**CSE4001-Parallel and distributed computing**

**Name:** O G RAGAVI

**Reg no:** 20BCE1988

**Lab Ex:** 1-3

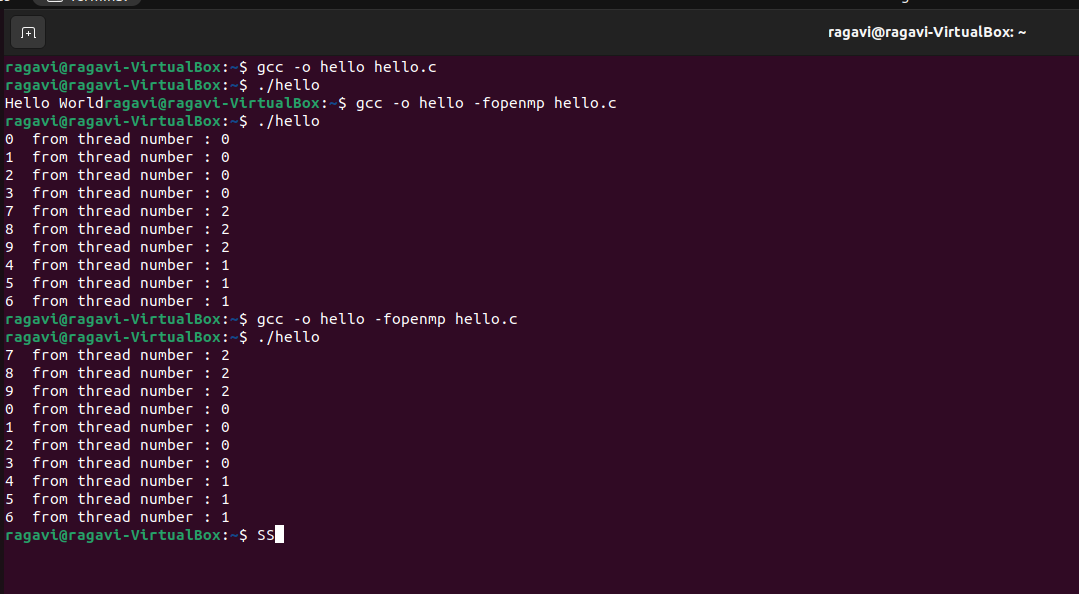
**Date:** 21.08.2022

**1.Sample program**

**Code:**



**Output:**

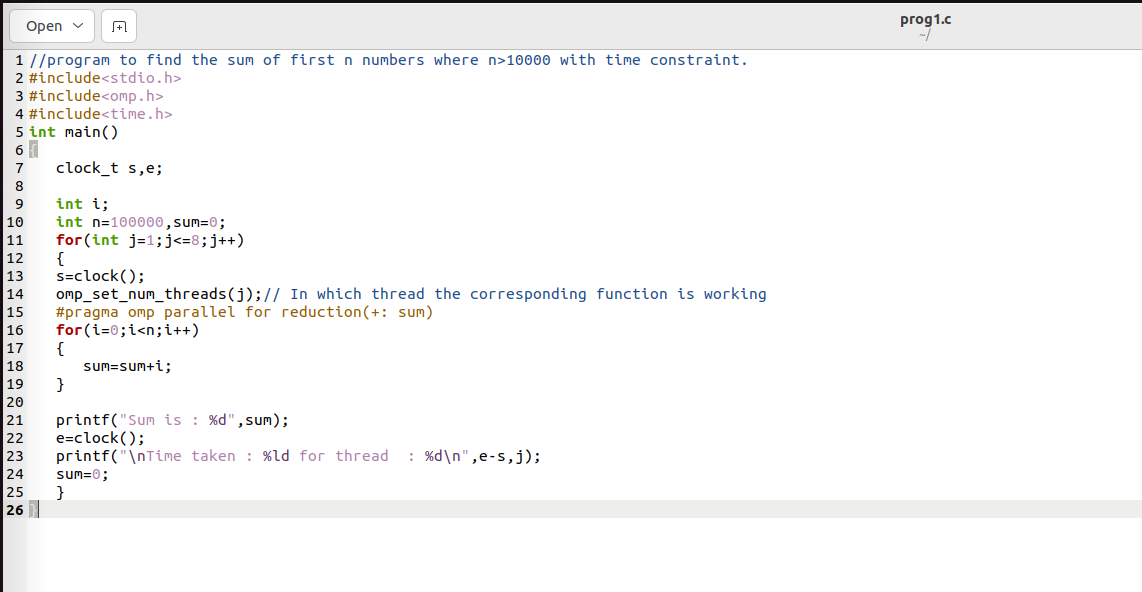


**2.Program to find the sum of first n numbers.**

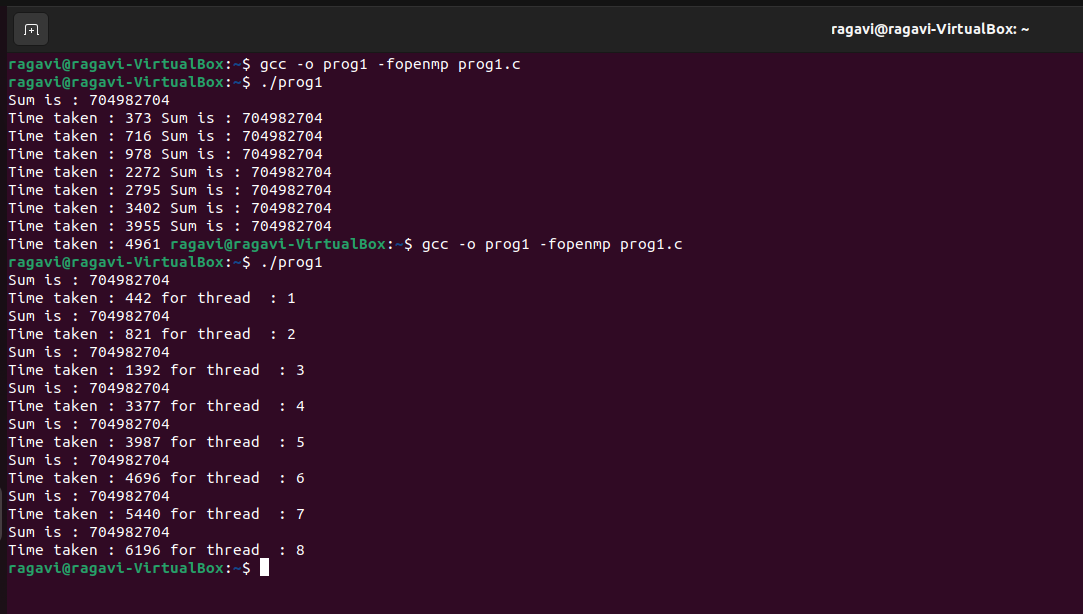
**Algorithm:**

* **Start the clock before each and every iteration and stop it before it ends.**
* **N is initialised and for every iteration no of threads is set.**
* **Using reduction , we run a for loop and increment the sum by i**
* **We end the clock and note the time difference(total time taken)**

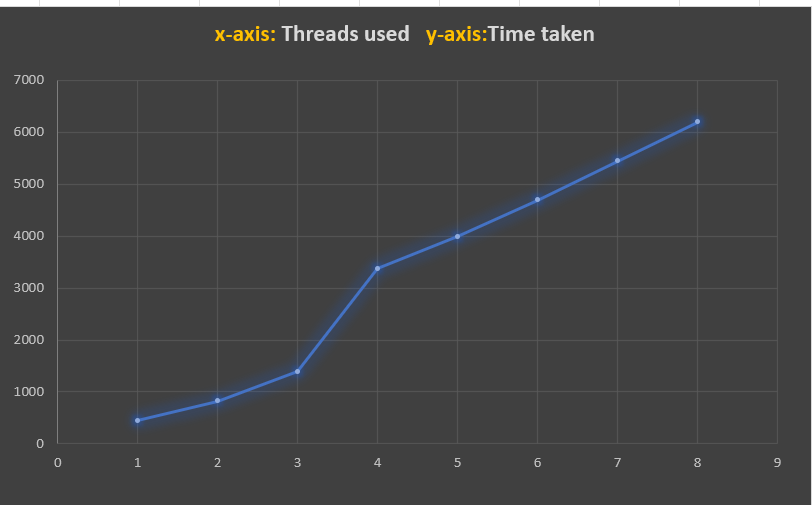
**Code:**



**Output:**



**Graph Plot:**

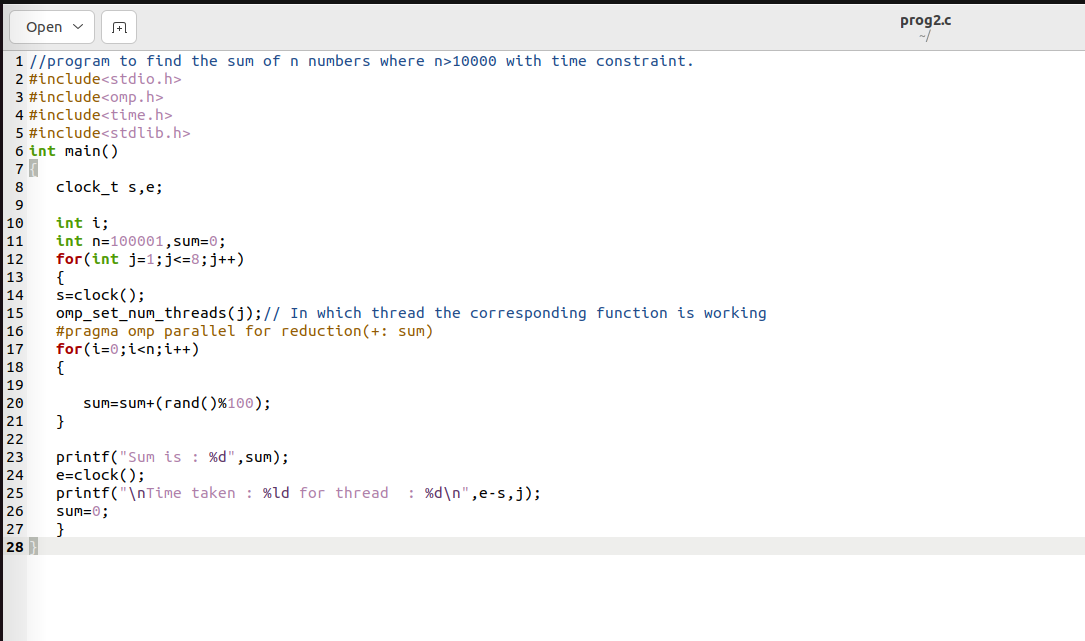


**2.Program to find the sum of n numbers: n>10000**

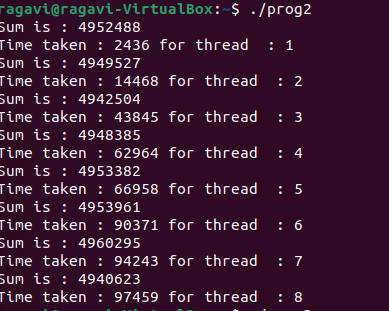
**Algorithm:**

* **Start the clock before each and every iteration and stop it before it ends.**
* **N is initialised and for every iteration no of threads is set.**
* **Using reduction , we run a for loop and increment the sum by any random number generated.(0-100)**
* **We end the clock and note the time difference(total time taken)**

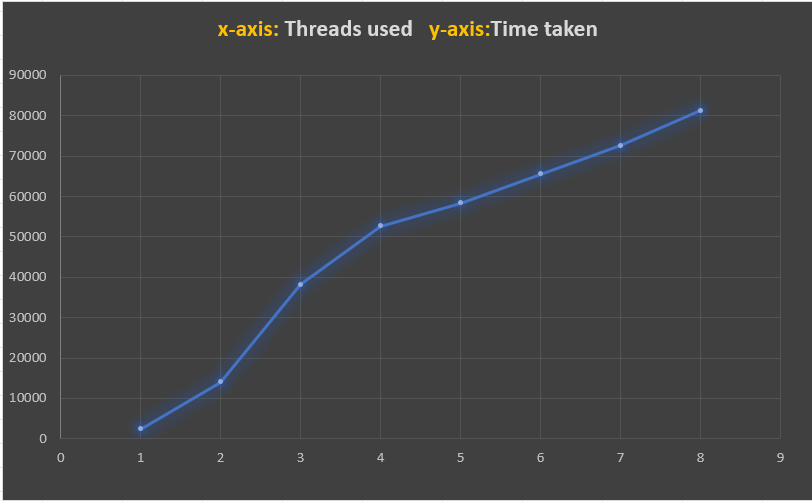
**Code:**



**Output:**



**Graph Plot:**

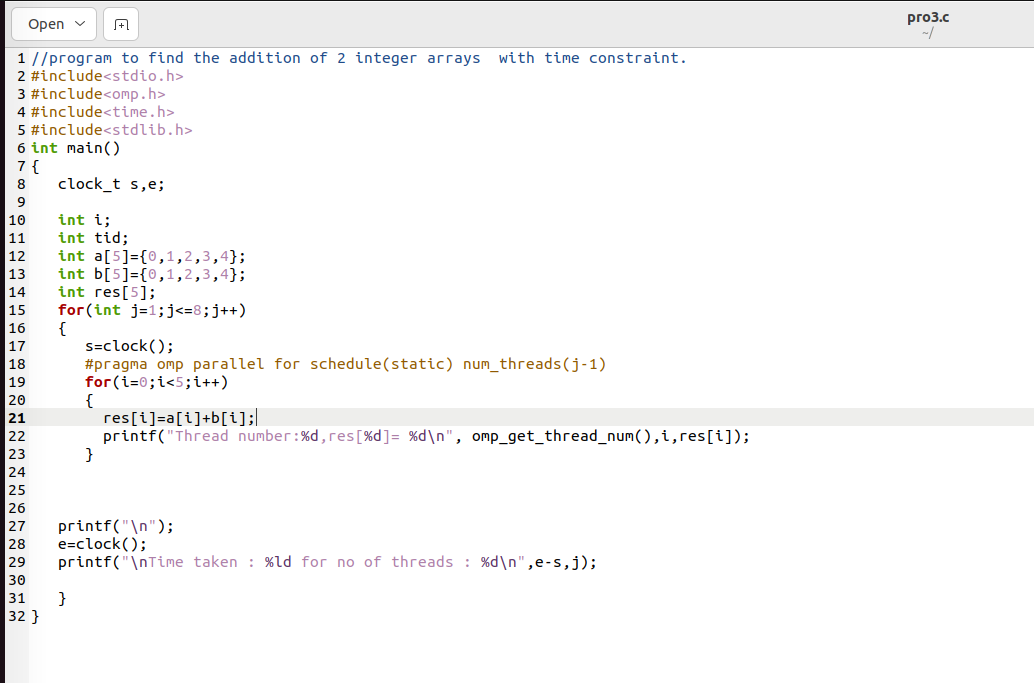


**3.Program to find the addition of 2 integer arrays :**

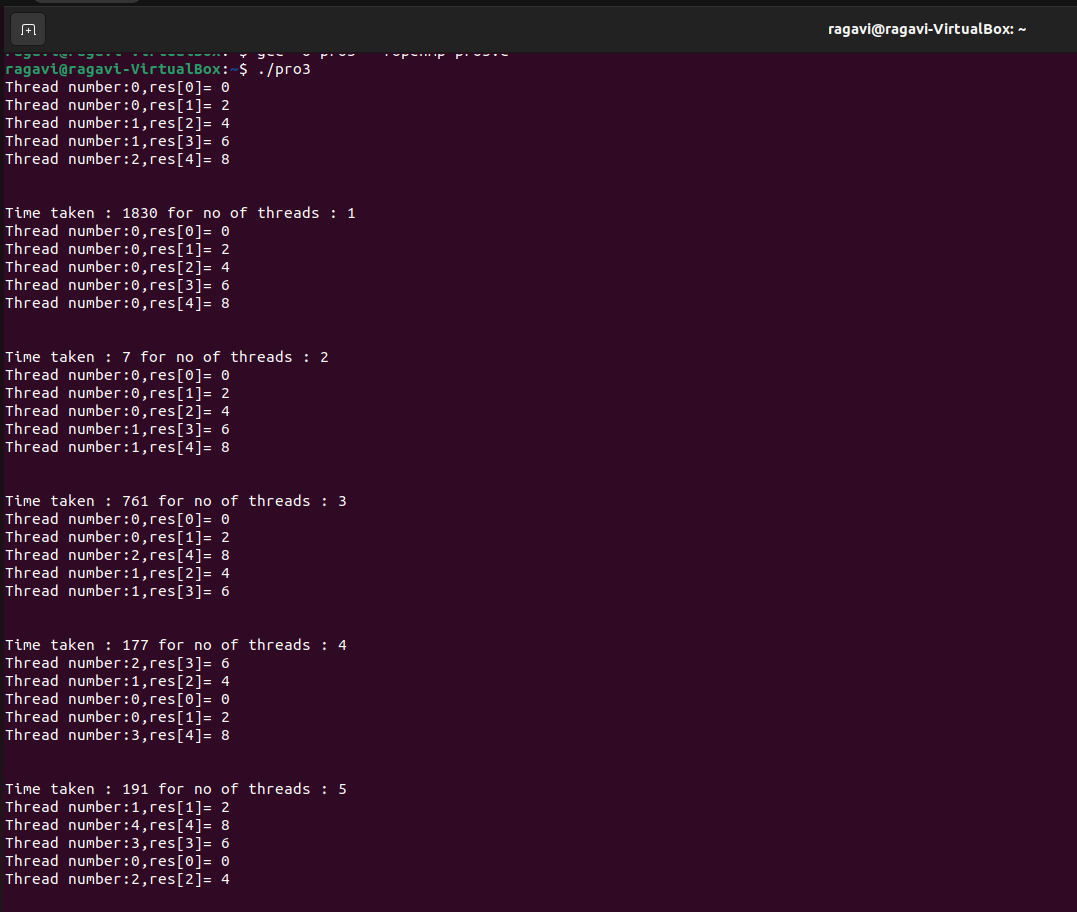
**Algorithm:**

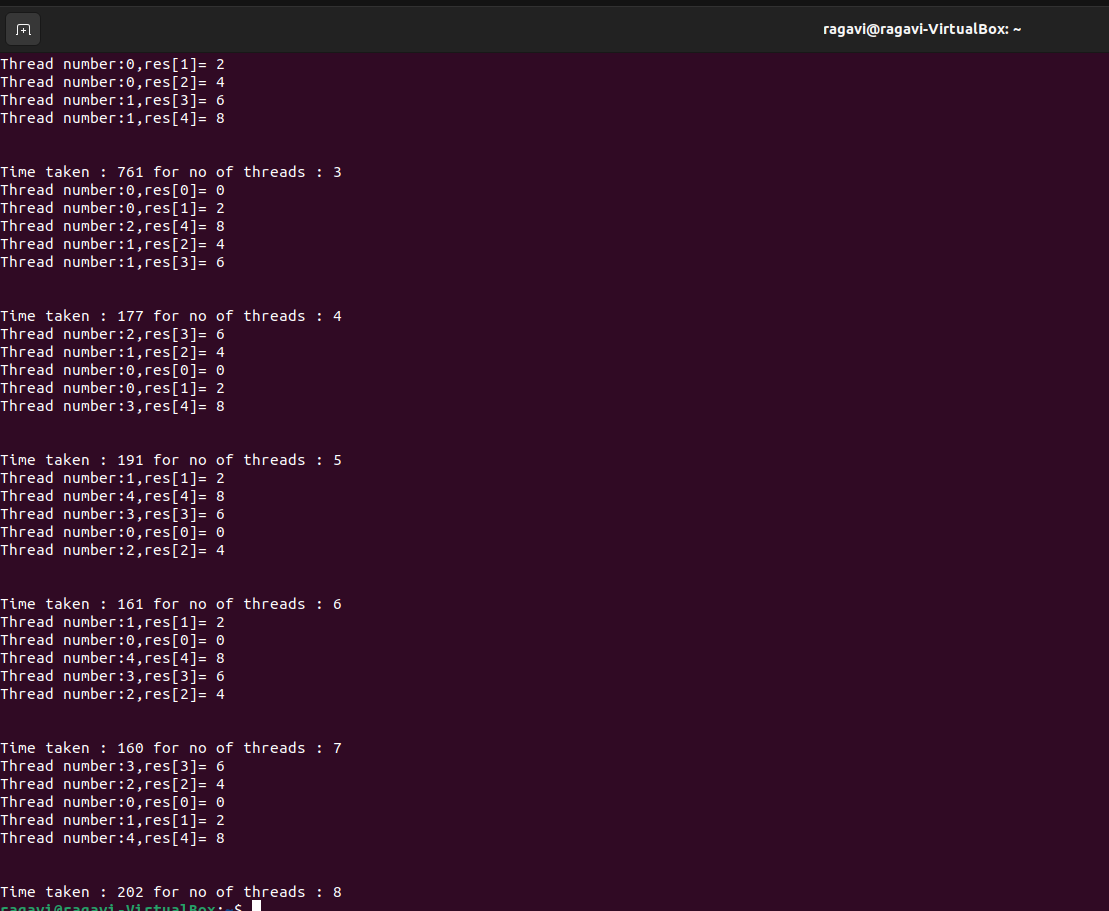
* **Initialise 2 arrays a and b.**
* **Use a for loop to get the sum at each index.**
* **Omp\_get\_num\_threads – building function that takes void input and returns no threads.**
* **Stop the clock.**

**Code:**

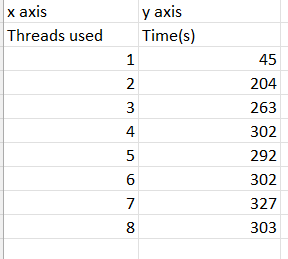


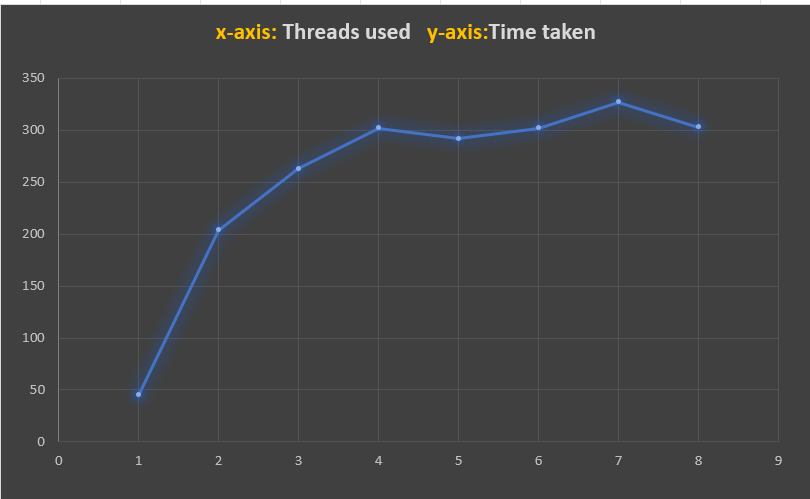
**Output:**





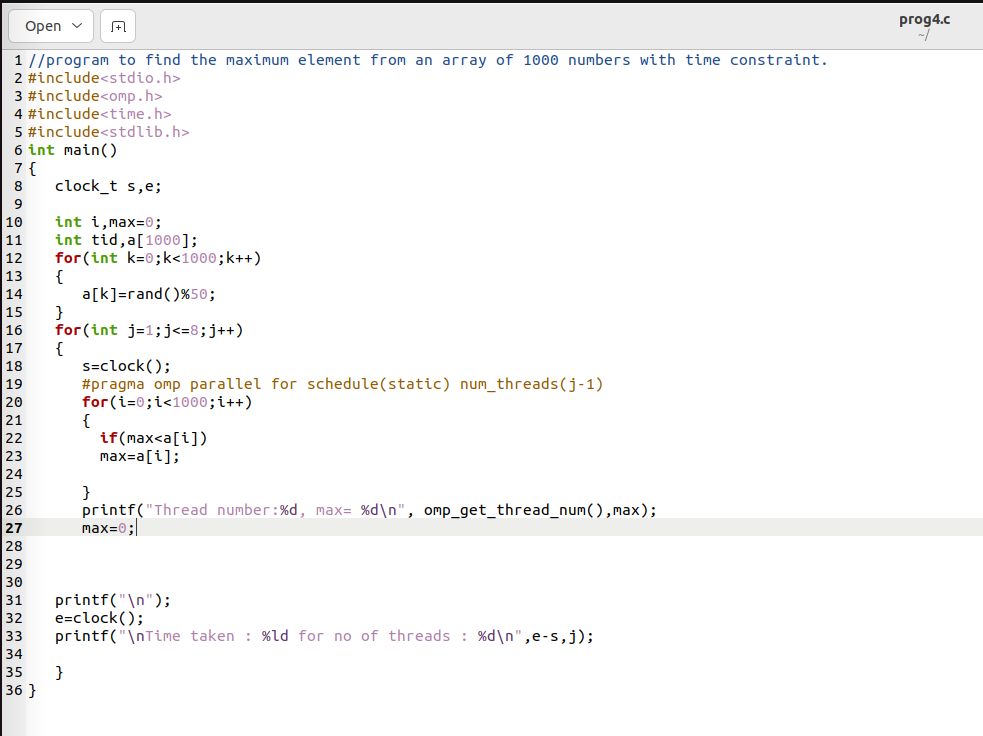
**Graph Plot:**



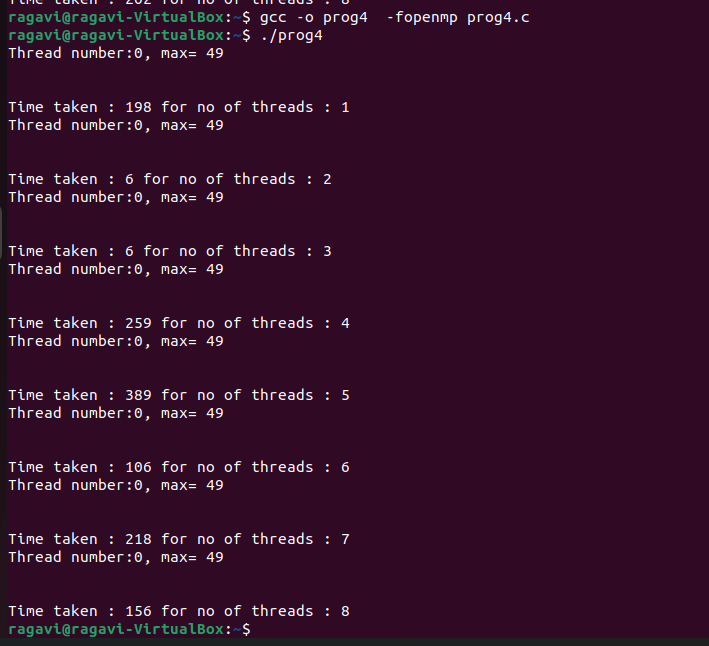


**4.Program to find the maximum element from an array of 1000 numbers.**

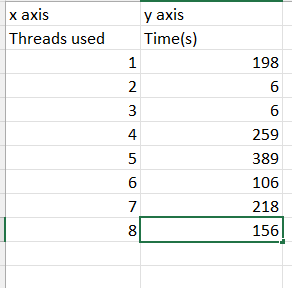
**Code:**

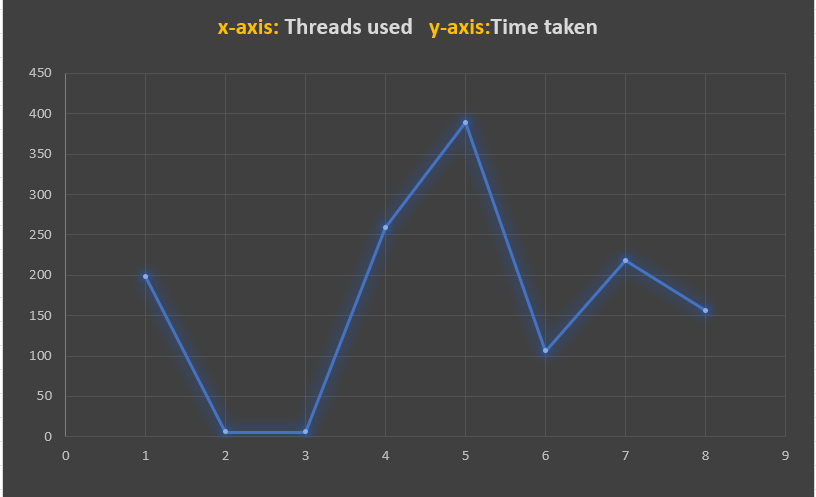


**Output:**



**Graph Plot:**



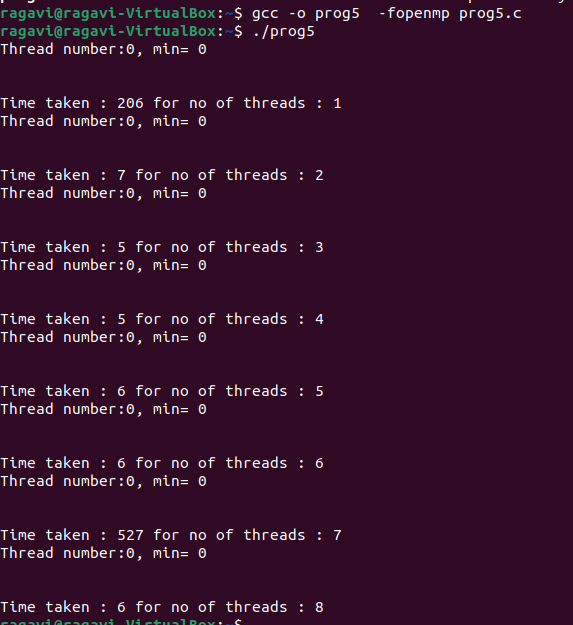


**5.Program to find the minimum element from the array of n numbers.**

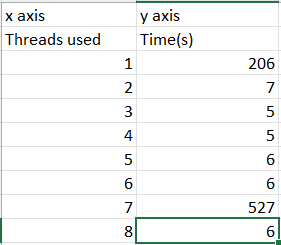
**Code:**

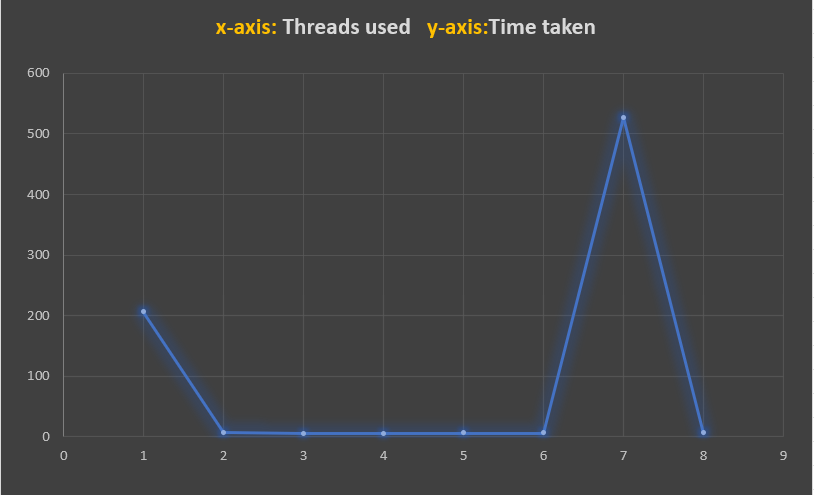


**Output:**



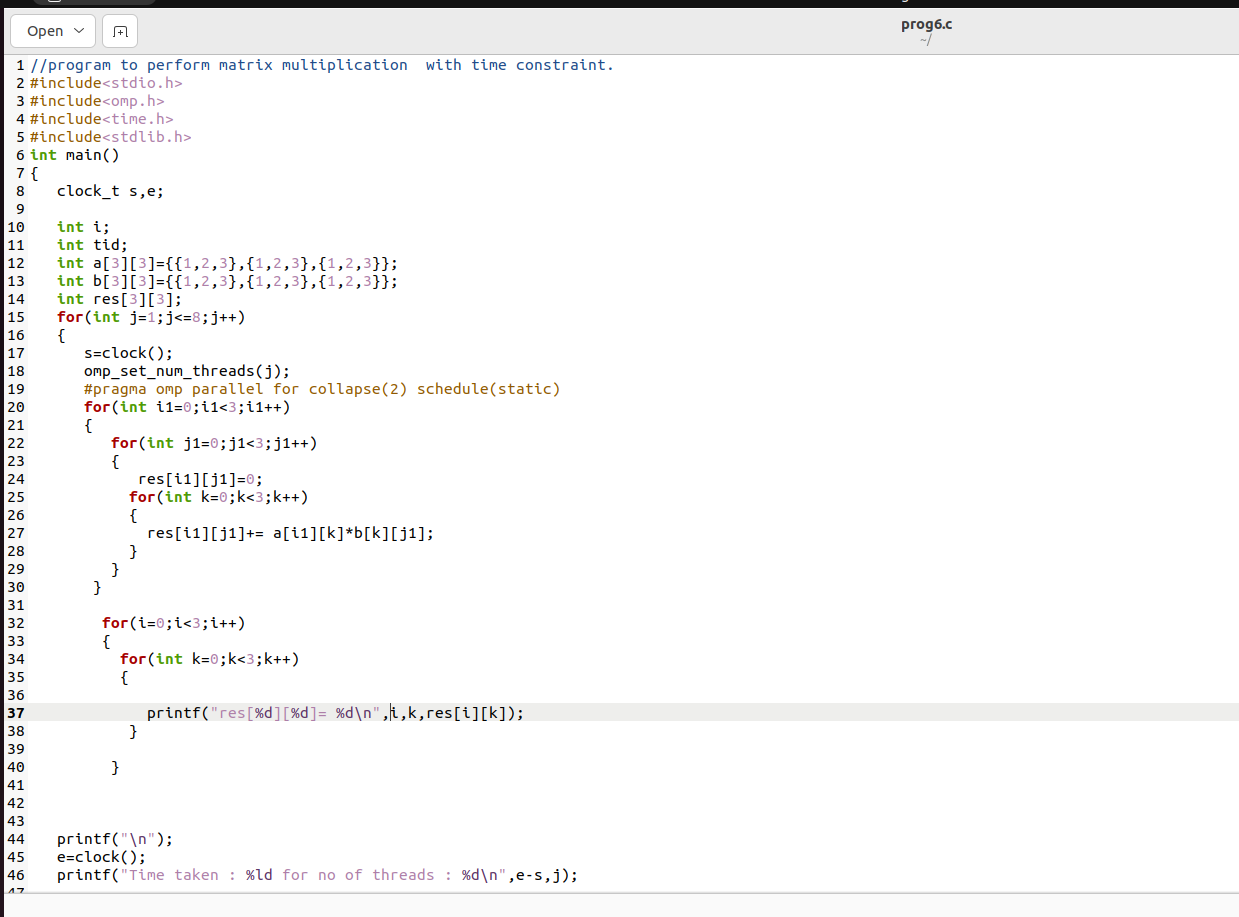
**Graph Plot:**



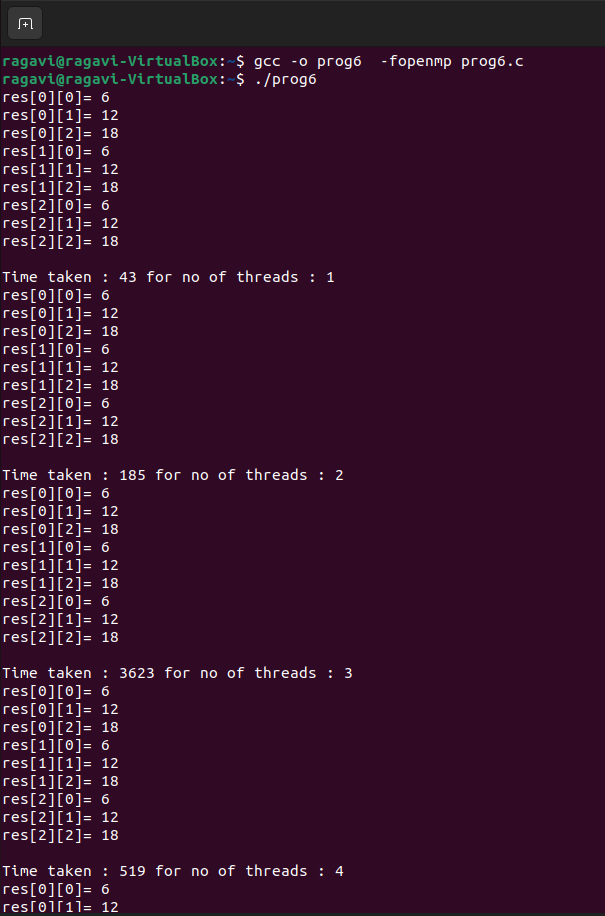


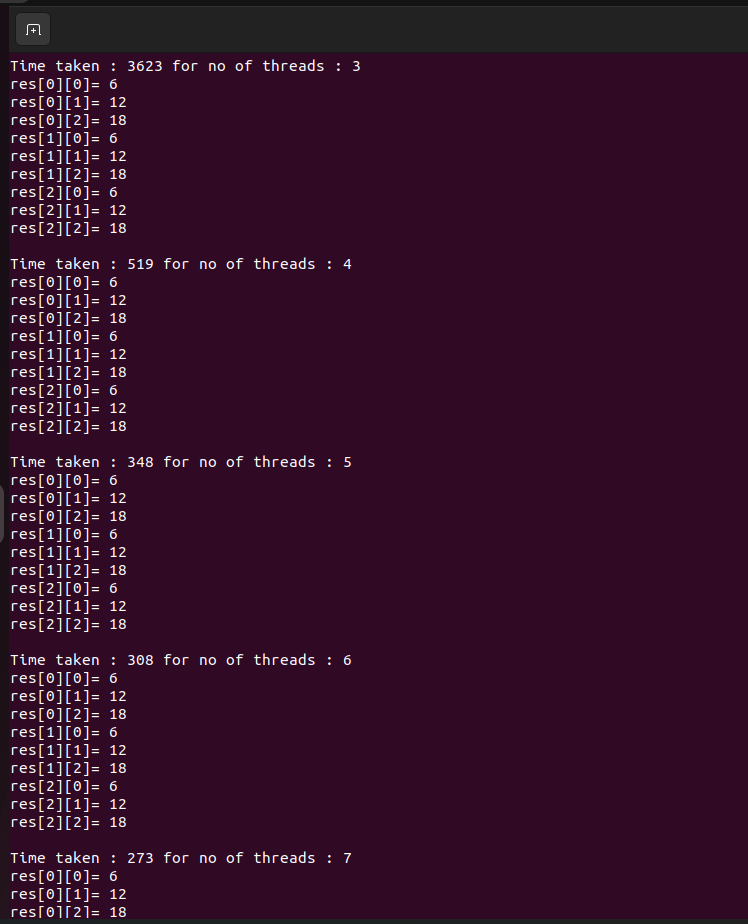
**6.Program to find resultant of matrix multiplication.**

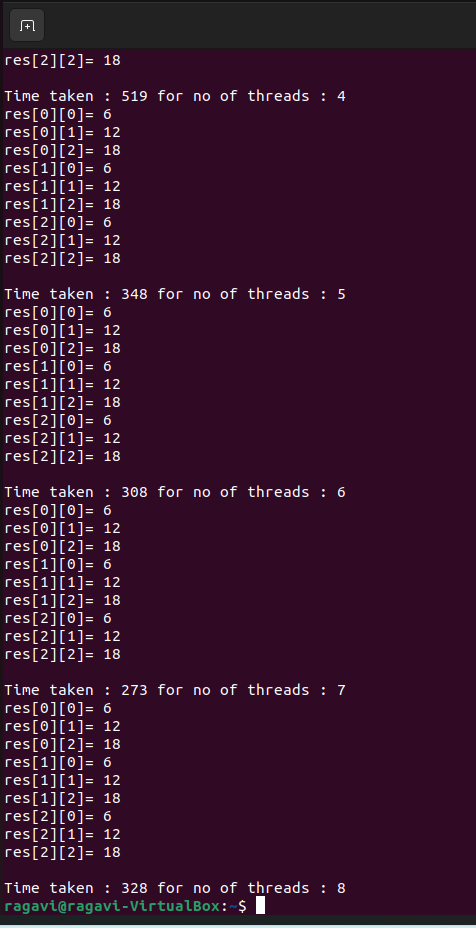
**Code:**



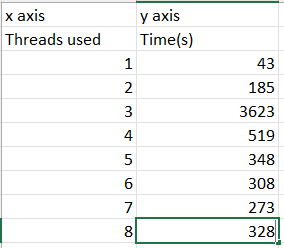
**Output:**

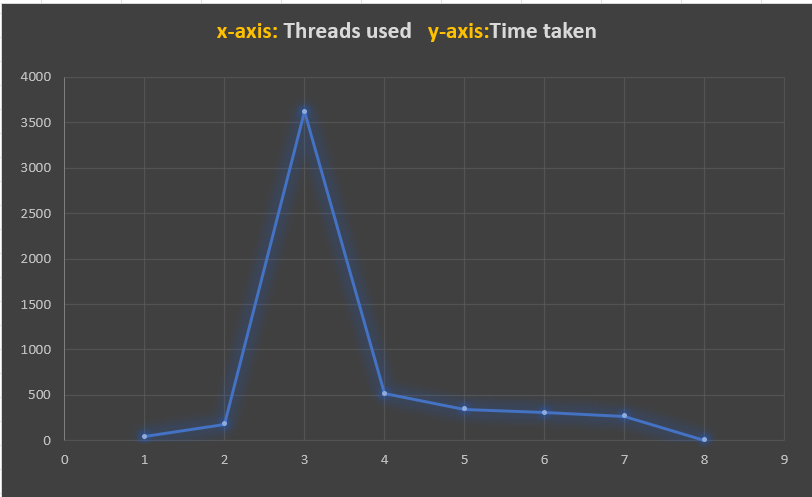






**Graph Plot:**





7.a)No-wait:

#pragma omp parallel

{

#pragma omp for nowait

for (i=1; i<n; i++)

b[i] = (a[i] + a[i-1]) / 2.0;

#pragma omp for nowait

for (i=0; i<m; i++)

y[i] = sqrt(z[i]);

}

b)Barrier:

#pragma omp parallel default(shared)

{

#pragma omp for

for (i=0; i<n; i++)

{

#pragma omp parallel shared(i, n)

{

#pragma omp for

for (j=0; j<n; j++)

work(i, j);

}

}

}