Practical No 4

PRN: 22520005

Name: Aftab Imtiyaj Bhadgaonkar

Batch: B6

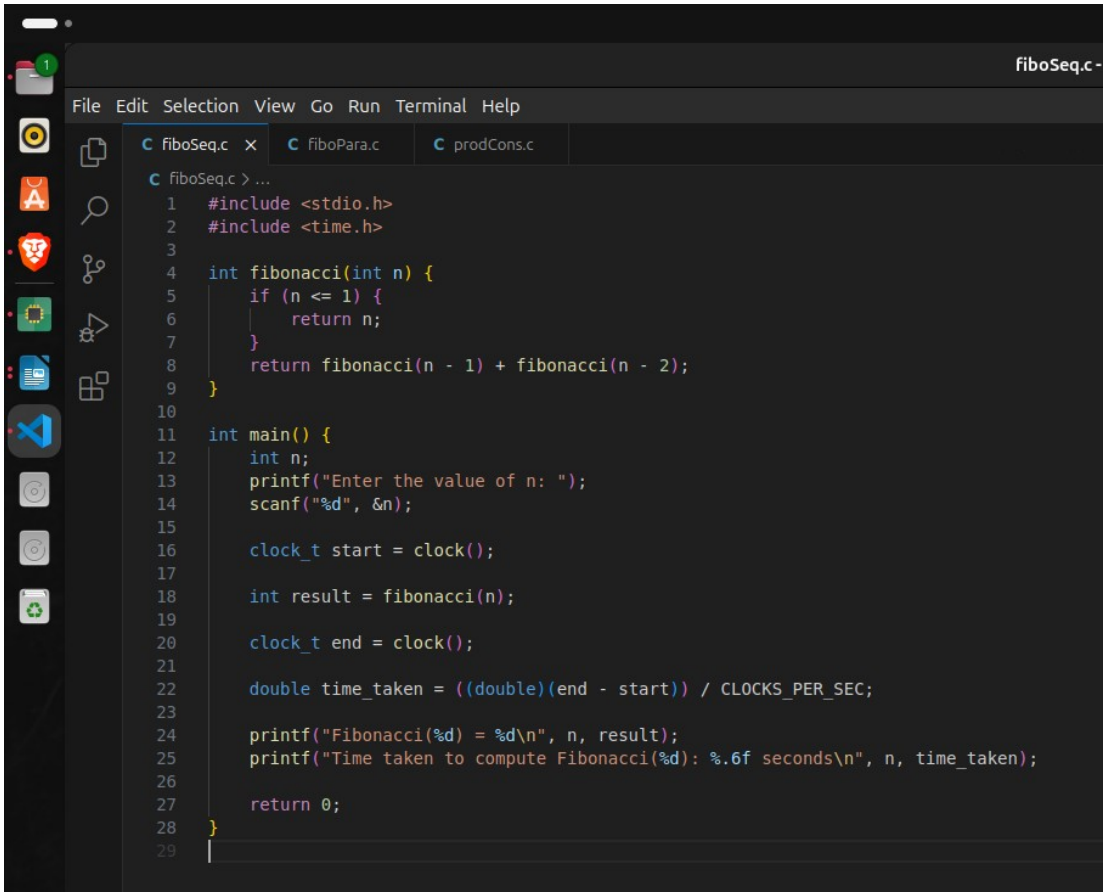
Course: High Performance Computing Lab

Problem Statement 1:

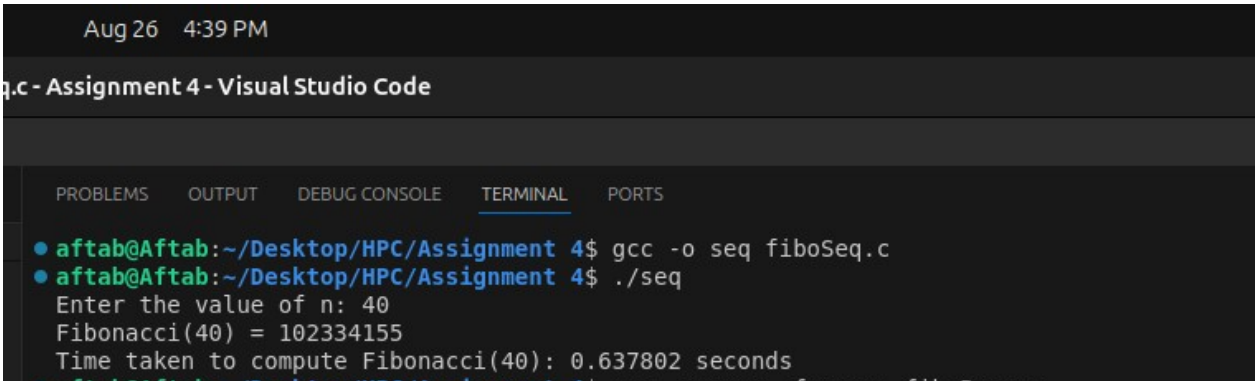
Analyse and implement a Parallel code for below programs using OpenMP considering synchronization requirements. (Demonstrate the use of different clauses and constructs wherever applicable)

Fibonacci Computation:

Code(Sequential):



Screenshot:

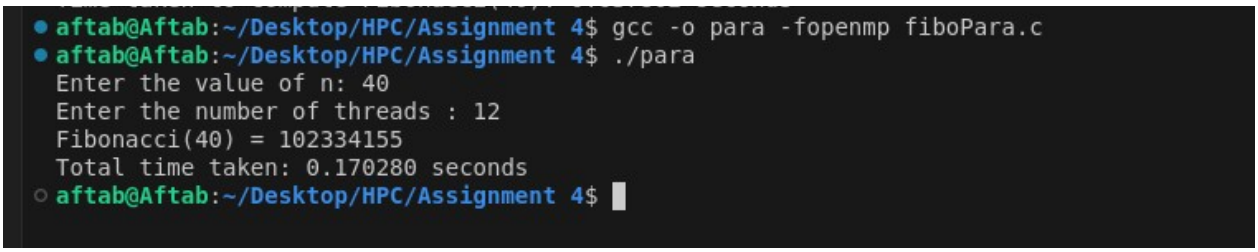


Code(Parallel):

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

unsigned long long fibonacci(int n) {
if (n <= 1) return n;

unsigned long long fib1, fib2;

if (n > 5)
{
#pragma omp task shared(fib1)
fib1 = fibonacci(n - 1);

#pragma omp task shared(fib2)
fib2 = fibonacci(n - 2);

#pragma omp taskwait
}
else
{
fib1 = fibonacci(n - 1);
fib2 = fibonacci(n - 2);
}

return fib1 + fib2;
}
```

```
int main() {
int n, num_threads;
unsigned long long result;

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

printf("Enter the number of threads : ");
scanf("%d", &num_threads);

omp_set_num_threads(num_threads);

double start_time = omp_get_wtime();

#pragma omp parallel
{
#pragma omp single
{
result = fibonacci(n);
}
}

double end_time = omp_get_wtime();

printf("Fibonacci(%d) = %llu\n", n, result);
printf("Total time taken: %f seconds\n", end_time - start_time);

return 0;
}
```

Screenshots:



Analysis:

Input Value	Parallel	Sequential
40	0.170280	0.637802

Problem Statement 2:

Analyse and implement a Parallel code for below programs using OpenMP considering synchronization requirements. (Demonstrate the use of different clauses and constructs wherever applicable)

Producer Consumer Problem

Code:

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

int full = 0;
int empty = 10, x = 0;
omp_lock_t lock;

void producer() {
omp_set_lock(&lock);
if (empty > 0) {
full++;
empty--;
x++;
printf("\nProducer produces item %d\n", x);
} else {
printf("\nBuffer is full!\n");
}
omp_unset_lock(&lock);
}

void consumer() {
omp_set_lock(&lock);
if (full > 0) {
full--;
empty++;
printf("\nConsumer consumes item %d\n", x);
x--;
} else {
```

```
printf("\nBuffer is empty!\n");
}
omp_unset_lock(&lock);
}

int main() {
int n;

omp_init_lock(&lock);

while (1) {
printf(
"\n1. Press 1 for Producer"
"\n2. Press 2 for Consumer"
"\n3. Press 3 for Exit");

printf("\nEnter your choice: ");
scanf("%d", &n);

switch (n) {
case 1:
#pragma omp task
{
producer();
}
break;

case 2:
#pragma omp task
{
consumer();
}
break;

case 3:
omp_destroy_lock(&lock);
exit(0);

default:
```

```
printf("\nInvalid choice! Please try again.");
break;
}
}

return 0;
}
```

Screenshot: