NATIONAL INSTITUTE OF TECHNOLOGY KARNATAKA SURATHKAL
DEPARTMENT OF INFORMATION TECHNOLOGY

IT 301 Parallel Computing LAB 4

24th August 2021
Faculty: Dr. Geetha V

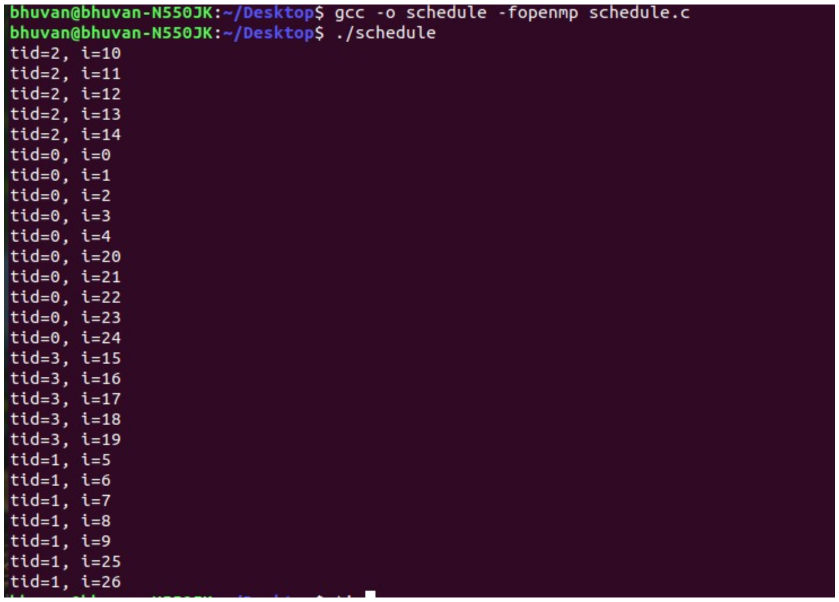
NAME: BHUVANESWAR DHARMASIVAM

ROLL NO: 191IT107

1. Understanding concept of schedule. Write the observation using schedule (static, 5), schedule (dynamic, 5) and schedule (guided, 5) [Marks: 1+1+1=3]

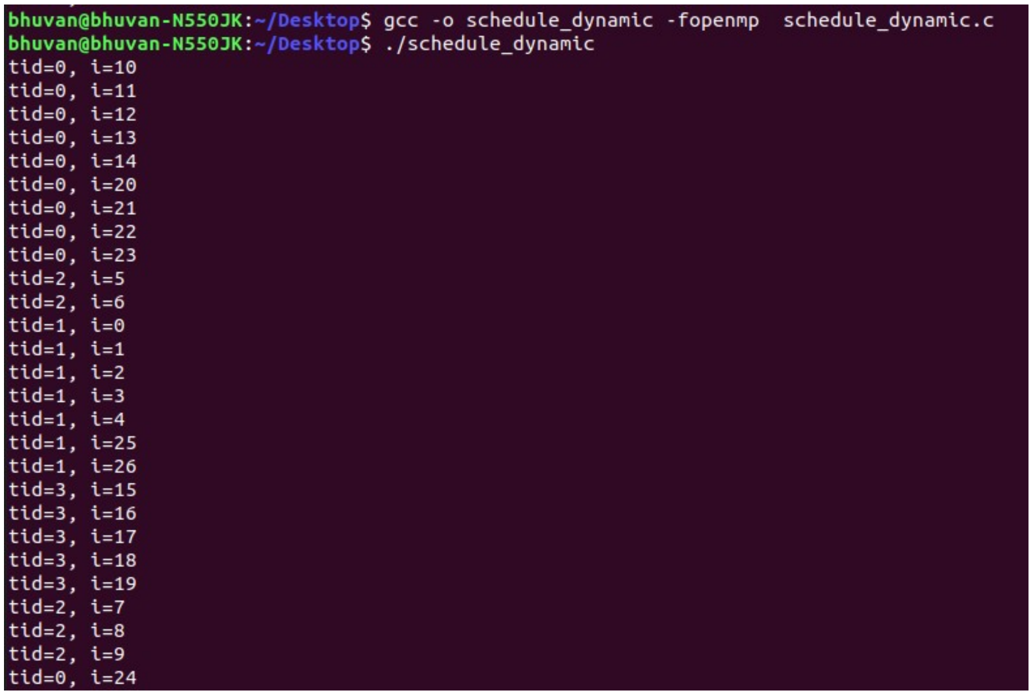
```
#include <stdio.h>
#include <stdlib.h>
#include <omp.h>
int main (void) {
int i;
#pragma omp parallel num_threads(4)
{
  #pragma omp for schedule(guided,5) private(i)
    for(i=0;i<27;i++)
     {
        printf("tid=%d, i=%d \n",omp_get_thread_num(),i);
     }
    }
  return 0;
}
```

static



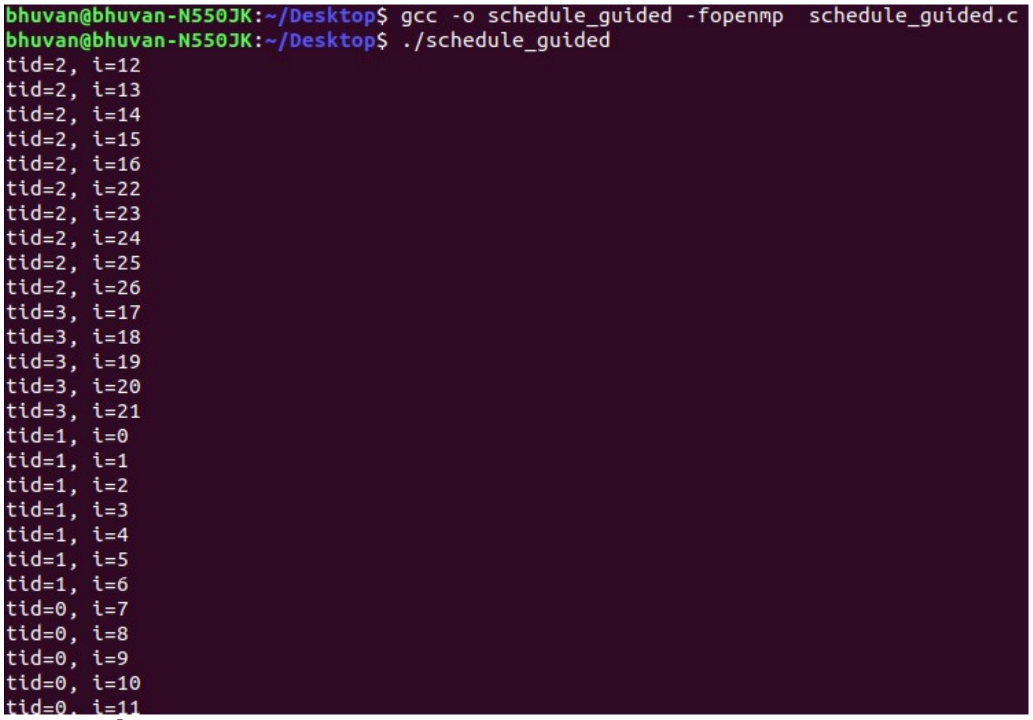
```
bhuvan@bhuvan-N550JK:~/Desktop$ gcc -o schedule -fopenmp schedule.c
bhuvan@bhuvan-N550JK:~/Desktop$ ./schedule
tid=2, i=10
tid=2, i=11
tid=2, i=12
tid=2, i=13
tid=2, i=14
tid=0, i=0
tid=0, i=1
tid=0, i=2
tid=0, i=3
tid=0, i=4
tid=0, i=20
tid=0, i=21
tid=0, i=22
tid=0, i=23
tid=0, i=24
tid=3, i=15
tid=3, i=16
tid=3, i=17
tid=3, i=18
tid=3, i=19
tid=1, i=5
tid=1, i=6
tid=1, i=7
tid=1, i=8
tid=1, i=9
tid=1, i=25
tid=1, i=26
```

dynamic



```
bhuvan@bhuvan-N550JK:~/Desktop$ gcc -o schedule_dynamic -fopenmp  schedule_dynamic.c
bhuvan@bhuvan-N550JK:~/Desktop$ ./schedule_dynamic
tid=0, i=10
tid=0, i=11
tid=0, i=12
tid=0, i=13
tid=0, i=14
tid=0, i=20
tid=0, i=21
tid=0, i=22
tid=0, i=23
tid=2, i=5
tid=2, i=6
tid=1, i=0
tid=1, i=1
tid=1, i=2
tid=1, i=3
tid=1, i=4
tid=1, i=25
tid=1, i=26
tid=3, i=15
tid=3, i=16
tid=3, i=17
tid=3, i=18
tid=3, i=19
tid=2, i=7
tid=2, i=8
tid=2, i=9
tid=0, i=24
```

guided



```
bhuvan@bhuvan-N550JK:~/Desktop$ gcc -o schedule_guided -fopenmp  schedule_guided.c
bhuvan@bhuvan-N550JK:~/Desktop$ ./schedule_guided
tid=2, i=12
tid=2, i=13
tid=2, i=14
tid=2, i=15
tid=2, i=16
tid=2, i=22
tid=2, i=23
tid=2, i=24
tid=2, i=25
tid=2, i=26
tid=3, i=17
tid=3, i=18
tid=3, i=19
tid=3, i=20
tid=3, i=21
tid=1, i=0
tid=1, i=1
tid=1, i=2
tid=1, i=3
tid=1, i=4
tid=1, i=5
tid=1, i=6
tid=0, i=7
tid=0, i=8
tid=0, i=9
tid=0, i=10
tid=0, i=11
```

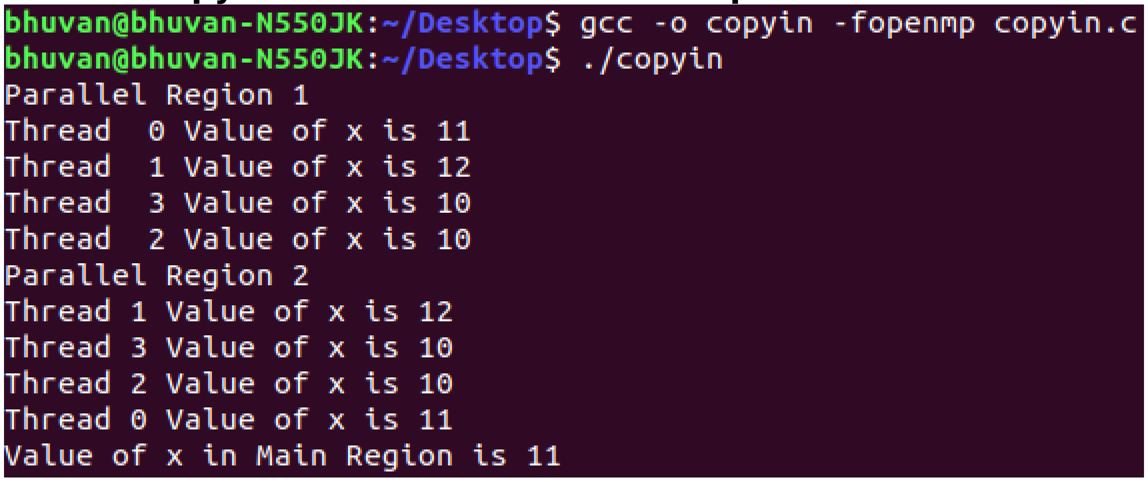
Observation:
In **static,** iterations are divides equally to different threads in chunk sizes and executed. In **Dynamic**, iterations are assigned to each thread in chunk sizes and last thread might have different. In **Guided**, thread executes the chunk of iteration and then requests another chunk, until all iterations are complete

2. Execute following code and observe the working of threadprivate directive and copyin clause:

```
#include<stdio.h>
#include<omp.h>
int tid,x;
#pragma omp threadprivate(x,tid)
void main()
{
x=10;
#pragma omp parallel num_threads(4) copyin(x)
{
tid=omp_get_thread_num();
#pragma omp master
{
printf("Parallel Region 1 \n");
x=x+1;
}
#pragma omp barrier
if(tid==1)
x=x+2;
printf("Thread % d Value of x is %d\n",tid,x);
}//#pragma omp barrier
#pragma omp parallel num_threads(4)
{
#pragma omp master
{
printf("Parallel Region 2 \n");
}
#pragma omp barrier
printf("Thread %d Value of x is %d\n",tid,x);
}
printf("Value of x in Main Region is %d\n",x);
}
```

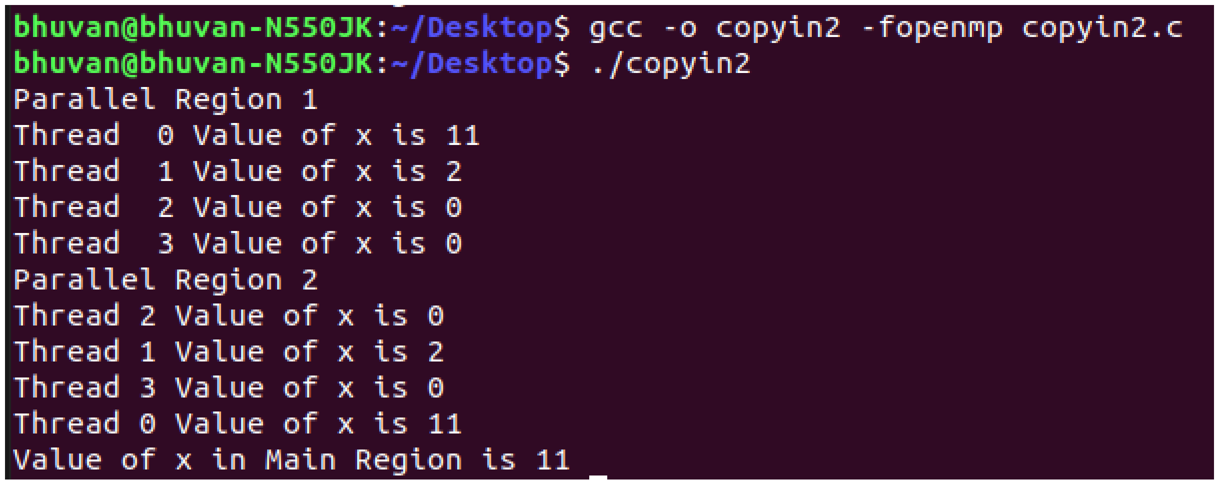
Do the following: [Marks: 1+1=2]

a. Remove copyin clause and check the output.



```
bhuvan@bhuvan-N550JK:~/Desktop$ gcc -o copyin -fopenmp copyin.c
bhuvan@bhuvan-N550JK:~/Desktop$ ./copyin
Parallel Region 1
Thread  0 Value of x is 11
Thread  1 Value of x is 12
Thread  3 Value of x is 10
Thread  2 Value of x is 10
Parallel Region 2
Thread 1 Value of x is 12
Thread 3 Value of x is 10
Thread 2 Value of x is 10
Thread 0 Value of x is 11
Value of x in Main Region is 11
```

b. Remove copyin clause and initialize x globally.
Note the observation about threadprivate directive and copyin clause.



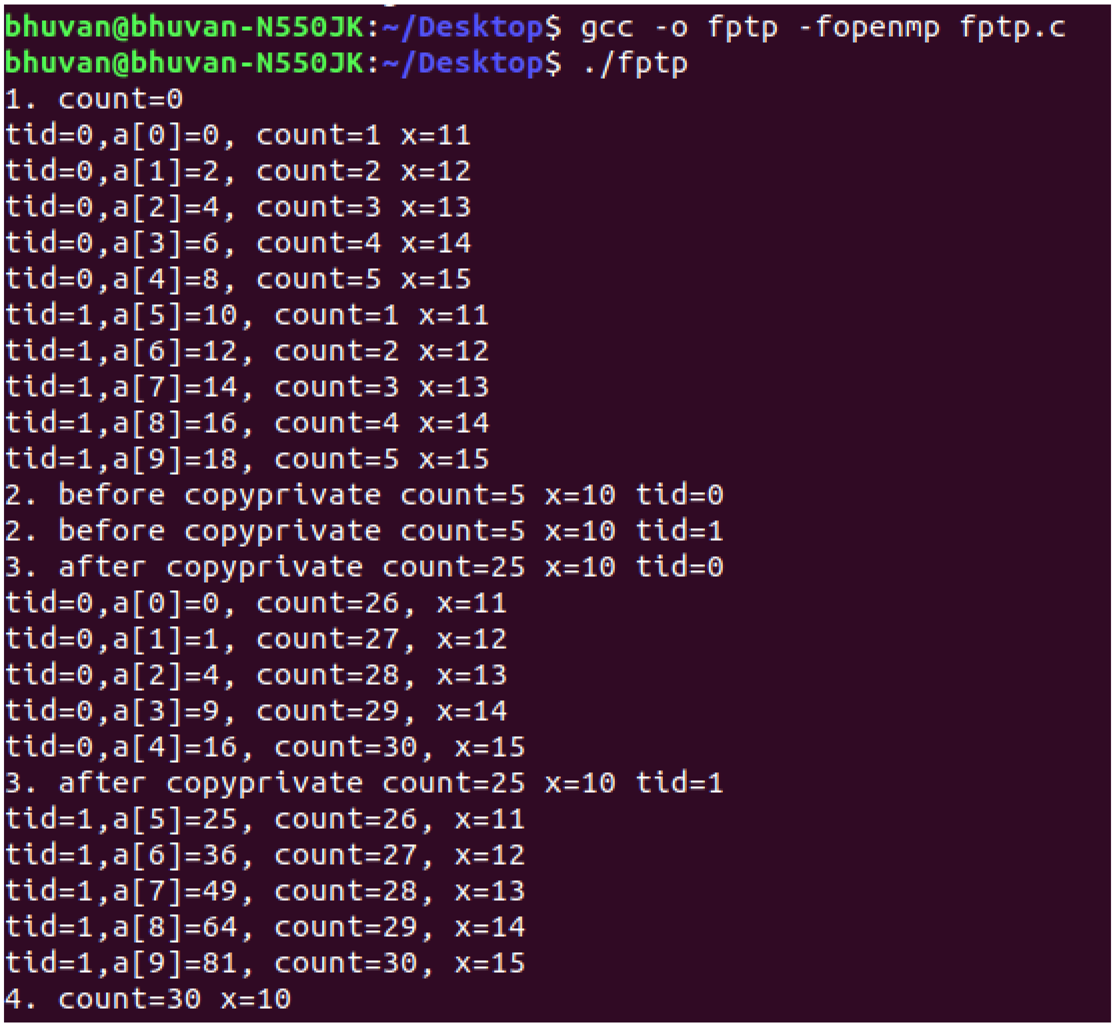
```
bhuvan@bhuvan-N550JK:~/Desktop$ gcc -o copyin2 -fopenmp copyin2.c
bhuvan@bhuvan-N550JK:~/Desktop$ ./copyin2
Parallel Region 1
Thread  0 Value of x is 11
Thread  1 Value of x is 2
Thread  2 Value of x is 0
Thread  3 Value of x is 0
Parallel Region 2
Thread 2 Value of x is 0
Thread 1 Value of x is 2
Thread 3 Value of x is 0
Thread 0 Value of x is 11
Value of x in Main Region is 11
```

Observation: Threadprivate allows each thread is allowed to have its own temporary view of the shared memory. And copyin allows threads to access the master thread's value, for a threadprivate variable.

--

3. Learn the concept of firstprivate() and threadprivate()

```
#include <stdio.h>
#include <stdlib.h>
#include <omp.h>
int count=0;
#pragma omp threadprivate(count)

int main (void) {
int x=10, y=20,a[10],b[10],c[10],i;
//int count=0;
for(i=0;i<10;i++)
 b[i]=c[i]=i;

printf("1. count=%d\n",count);
#pragma omp parallel num_threads(2) copyin(count)
{
 #pragma omp for schedule(static,5) firstprivate(x)
  for(i=0;i<10;i++)
   {
    int tid1=omp_get_thread_num();
    a[i]=b[i]+c[i];
    count++;
    x++;
    printf("tid=%d,a[%d]=%d, count=%d x=%d\n",tid1,i,a[i],count,x);
    }

  #pragma omp barrier
  printf("2. before copyprivate count=%d x=%d tid=%d\
n",count,x,omp_get_thread_num());
 #pragma omp single copyprivate(count)
  {
   count=count+20;
}
printf("3. after copyprivate count=%d x=%d tid=%d\
n",count,x,omp_get_thread_num());

#pragma omp for schedule(static,5) firstprivate(x)
  for(i=0;i<10;i++)
   {
    int tid1=omp_get_thread_num();
    a[i]=b[i]*c[i];
    count++;
    x++;
    printf("tid=%d,a[%d]=%d, count=%d, x=%d\n",tid1,i,a[i],count,x);
    }
   }
#pragma omp barrier
 printf("4. count=%d x=%d\n",count,x);
printf("\n");
return 0;
}
```
Analyse the results for variable count and x. write your observation [Marks: 1+1=2]



```
bhuvan@bhuvan-N550JK:~/Desktop$ gcc -o fptp -fopenmp fptp.c
bhuvan@bhuvan-N550JK:~/Desktop$ ./fptp
1. count=0
tid=0,a[0]=0, count=1 x=11
tid=0,a[1]=2, count=2 x=12
tid=0,a[2]=4, count=3 x=13
tid=0,a[3]=6, count=4 x=14
tid=0,a[4]=8, count=5 x=15
tid=1,a[5]=10, count=1 x=11
tid=1,a[6]=12, count=2 x=12
tid=1,a[7]=14, count=3 x=13
tid=1,a[8]=16, count=4 x=14
tid=1,a[9]=18, count=5 x=15
2. before copyprivate count=5 x=10 tid=0
2. before copyprivate count=5 x=10 tid=1
3. after copyprivate count=25 x=10 tid=0
tid=0,a[0]=0, count=26, x=11
tid=0,a[1]=1, count=27, x=12
tid=0,a[2]=4, count=28, x=13
tid=0,a[3]=9, count=29, x=14
tid=0,a[4]=16, count=30, x=15
3. after copyprivate count=25 x=10 tid=1
tid=1,a[5]=25, count=26, x=11
tid=1,a[6]=36, count=27, x=12
tid=1,a[7]=49, count=28, x=13
tid=1,a[8]=64, count=29, x=14
tid=1,a[9]=81, count=30, x=15
4. count=30 x=10
```

Observation: Threadprivate allows each thread is allowed to have its own temporary view of the shared memory. And copyprivaet broadcasts a value from the data environment of one implicit task to the data environments of the other implicit tasks belonging to the parallel region.

--

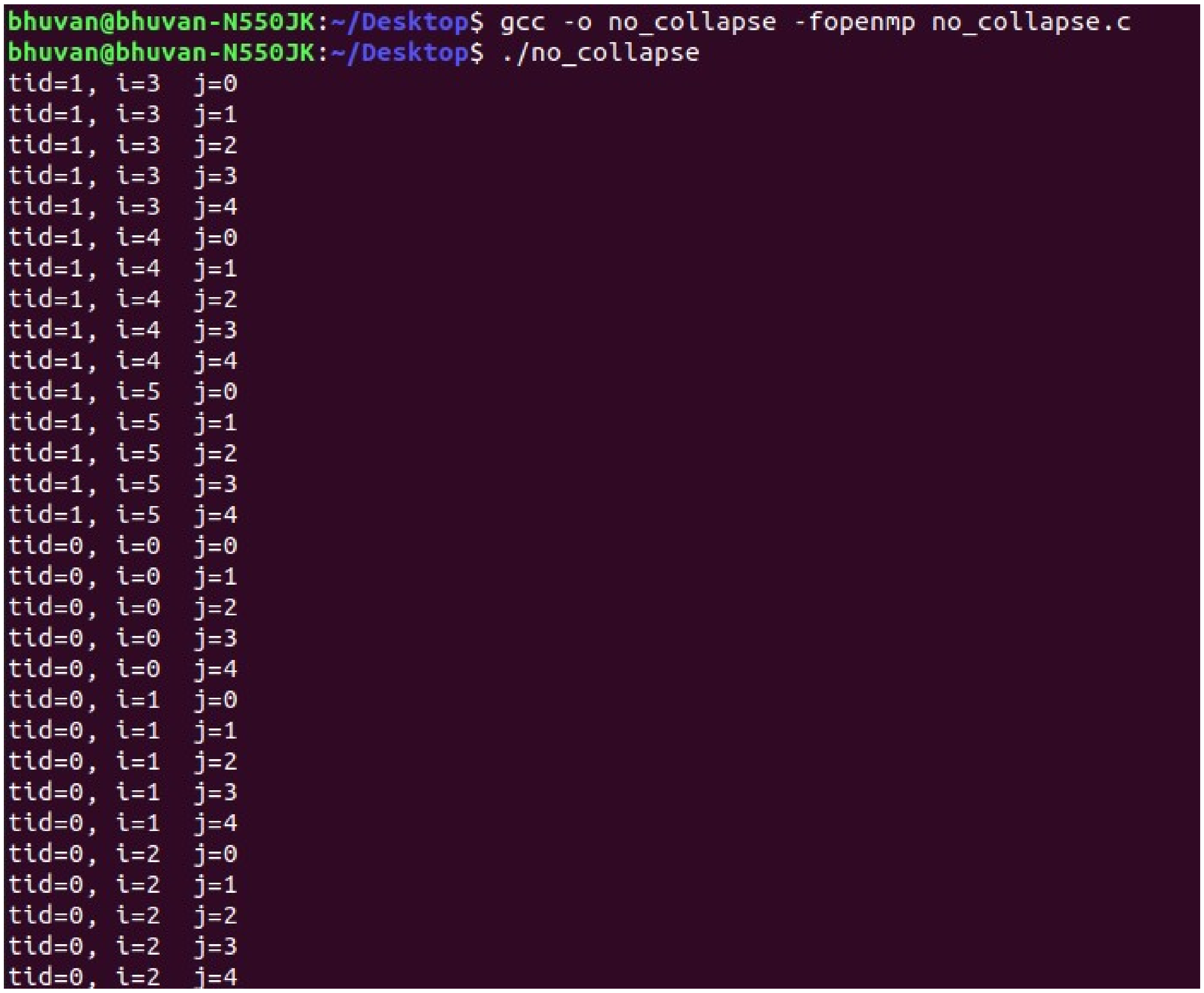
4. Program to understand the concept of collapse()

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


int main (void) {
int i,j;
#pragma omp parallel
{
   #pragma omp for schedule(static,3) private(i,j) collapse(2)
     for(i=0;i<6;i++)
      for(j=0;j<5;j++)
     {
      int tid2=omp_get_thread_num();
       printf("tid=%d, i=%d  j=%d\n",omp_get_thread_num(),i,j);
     }
    }

return 0;
}
```

Consider <u>three for loops</u> and check the result with no collapse(), collapse(2) and collapse(3). [1+1+1=3 Marks]

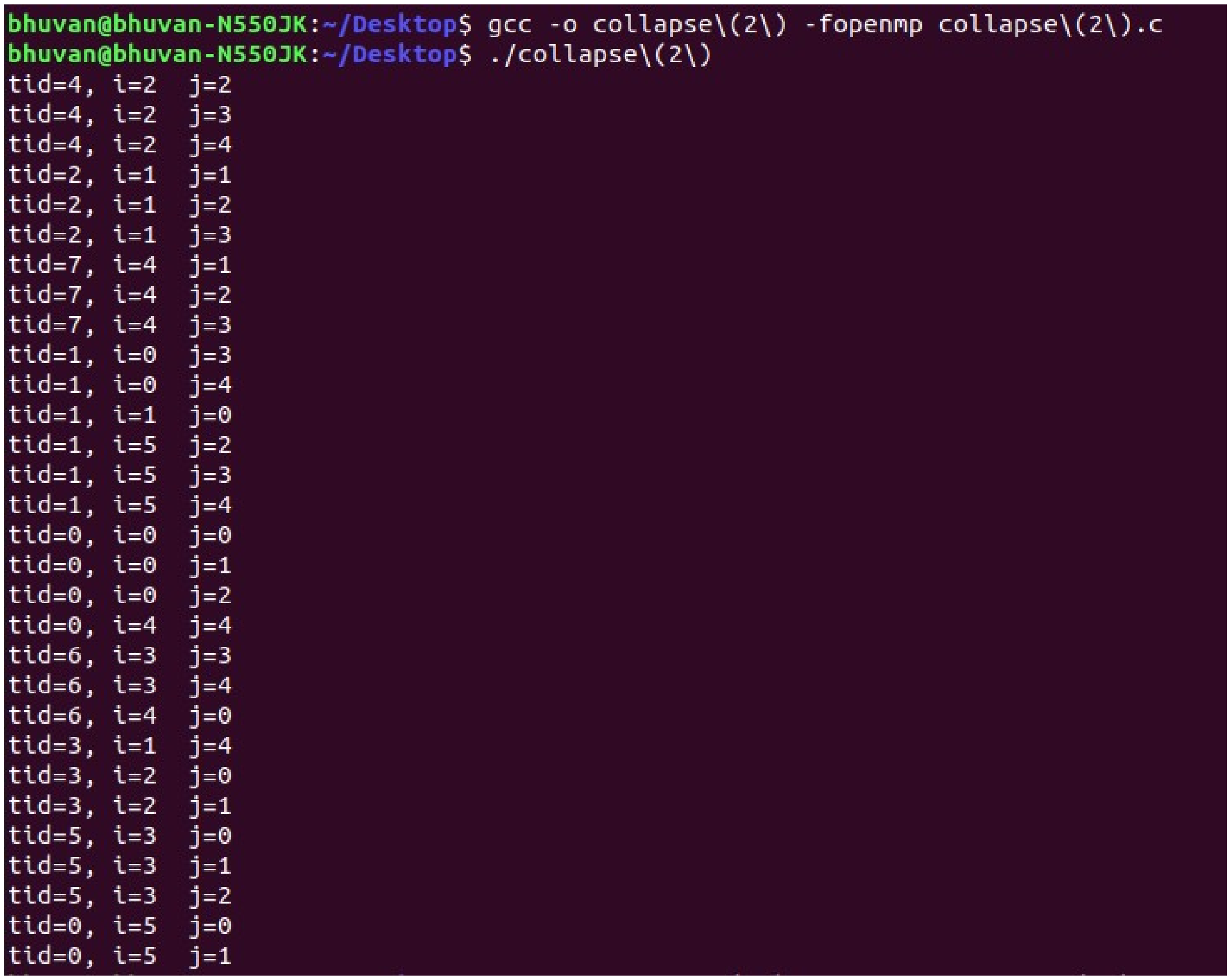
Observation: It increases the total number of iterations that will be partitioned across the available number of OMP threads by reducing the granularity of work to be done by each thread.

no collapse()



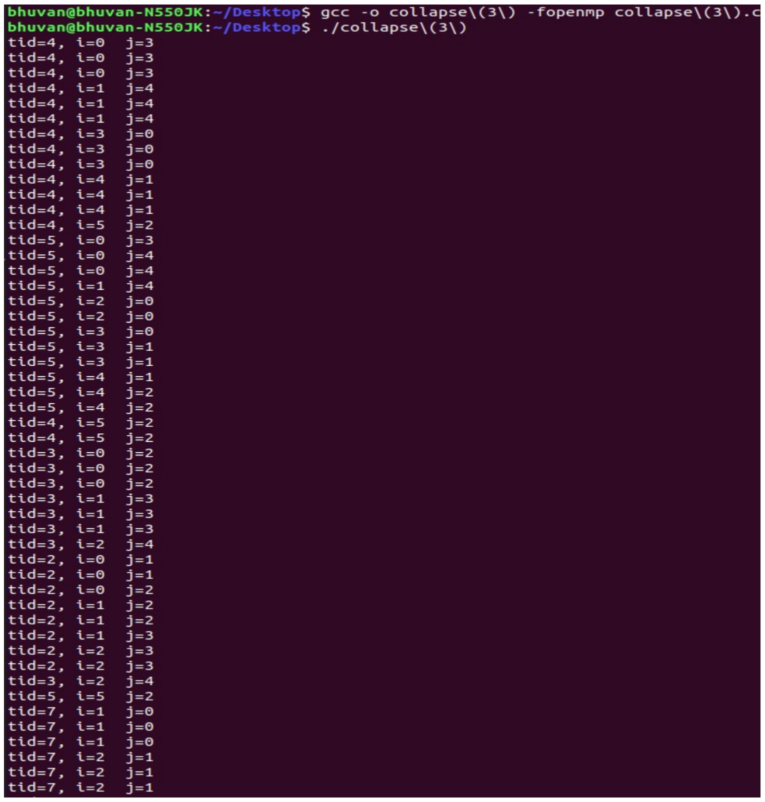
```
bhuvan@bhuvan-N550JK:~/Desktop$ gcc -o no_collapse -fopenmp no_collapse.c
bhuvan@bhuvan-N550JK:~/Desktop$ ./no_collapse
tid=1, i=3  j=0
tid=1, i=3  j=1
tid=1, i=3  j=2
tid=1, i=3  j=3
tid=1, i=3  j=4
tid=1, i=4  j=0
tid=1, i=4  j=1
tid=1, i=4  j=2
tid=1, i=4  j=3
tid=1, i=4  j=4
tid=1, i=5  j=0
tid=1, i=5  j=1
tid=1, i=5  j=2
tid=1, i=5  j=3
tid=1, i=5  j=4
tid=0, i=0  j=0
tid=0, i=0  j=1
tid=0, i=0  j=2
tid=0, i=0  j=3
tid=0, i=0  j=4
tid=0, i=1  j=0
tid=0, i=1  j=1
tid=0, i=1  j=2
tid=0, i=1  j=3
tid=0, i=1  j=4
tid=0, i=2  j=0
tid=0, i=2  j=1
tid=0, i=2  j=2
tid=0, i=2  j=3
tid=0, i=2  j=4
```

collapse(2)

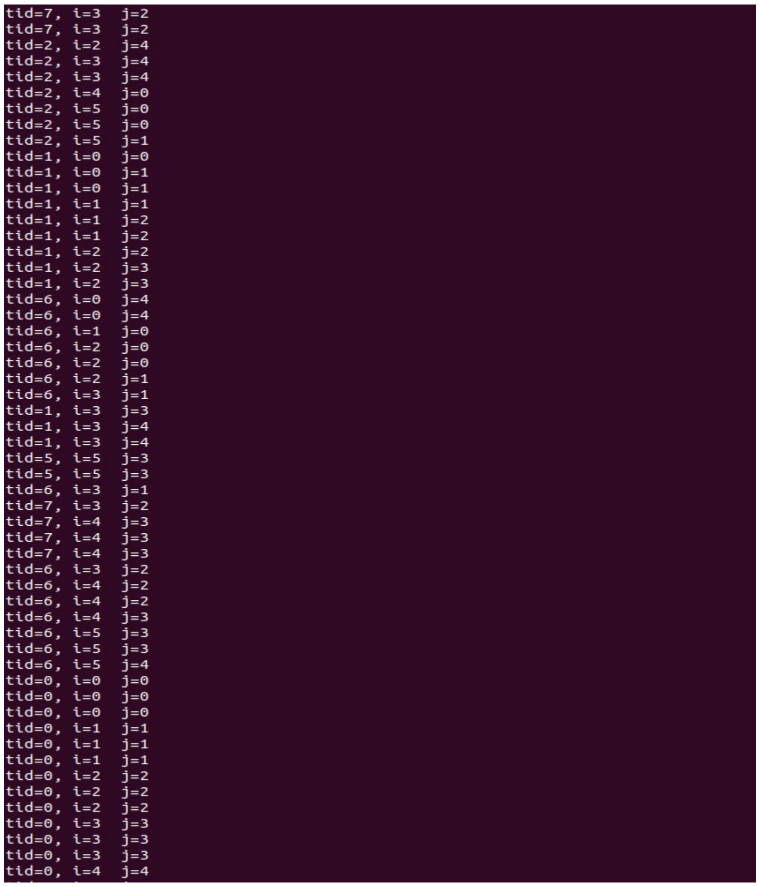


```
bhuvan@bhuvan-N550JK:~/Desktop$ gcc -o collapse\(2\) -fopenmp collapse\(2\).c
bhuvan@bhuvan-N550JK:~/Desktop$ ./collapse\(2\)
tid=4, i=2  j=2
tid=4, i=2  j=3
tid=4, i=2  j=4
tid=2, i=1  j=1
tid=2, i=1  j=2
tid=2, i=1  j=3
tid=7, i=4  j=1
tid=7, i=4  j=2
tid=7, i=4  j=3
tid=1, i=0  j=3
tid=1, i=0  j=4
tid=1, i=1  j=0
tid=1, i=5  j=2
tid=1, i=5  j=3
tid=1, i=5  j=4
tid=0, i=0  j=0
tid=0, i=0  j=1
tid=0, i=0  j=2
tid=0, i=4  j=4
tid=6, i=3  j=3
tid=6, i=3  j=4
tid=6, i=4  j=0
tid=3, i=1  j=4
tid=3, i=2  j=0
tid=3, i=2  j=1
tid=5, i=3  j=0
tid=5, i=3  j=1
tid=5, i=3  j=2
tid=0, i=5  j=0
tid=0, i=5  j=1
```

collapse(3)



```
bhuvan@bhuvan-N550JK:~/Desktop$ gcc -o collapse\(3\) -fopenmp collapse\(3\).c
bhuvan@bhuvan-N550JK:~/Desktop$ ./collapse\(3\)
tid=4, i=0  j=3
tid=4, i=0  j=3
tid=4, i=0  j=3
tid=4, i=1  j=4
tid=4, i=1  j=4
tid=4, i=1  j=4
tid=4, i=3  j=0
tid=4, i=3  j=0
tid=4, i=3  j=0
tid=4, i=4  j=1
tid=4, i=4  j=1
tid=4, i=4  j=1
tid=4, i=5  j=2
tid=5, i=0  j=3
tid=5, i=0  j=4
tid=5, i=0  j=4
tid=5, i=1  j=4
tid=5, i=2  j=0
tid=5, i=2  j=0
tid=5, i=3  j=0
tid=5, i=3  j=1
tid=5, i=3  j=1
tid=5, i=4  j=1
tid=5, i=4  j=2
tid=5, i=4  j=2
tid=4, i=5  j=2
tid=4, i=5  j=2
tid=3, i=0  j=2
tid=3, i=0  j=2
tid=3, i=0  j=2
tid=3, i=1  j=3
tid=3, i=1  j=3
tid=3, i=1  j=3
tid=3, i=2  j=4
tid=2, i=0  j=1
tid=2, i=0  j=1
tid=2, i=0  j=2
tid=2, i=1  j=2
tid=2, i=1  j=2
tid=2, i=1  j=3
tid=2, i=2  j=3
tid=2, i=2  j=3
tid=3, i=2  j=4
tid=5, i=5  j=2
tid=7, i=1  j=0
tid=7, i=1  j=0
tid=7, i=1  j=0
tid=7, i=2  j=1
tid=7, i=2  j=1
tid=7, i=2  j=1
```



```
tid=7, i=3  j=2
tid=7, i=3  j=2
tid=2, i=2  j=4
tid=2, i=3  j=4
tid=2, i=3  j=4
tid=2, i=4  j=0
tid=2, i=5  j=0
tid=2, i=5  j=0
tid=2, i=5  j=1
tid=1, i=0  j=0
tid=1, i=0  j=1
tid=1, i=0  j=1
tid=1, i=1  j=1
tid=1, i=1  j=2
tid=1, i=1  j=2
tid=1, i=2  j=2
tid=1, i=2  j=3
tid=1, i=2  j=3
tid=6, i=0  j=4
tid=6, i=0  j=4
tid=6, i=1  j=0
tid=6, i=2  j=0
tid=6, i=2  j=0
tid=6, i=2  j=1
tid=6, i=3  j=1
tid=1, i=3  j=3
tid=1, i=3  j=4
tid=1, i=3  j=4
tid=5, i=5  j=3
tid=5, i=5  j=3
tid=6, i=3  j=1
tid=7, i=3  j=2
tid=7, i=4  j=3
tid=7, i=4  j=3
tid=7, i=4  j=3
tid=6, i=3  j=2
tid=6, i=4  j=2
tid=6, i=4  j=2
tid=6, i=4  j=3
tid=6, i=5  j=3
tid=6, i=5  j=3
tid=6, i=5  j=4
tid=0, i=0  j=0
tid=0, i=0  j=0
tid=0, i=0  j=0
tid=0, i=1  j=1
tid=0, i=1  j=1
tid=0, i=1  j=1
tid=0, i=2  j=2
tid=0, i=2  j=2
tid=0, i=2  j=2
tid=0, i=3  j=3
tid=0, i=3  j=3
tid=0, i=3  j=3
tid=0, i=4  j=4
```