**Open Multi-Processing:**

OpenMP, which stands for Open Multi-Processing, is an API (Application Programming Interface) used for parallel programming in shared-memory systems. It's used to the creation of parallel code and is particularly useful for multi-threaded programming on CPUs.

**Topic:**

This program uses OpenMP, a tool for making programs run faster by using multiple processors at the same time (parallel processing)

**Justification for Selecting this program:**

**Libraries Included**: The program includes various libraries (**iostream, vector, cstdlib, ctime, cmath, omp.h**) to perform specific tasks like input/output, handling arrays, generating random numbers, mathematical operations, and utilizing OpenMP.

**is\_prime Function**: This function checks if a given number is a prime number. It goes through a set of rules to determine if the number is divisible by anything other than 1 and itself.

**Main Function**: This is where the program starts running.

**Variables:**

**num\_random\_numbers**: Stores the count of random numbers to generate.

**random\_numbers**: A vector that will hold the randomly generated numbers.

**prime\_sum**:Stores the sum of prime numbers among the generated random numbers.

**Initializing Random Number Generator:** The program uses std::srand(std::time(0)) to initialize the random number generator based on the current time.

**Generating Random Numbers in Parallel:**

The program **uses OpenMP's #pragma omp parallel** for to generate random numbers in parallel. The loop fills the ‘**random\_numbers’ vector** with random values.

**Calculating Sum of Prime Numbers in Parallel:**

Another parallel loop **(#pragma omp parallel for reduction(+:prime\_sum**)) checks each number in the ‘**random\_numbers’** vector to see if it's a prime number. If it's prime, its value is added to ‘**prime\_sum’**.

**Displaying Result**: After calculating the sum of prime numbers, the program prints the total sum.

**Progam:**

#include <iostream>

#include <vector>

#include <cstdlib>

#include <ctime>

#include <cmath>

#include <omp.h>

bool is\_prime(int n) {

if (n <= 1) return false;

if (n == 2) return true;

if (n % 2 == 0) return false;

for (int i = 3; i <= std::sqrt(n); i += 2) {

if (n % i == 0) return false;

}

return true;

}

int main() {

const int num\_random\_numbers = 100;

std::vector<int> random\_numbers(num\_random\_numbers);

int prime\_sum = 0;

// Initialize random number generator

std::srand(std::time(0));

// Generate random numbers in parallel using OpenMP

#pragma omp parallel for

for (int i = 0; i < num\_random\_numbers; ++i) {

random\_numbers[i] = std::rand() % 1000; // You can change the range as needed

}

// Calculate sum of prime numbers from random numbers in parallel

#pragma omp parallel for reduction(+:prime\_sum)

for (int i = 0; i < num\_random\_numbers; ++i) {

if (is\_prime(random\_numbers[i])) {

prime\_sum += random\_numbers[i];

}

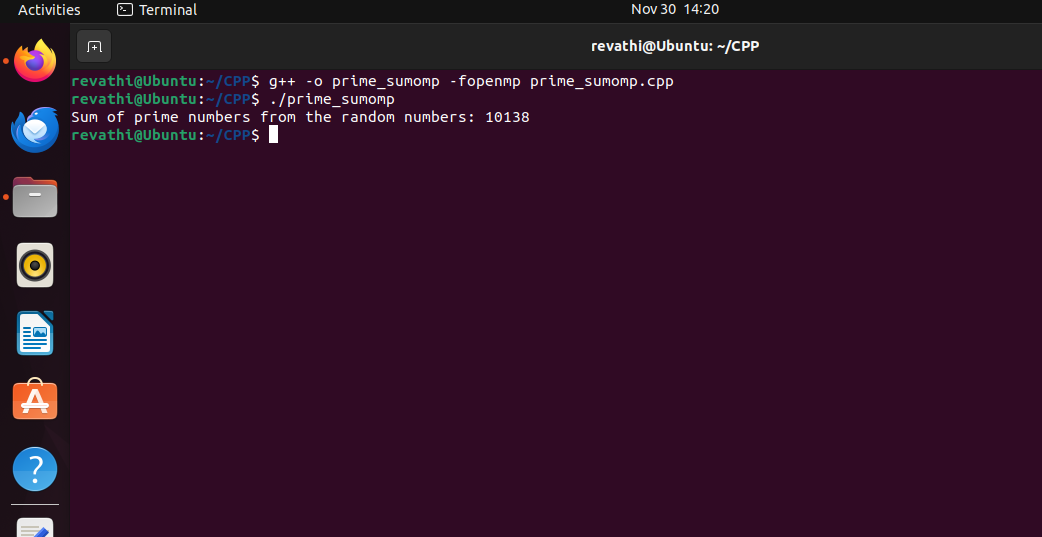
}

std::cout << "Sum of prime numbers from the random numbers: " << prime\_sum << std::endl;

return 0;

}

**Program output:**

****