

Class: Final Year (Computer Science and Engineering)

Year: 2022-23

Semester: 07

Course: High Performance Computing Lab

Practical No. 04

Exam Seat No:

1. 2019BTECS00033 – Teknath K Jha

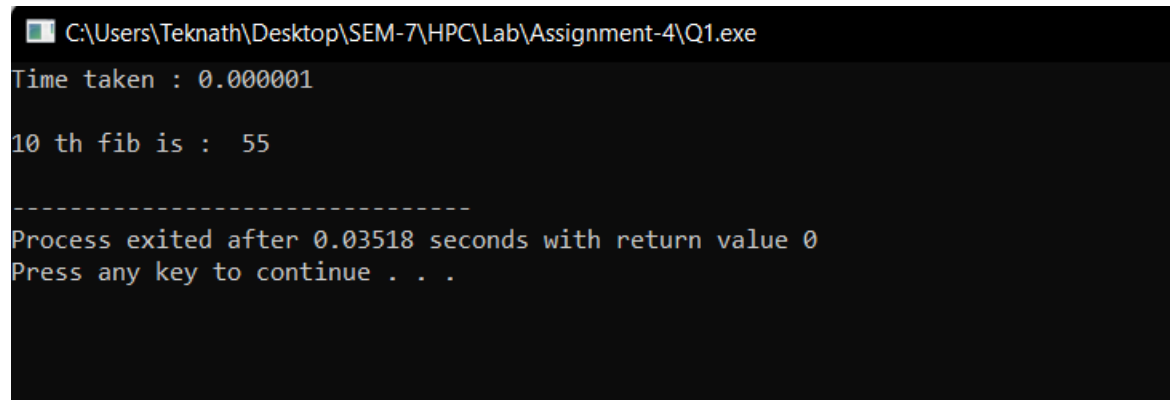
Title of practical:

Use of clauses on given problems

Problem Statement 1:

- 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)

Screenshot 1:



```
C:\Users\Teknath\Desktop\SEM-7\HPC\Lab\Assignment-4\Q1.exe
Time taken : 0.000001
10 th fib is : 55
-----
Process exited after 0.03518 seconds with return value 0
Press any key to continue . . .
```

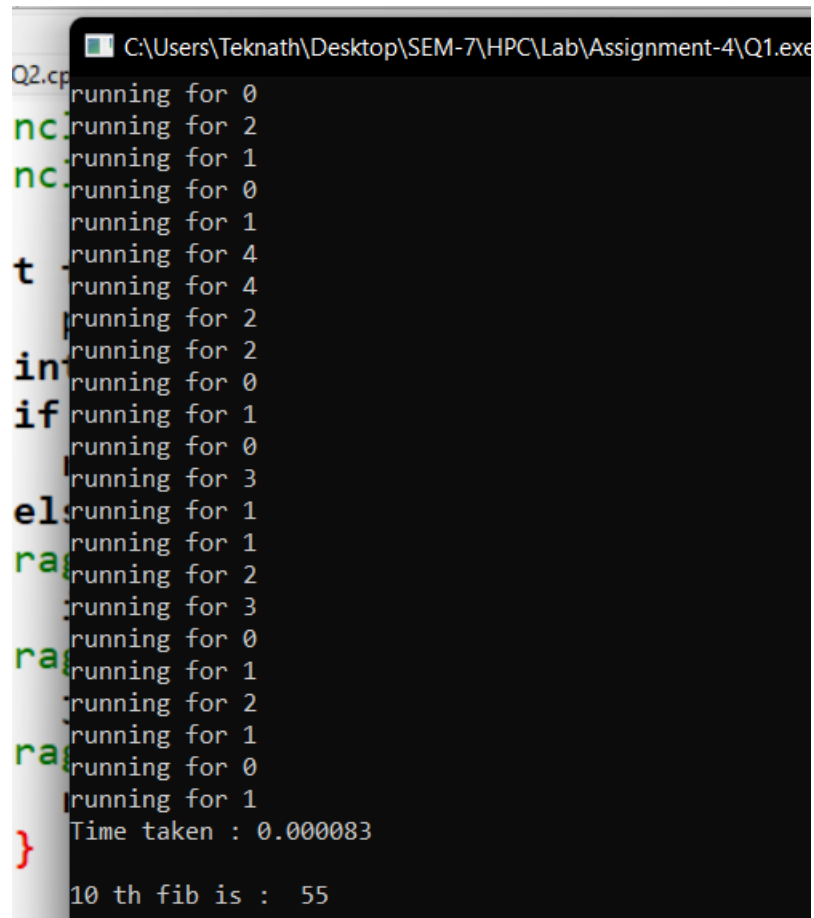
Here I have used **nowait** clause to make synchronization between **ith** and **(i+1)th** which is used in Fibonacci series.

The following code is the recursive parallel code for the Fibonacci series. As there are some programs which can only be solved serially because more number of dependencies get involved to make them parallel which increases the overhead. So DP solution of Fibonacci problem is difficult to do in parallel

In the below code, the parallel directive denotes a parallel region which will be executed by four threads. In the parallel construct, the single directive is used to indicate that only one of the threads will execute the print statement that calls fib(n).

The call to fib(n) generates two tasks, indicated by the task directive. One of the tasks computes fib(n-1) and the other computes fib(n-2), and the return values are added together to produce the value returned by fib(n). Each of the calls to fib(n-1) and fib(n-2) will in turn generate two tasks. Tasks will be recursively generated until the argument passed to fib() is less than 2

Screenshot 2:

A screenshot of a C# console application window titled "C:\Users\Teknath\Desktop\SEM-7\HPC\Lab\Assignment-4\Q1.exe". The console output shows a series of "running for" messages for different values of n, indicating parallel execution. The messages are: "running for 0", "running for 2", "running for 1", "running for 0", "running for 1", "running for 4", "running for 4", "running for 2", "running for 2", "running for 0", "running for 1", "running for 0", "running for 3", "running for 1", "running for 1", "running for 2", "running for 3", "running for 0", "running for 1", "running for 2", "running for 1", "running for 0", "running for 1". After these messages, the output shows "Time taken : 0.000083" and "10 th fib is : 55".

```
Q2.cs
nc
nc
t
in
if
el
ra
ra
ra
}
Time taken : 0.000083
10 th fib is : 55
```

Information 2:

As demonstrated in console I have use 3 threads for calculating Nth Fibonacci number .

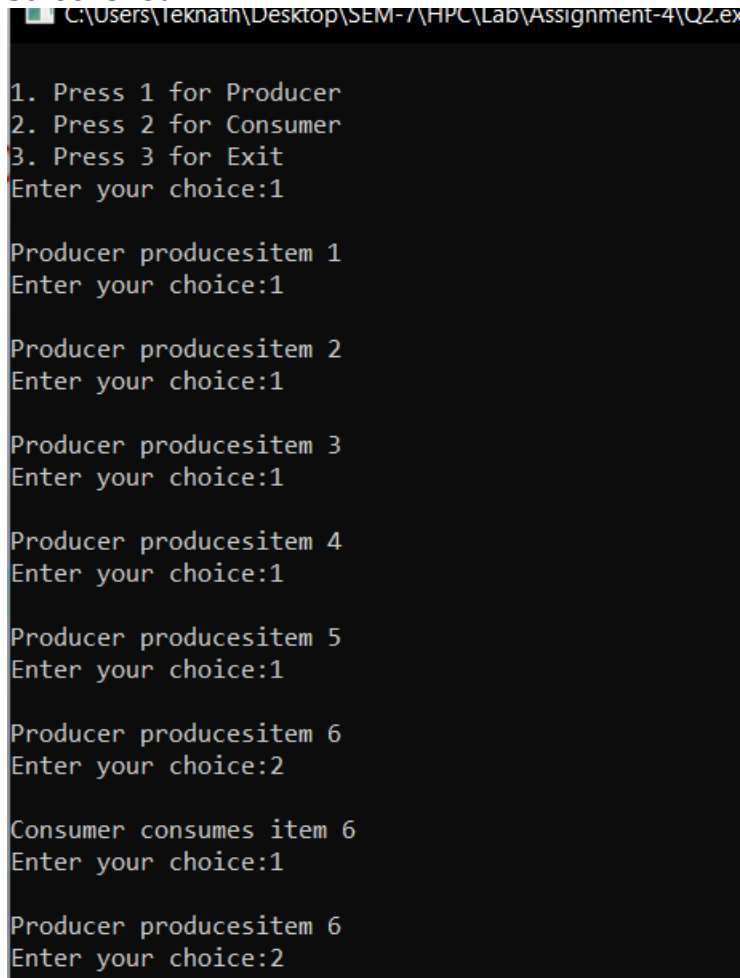
The taskwait directive ensures that the two tasks generated in an invocation of fib() are completed (that is. the tasks compute i and j) before that invocation of fib() returns.

Note that although only one thread executes the single directive and hence the call to fib(n), all four threads will participate in executing the tasks generated.

Problem Statement 2:

Analyze 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:

Screenshot 1:



```
C:\Users\Teknath\Desktop\SEM-7\HPC\Lab\Assignment-4\Q2.exe
1. Press 1 for Producer
2. Press 2 for Consumer
3. Press 3 for Exit
Enter your choice:1

Producer produces item 1
Enter your choice:1

Producer produces item 2
Enter your choice:1

Producer produces item 3
Enter your choice:1

Producer produces item 4
Enter your choice:1

Producer produces item 5
Enter your choice:1

Producer produces item 6
Enter your choice:2

Consumer consumes item 6
Enter your choice:1

Producer produces item 6
Enter your choice:2
```

```
C:\Users\Teknath\Desktop\SEM-7\HPC\Lab\Assignment-4\
1. Press 1 for Producer
2. Press 2 for Consumer
3. Press 3 for Exit
Enter your choice:2
Buffer is empty!
Enter your choice:2
Buffer is empty!
Enter your choice:1
Producer produces item 1
Enter your choice:2
Consumer consumes item 1
Enter your choice:2
Buffer is empty!
Enter your choice:█
```

Information 1 and 2:

In producer consumer problem the producer generates the data and put in the buffer which is consumed by the consumer. This putting and taking of the data by producer and consumer is in parallel.

So, to do this we use the pragma omp file given by cpp predefined making the for loop parallel. But if we do this then there is a synchronisation issue and the data would go wrong. So to avoid this we have to synchronise the code which is common in them (i.e. the critical part)

In the above code we have used “#pragma omp critical” so the critical section works properly. This line ‘critical’ specifies that code is executed by only one thread at a time i.e., only one thread can enter the critical section at a time.

Github Link: <https://github.com/Teknath-jha/HPC-LAB-2019BTECS00033/tree/main/Assignment-4>

