

**Write a OpenMP program to calculate n Fibonacci numbers using tasks.**

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

int fib(int n) {
    int x, y;

    if (n < 2)
        return n;

    #pragma omp task shared(x)
    x = fib(n - 1);

    #pragma omp task shared(y)
    y = fib(n - 2);

    #pragma omp taskwait
    return x + y;
}

int main() {
    int n;

    printf("Enter number of Fibonacci terms: ");
    scanf("%d", &n);

    printf("Fibonacci Series:\n");

    for (int i = 0; i < n; i++) {
        int result;

        #pragma omp parallel
        {
            #pragma omp single
            {
                result = fib(i);
            }
        }

        printf("%d ", result);
    }

    printf("\n");
    return 0;
}
```

```
braham@braham-VirtualBox: ~/Desktop/ParallelPrograms
braham@braham-VirtualBox:~/Desktop/ParallelPrograms$ gedit prg3.c
braham@braham-VirtualBox:~/Desktop/ParallelPrograms$ gcc -fopenmp prg3.c -o prg3
braham@braham-VirtualBox:~/Desktop/ParallelPrograms$ export OMP_NUM_THREADS=4 && ./prg3
Enter number of Fibonacci terms: 6
Fibonacci Series:
0 1 1 2 3 5
braham@braham-VirtualBox:~/Desktop/ParallelPrograms$
```

**Write a OpenMP program to find the prime numbers from 1 to n employing parallel for directive. Record both serial and parallel execution times.**

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

int is_prime(int num) {
    if (num < 2) return 0;
    for (int i = 2; i <= sqrt(num); i++) {
        if (num % i == 0)
            return 0;
    }
    return 1;
}

int main() {
    int n;
    printf("Enter the value of n: ");
    scanf("%d", &n);

    printf("\nPrime numbers from 1 to %d:\n", n);
    for (int i = 1; i <= n; i++) {
        if (is_prime(i))
            printf("%d ", i);
    }
    printf("\n");

    double start_time, end_time;
```

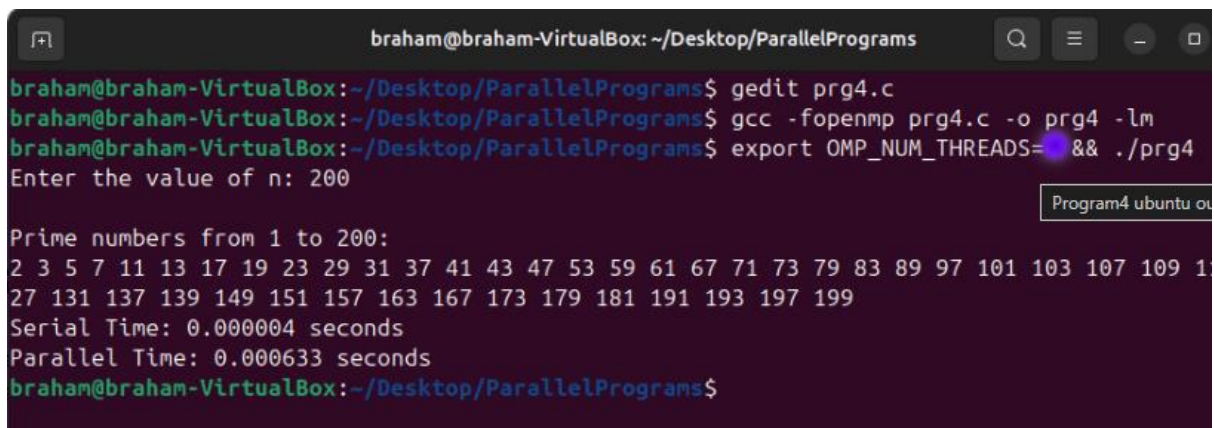
```

start_time = omp_get_wtime();
for (int i = 1; i <= n; i++) {
    is_prime(i);
}
end_time = omp_get_wtime();
printf("Serial Time: %f seconds\n", end_time - start_time);

start_time = omp_get_wtime();
#pragma omp parallel for
for (int i = 1; i <= n; i++) {
    is_prime(i);
}
end_time = omp_get_wtime();
printf("Parallel Time: %f seconds\n", end_time - start_time);

return 0;
}

```



```

braham@braham-VirtualBox: ~/Desktop/ParallelPrograms
braham@braham-VirtualBox:~/Desktop/ParallelPrograms$ gedit prg4.c
braham@braham-VirtualBox:~/Desktop/ParallelPrograms$ gcc -fopenmp prg4.c -o prg4 -lm
braham@braham-VirtualBox:~/Desktop/ParallelPrograms$ export OMP_NUM_THREADS=2 && ./prg4
Enter the value of n: 200

Prime numbers from 1 to 200:
2 3 5 7 11 13 17 19 23 29 31 37 41 43 47 53 59 61 67 71 73 79 83 89 97 101 103 107 109 1
27 131 137 139 149 151 157 163 167 173 179 181 191 193 197 199
Serial Time: 0.000004 seconds
Parallel Time: 0.000633 seconds
braham@braham-VirtualBox:~/Desktop/ParallelPrograms$

```