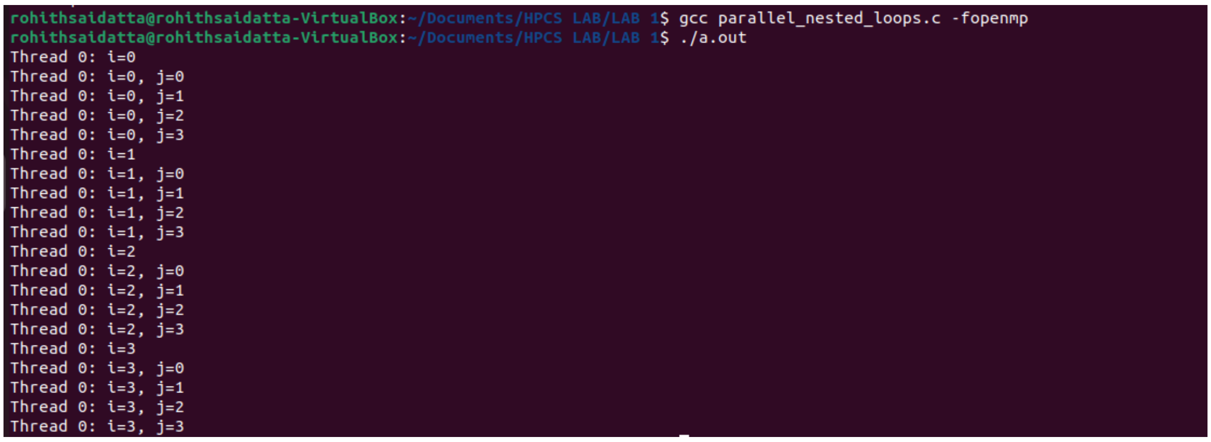
4. Write an OpenMP program to parallelize nested for loop for any program

Code:

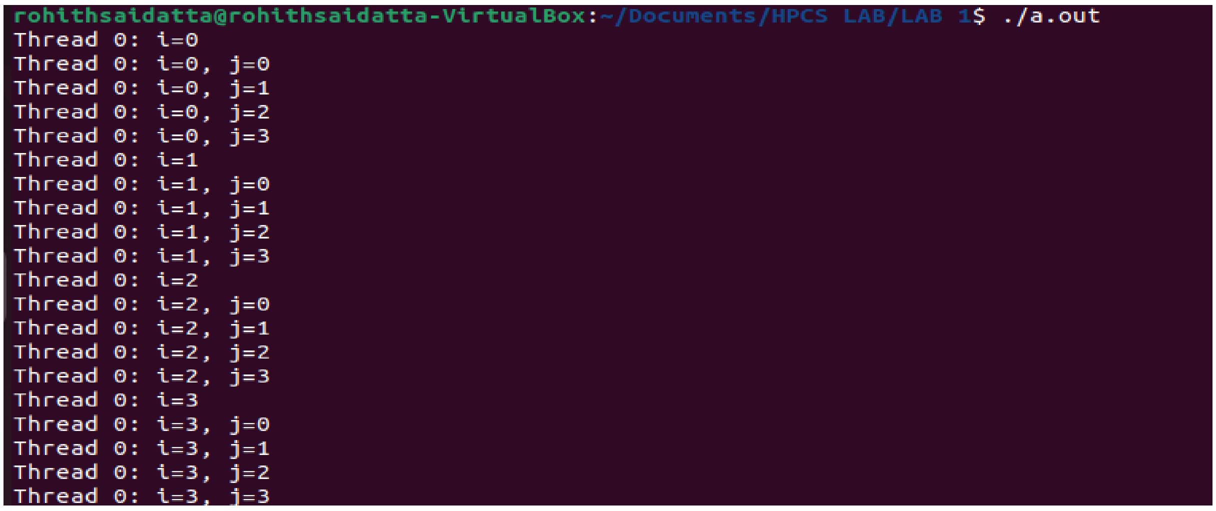


```c
#include <stdio.h>
#include <omp.h>

int main() {
    // OpenMP parallel region
    #pragma omp parallel
    {
        int num_threads = omp_get_num_threads(); // Get the total number of threads
        int thread_id = omp_get_thread_num();     // Get the current thread ID

        for (int i = thread_id; i < 4; i += num_threads) {
            // Print the thread ID and outer loop index
            printf("Thread %d: i=%d\n", thread_id, i);

            for (int j = 0; j < 4; ++j) {
                // Print the thread ID, outer loop index, and inner loop index
                printf("Thread %d: i=%d, j=%d\n", thread_id, i, j);
            }
        }
    }

    return 0;
}
```

Result:



```
rohithsaidatta@rohithsaidatta-VirtualBox:~/Documents/HPCS LAB/LAB 1$ gcc parallel_nested_loops.c -fopenmp
rohithsaidatta@rohithsaidatta-VirtualBox:~/Documents/HPCS LAB/LAB 1$ ./a.out
Thread 0: i=0
Thread 0: i=0, j=0
Thread 0: i=0, j=1
Thread 0: i=0, j=2
Thread 0: i=0, j=3
Thread 0: i=1
Thread 0: i=1, j=0
Thread 0: i=1, j=1
Thread 0: i=1, j=2
Thread 0: i=1, j=3
Thread 0: i=2
Thread 0: i=2, j=0
Thread 0: i=2, j=1
Thread 0: i=2, j=2
Thread 0: i=2, j=3
Thread 0: i=3
Thread 0: i=3, j=0
Thread 0: i=3, j=1
Thread 0: i=3, j=2
Thread 0: i=3, j=3
```



```
rohithsaidatta@rohithsaidatta-VirtualBox:~/Documents/HPCS LAB/LAB 1$ ./a.out
Thread 0: i=0
Thread 0: i=0, j=0
Thread 0: i=0, j=1
Thread 0: i=0, j=2
Thread 0: i=0, j=3
Thread 0: i=1
Thread 0: i=1, j=0
Thread 0: i=1, j=1
Thread 0: i=1, j=2
Thread 0: i=1, j=3
Thread 0: i=2
Thread 0: i=2, j=0
Thread 0: i=2, j=1
Thread 0: i=2, j=2
Thread 0: i=2, j=3
Thread 0: i=3
Thread 0: i=3, j=0
Thread 0: i=3, j=1
Thread 0: i=3, j=2
Thread 0: i=3, j=3
```