

Practical No. 4

Exam Seat No: 21510111

Title of practical:

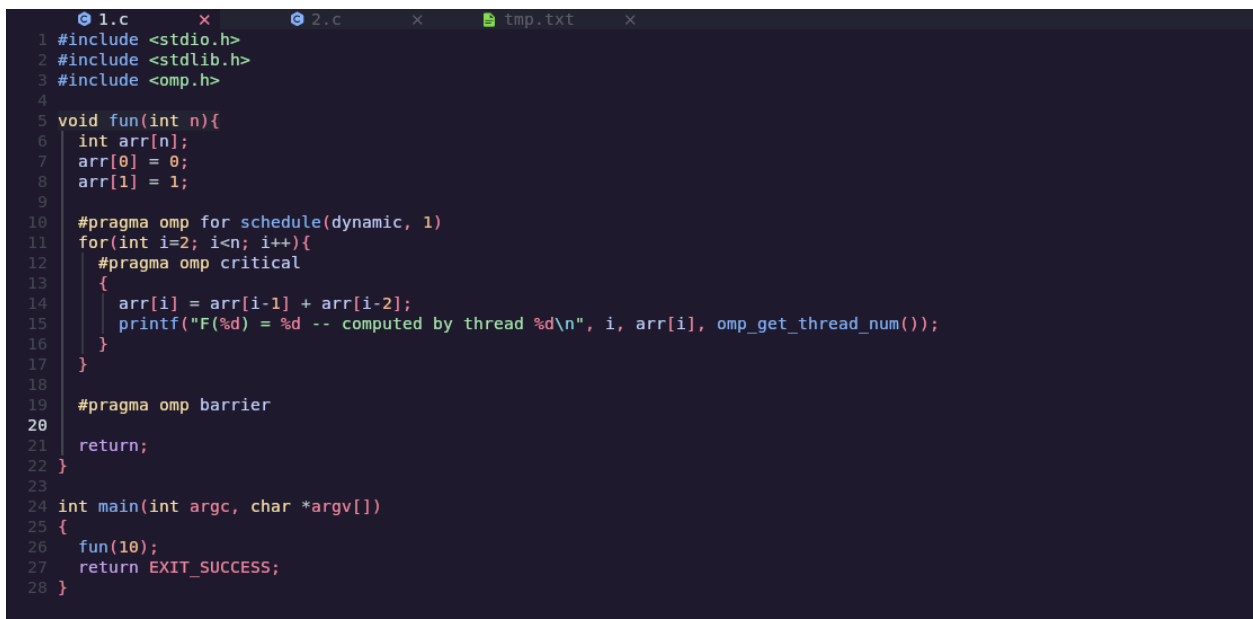
Study and Implementation of Synchronization

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:

Screenshots:



```
1 #include <stdio.h>
2 #include <stdlib.h>
3 #include <omp.h>
4
5 void fun(int n){
6     int arr[n];
7     arr[0] = 0;
8     arr[1] = 1;
9
10    #pragma omp for schedule(dynamic, 1)
11    for(int i=2; i<n; i++){
12        #pragma omp critical
13        {
14            arr[i] = arr[i-1] + arr[i-2];
15            printf("F(%d) = %d -- computed by thread %d\n", i, arr[i], omp_get_thread_num());
16        }
17    }
18
19    #pragma omp barrier
20
21    return;
22 }
23
24 int main(int argc, char *argv[])
25 {
26     fun(10);
27     return EXIT_SUCCESS;
28 }
```

Information and output:

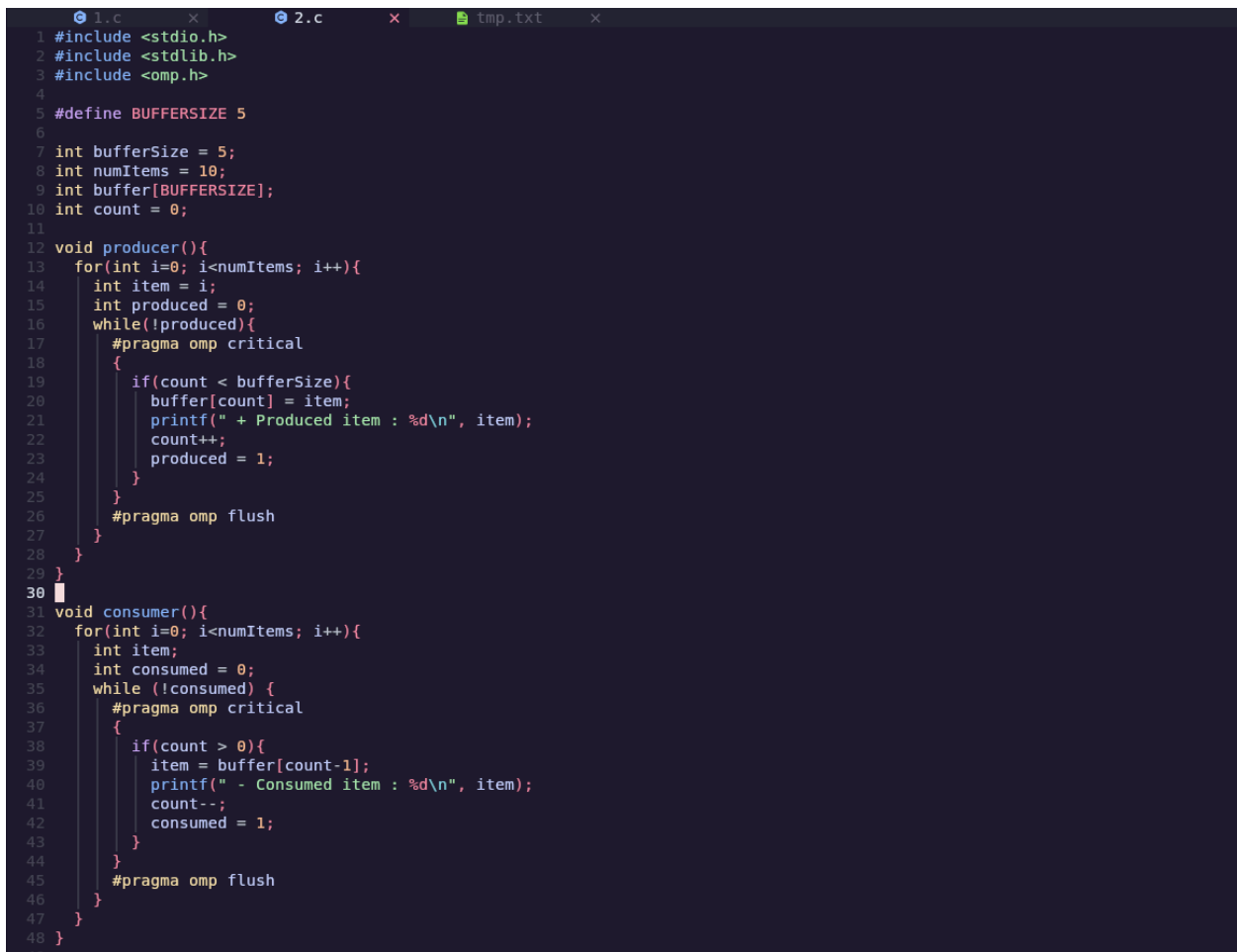
```
*[main][~/acad/hpc_lab/as4]$ gcc -fopenmp 1.c -o 1 && ./1
F(2) = 1 -- computed by thread 0
F(3) = 2 -- computed by thread 0
F(4) = 3 -- computed by thread 0
F(5) = 5 -- computed by thread 0
F(6) = 8 -- computed by thread 0
F(7) = 13 -- computed by thread 0
F(8) = 21 -- computed by thread 0
F(9) = 34 -- computed by thread 0
```

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

Screenshots:



```
1 #include <stdio.h>
2 #include <stdlib.h>
3 #include <omp.h>
4
5 #define BUFFERSIZE 5
6
7 int bufferSize = 5;
8 int numItems = 10;
9 int buffer[BUFFERSIZE];
10 int count = 0;
11
12 void producer(){
13     for(int i=0; i<numItems; i++){
14         int item = i;
15         int produced = 0;
16         while(!produced){
17             #pragma omp critical
18             {
19                 if(count < bufferSize){
20                     buffer[count] = item;
21                     printf(" + Produced item : %d\n", item);
22                     count++;
23                     produced = 1;
24                 }
25             }
26             #pragma omp flush
27         }
28     }
29 }
30
31 void consumer(){
32     for(int i=0; i<numItems; i++){
33         int item;
34         int consumed = 0;
35         while (!consumed) {
36             #pragma omp critical
37             {
38                 if(count > 0){
39                     item = buffer[count-1];
40                     printf(" - Consumed item : %d\n", item);
41                     count--;
42                     consumed = 1;
43                 }
44             }
45             #pragma omp flush
46         }
47     }
48 }
```

```
50 int main(int argc, char *argv[])
51 {
52     omp_set_num_threads(2);
53     #pragma omp parallel sections
54     {
55         #pragma omp section
56         {
57             producer();
58         }
59         #pragma omp section
60         {
61             consumer();
62         }
63     }
64     return EXIT_SUCCESS;
65 }
```

Information and output:

```
*[main][~/acad/hpc_lab/as4]$ gcc -fopenmp 2.c -o 2 && ./2
+ Produced item : 0
+ Produced item : 1
+ Produced item : 2
+ Produced item : 3
+ Produced item : 4
- Consumed item : 4
- Consumed item : 3
- Consumed item : 2
- Consumed item : 1
- Consumed item : 0
+ Produced item : 5
+ Produced item : 6
+ Produced item : 7
+ Produced item : 8
+ Produced item : 9
- Consumed item : 9
- Consumed item : 8
- Consumed item : 7
- Consumed item : 6
- Consumed item : 5
```

Github Link: <https://github.com/Sidd-77/hpc-lab/tree/main/as4>