National Institute of Technology Karnataka Surathkal,
Mangalore - 575025

DEPARTMENT OF INFORMATION TECHNOLOGY

LAB ASSIGNMENT 6

Submitted for Parallel Computing (IT301) By

Gaurav Chaurasia 181CV155

To

Dr. Geetha V *Dept of IT, NITK Surathkal*

Code link
github

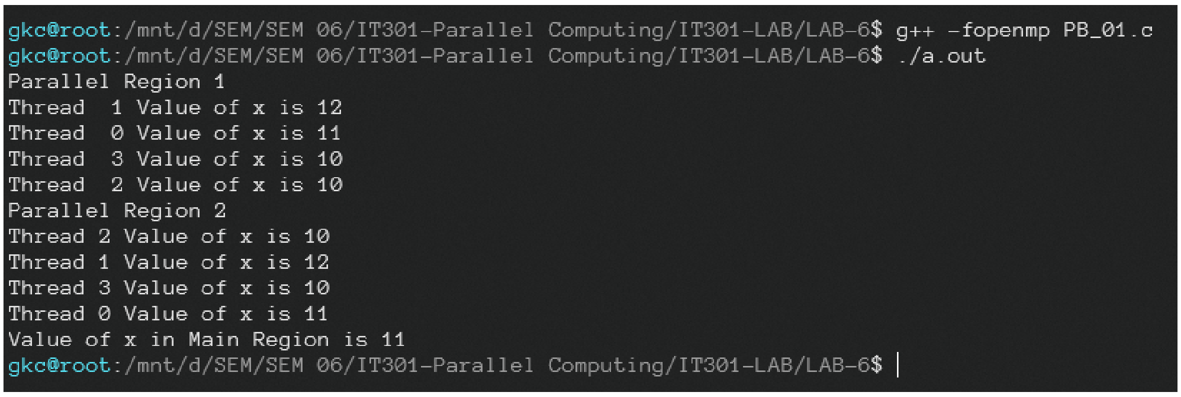
PROBLEM_01

Output using copyin() clause

Observation: - With copyin() clause

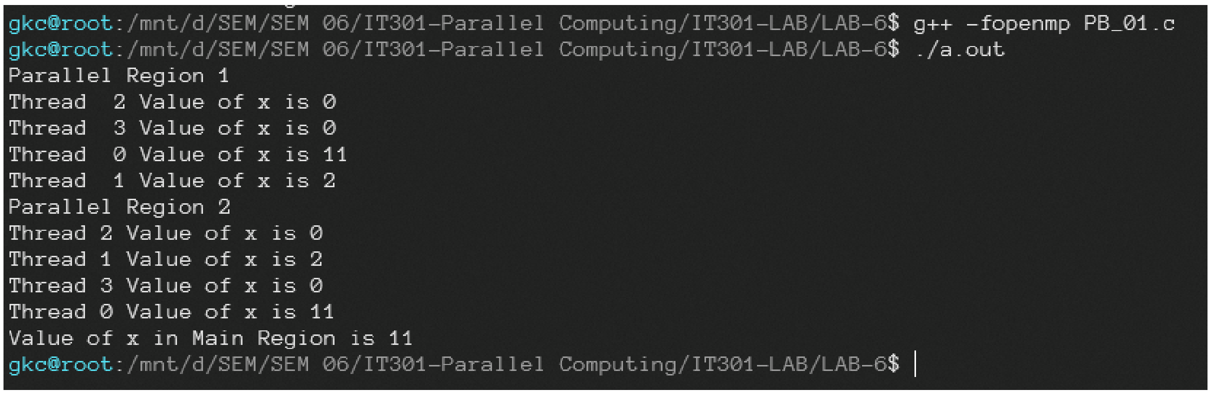
1. copyin(x) clause gives a copy of variable x to all the threads initially. 2. Master clause can be accessed only by master thread (id = 0).
3. barrier clause makes the threads wait until all the threads have completed the above operations.
4. Because of using threadprivate(x) clause, value of x is private to the threads which they are continuing in Parallel Region 2

```
#include <stdio.h>
#include <omp.h>
int tid, x;
#pragma omp threadprivate(x, tid)
int 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);
}
```



```
gkc@root:/mnt/d/SEM/SEM 06/IT301-Parallel Computing/IT301-LAB/LAB-6$ g++ -fopenmp PB_01.c
gkc@root:/mnt/d/SEM/SEM 06/IT301-Parallel Computing/IT301-LAB/LAB-6$ ./a.out
Parallel Region 1
Thread  1 Value of x is 12
Thread  0 Value of x is 11
Thread  3 Value of x is 10
Thread  2 Value of x is 10
Parallel Region 2
Thread 2 Value of x is 10
Thread 1 Value of x is 12
Thread 3 Value of x is 10
Thread 0 Value of x is 11
Value of x in Main Region is 11
gkc@root:/mnt/d/SEM/SEM 06/IT301-Parallel Computing/IT301-LAB/LAB-6$
```

Without copyin() clause

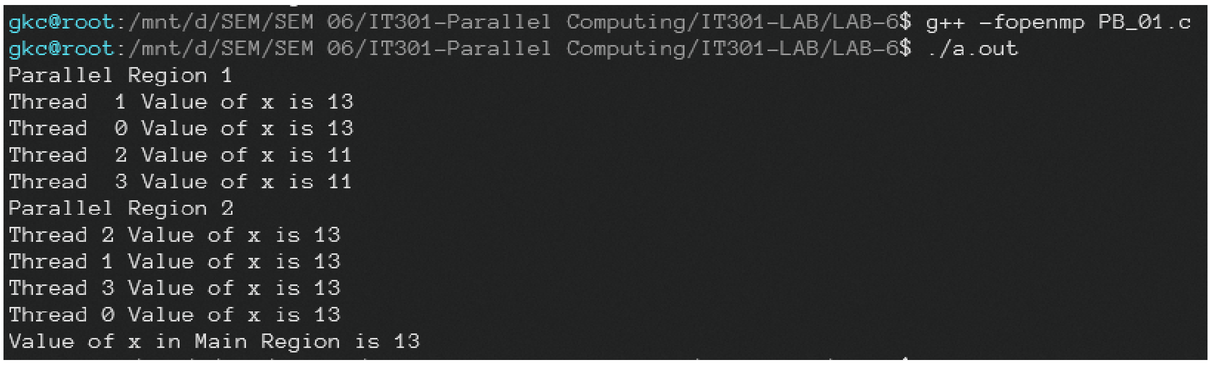


```
gkc@root:/mnt/d/SEM/SEM 06/IT301-Parallel Computing/IT301-LAB/LAB-6$ g++ -fopenmp PB_01.c
gkc@root:/mnt/d/SEM/SEM 06/IT301-Parallel Computing/IT301-LAB/LAB-6$ ./a.out
Parallel Region 1
Thread  2 Value of x is 0
Thread  3 Value of x is 0
Thread  0 Value of x is 11
Thread  1 Value of x is 2
Parallel 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
gkc@root:/mnt/d/SEM/SEM 06/IT301-Parallel Computing/IT301-LAB/LAB-6$
```

Here, x = 10 is not copied to all the threads. Only, master thread is given this value. Default value of x is 0 assigned to all other threads other than master threads.

After removing copyin() clause and initializing x globally

- X is assigned to all the threads as it is declared globally



```
gkc@root:/mnt/d/SEM/SEM 06/IT301-Parallel Computing/IT301-LAB/LAB-6$ g++ -fopenmp PB_01.c
gkc@root:/mnt/d/SEM/SEM 06/IT301-Parallel Computing/IT301-LAB/LAB-6$ ./a.out
Parallel Region 1
Thread  1 Value of x is 13
Thread  0 Value of x is 13
Thread  2 Value of x is 11
Thread  3 Value of x is 11
Parallel Region 2
Thread 2 Value of x is 13
Thread 1 Value of x is 13
Thread 3 Value of x is 13
Thread 0 Value of x is 13
Value of x in Main Region is 13
```

PROBLEM_02

Learn the concept of firstprivate() and threadprivate()

- Analyse the results for variable count and x.

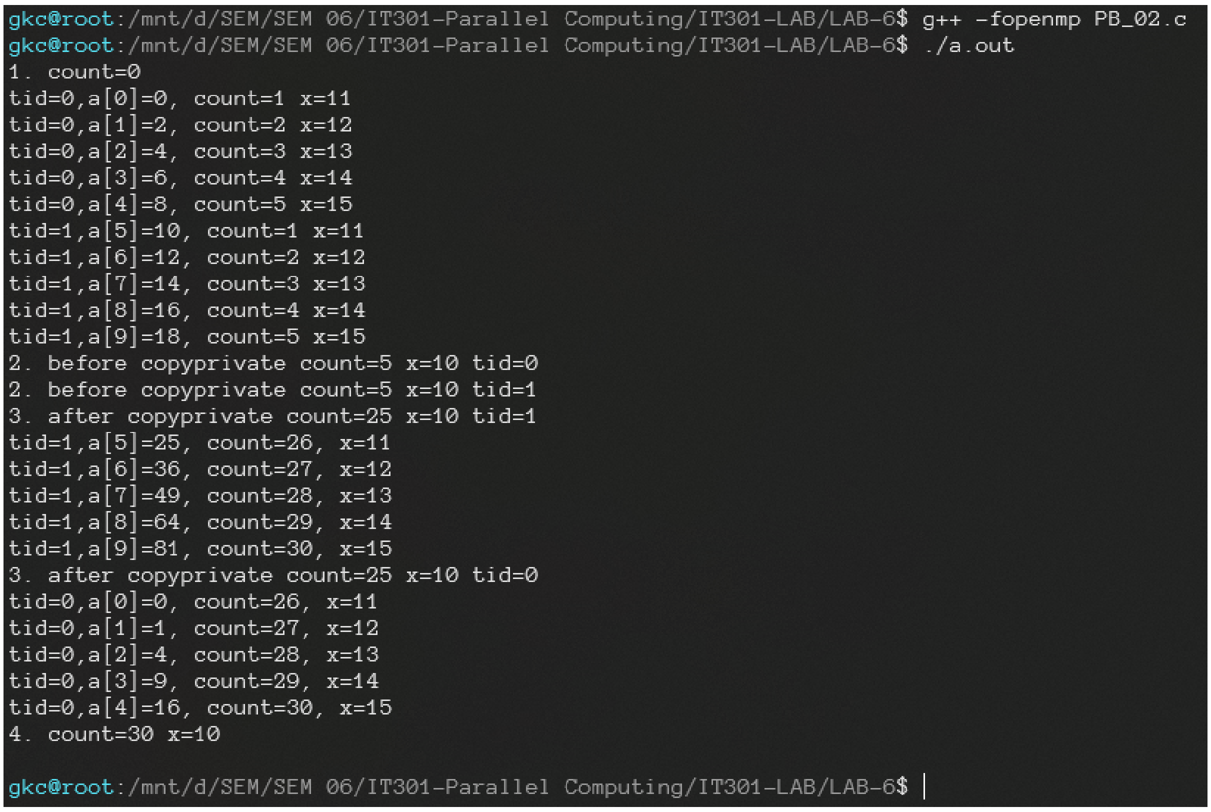
```
#include <stdio.h>
#include <stdlib.h>
#include <omp.h>
int count = 0;
#pragma omp threadprivate(count)

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;
}
```



```
gkc@root:/mnt/d/SEM/SEM 06/IT301-Parallel Computing/IT301-LAB/LAB-6$ g++ -fopenmp PB_02.c
gkc@root:/mnt/d/SEM/SEM 06/IT301-Parallel Computing/IT301-LAB/LAB-6$ ./a.out
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=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
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
4. count=30 x=10

gkc@root:/mnt/d/SEM/SEM 06/IT301-Parallel Computing/IT301-LAB/LAB-6$ |
```

variable: count

It is globally declared, so it's value is 0 throughout the program wherever no modification is done. Because of copyin(count), all threads are assigned count = 0. It is private to each thread as it is declared threadprivate(count). So, count increases for each thread from 1 to 5 and then because of single it's value increases to 20 + 5 = 25 but because of copyprivate(count), it's value is broadcasted to all other threads, so they also are printing count value from 26 to 30.

variable: x

The firstprivate(x) clause first assigns all the threads x = 10 and then this is private to each thread. X++ increases it's value and prints from 11 to 15 because of iteration (loop) for each thread. After copyprivate, x value is the same as declared earlier x = 10 (scope of variable). Then again because of firstprivate(x) and x ++, it's value is printed from 11 to 15

PROBLEM_03

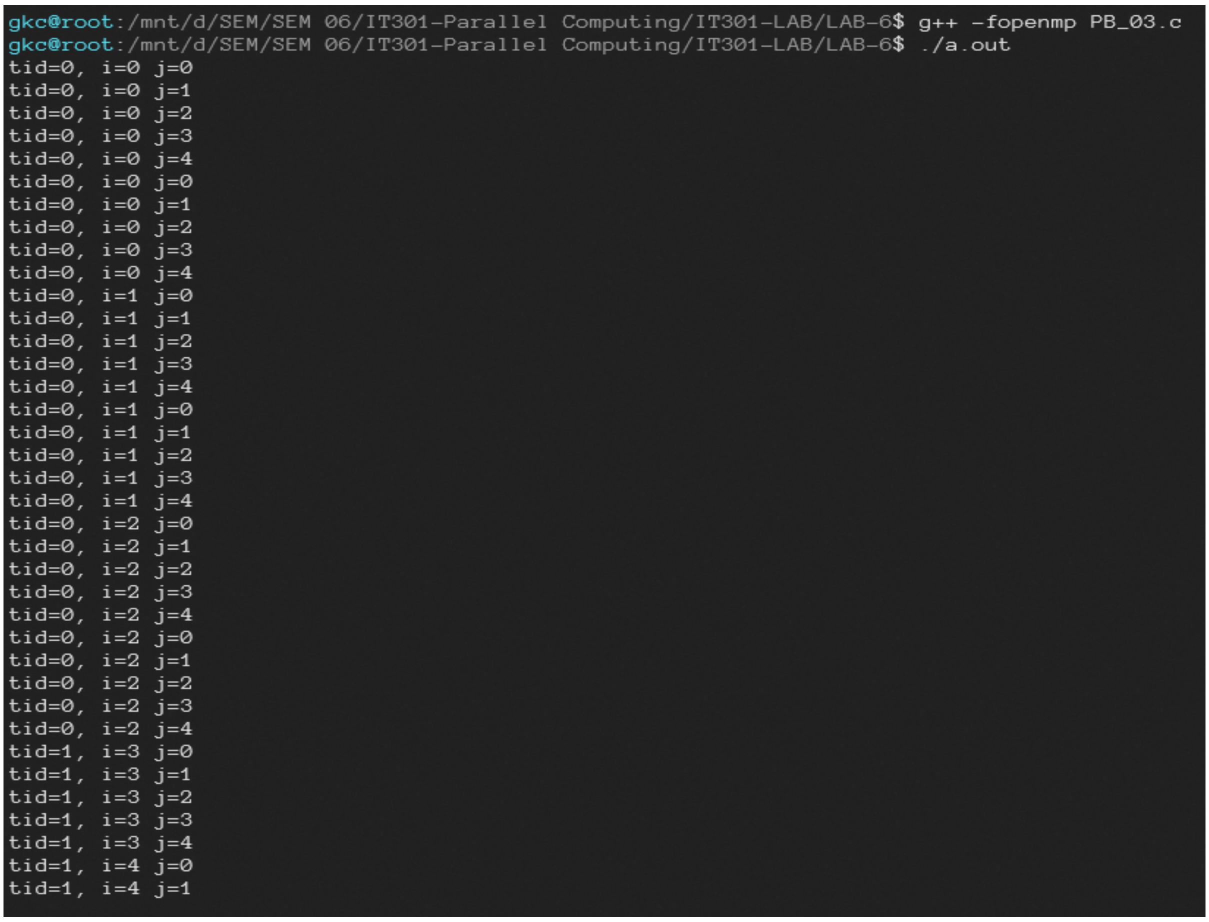
```
#include <stdio.h>
#include <stdlib.h>
#include <omp.h>
int main(void)
{
    int i, j, k;
#pragma omp parallel
    {
#pragma omp for schedule(static, 3) private(i, j)
        for (i = 0; i < 6; i++)
            for (k = 0; k < 2; k++) {
                for (j = 0; j < 5; j++) {
                 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

    return 0;
}
```

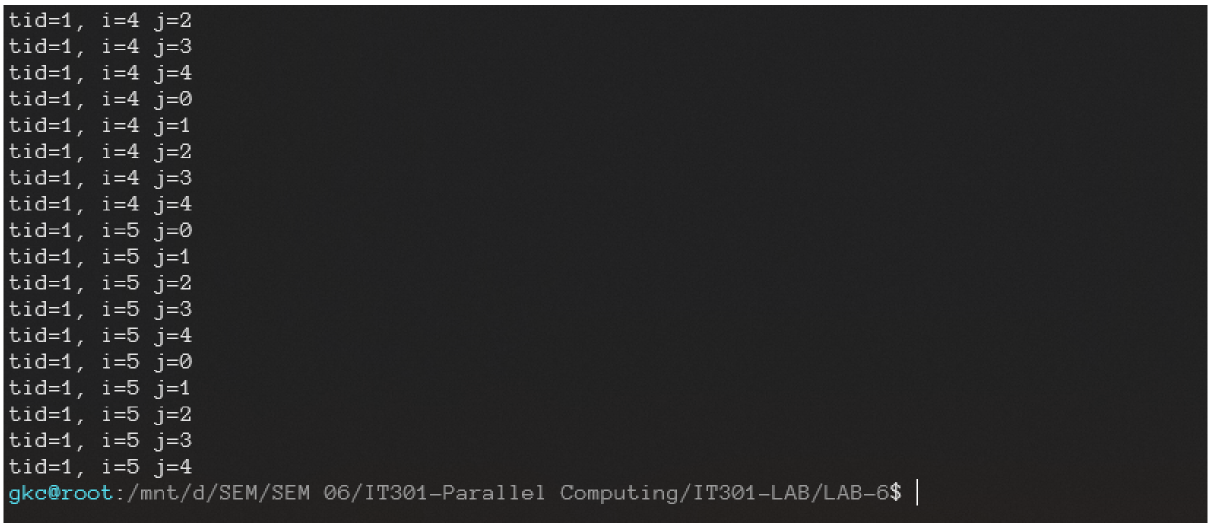
Output:

Without collapse()

Here, without collapse(), only due to schedule(static,3), each thread does three iterations considering i value. So, for tid = 0, i from 0 to 2 will perform then similarly for tid = 1.



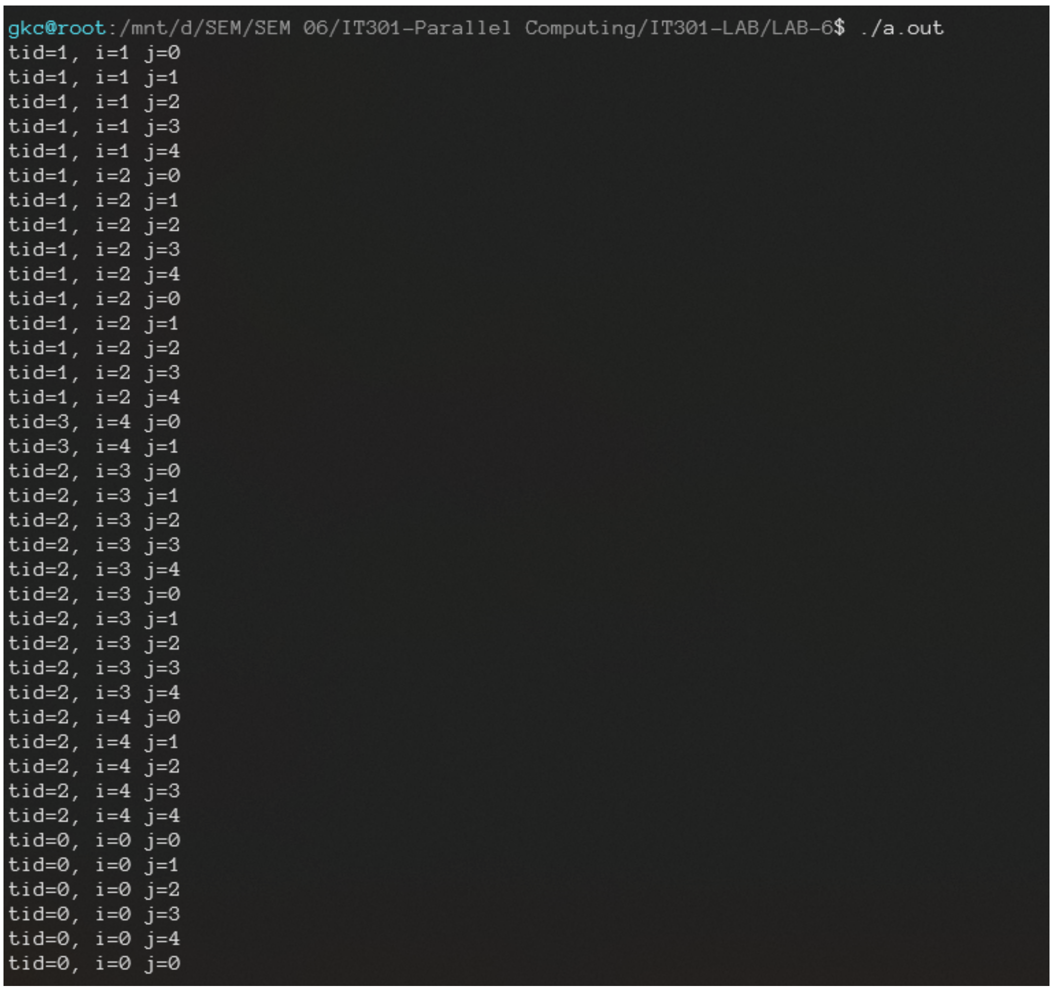
```
gkc@root:/mnt/d/SEM/SEM 06/IT301-Parallel Computing/IT301-LAB/LAB-6$ g++ -fopenmp PB_03.c
gkc@root:/mnt/d/SEM/SEM 06/IT301-Parallel Computing/IT301-LAB/LAB-6$ ./a.out
tid=0, i=0 j=0
tid=0, i=0 j=1
tid=0, i=0 j=2
tid=0, i=0 j=3
tid=0, i=0 j=4
tid=0, i=0 j=0
tid=0, i=0 j=1
tid=0, i=0 j=2
tid=0, i=0 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=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
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
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
```



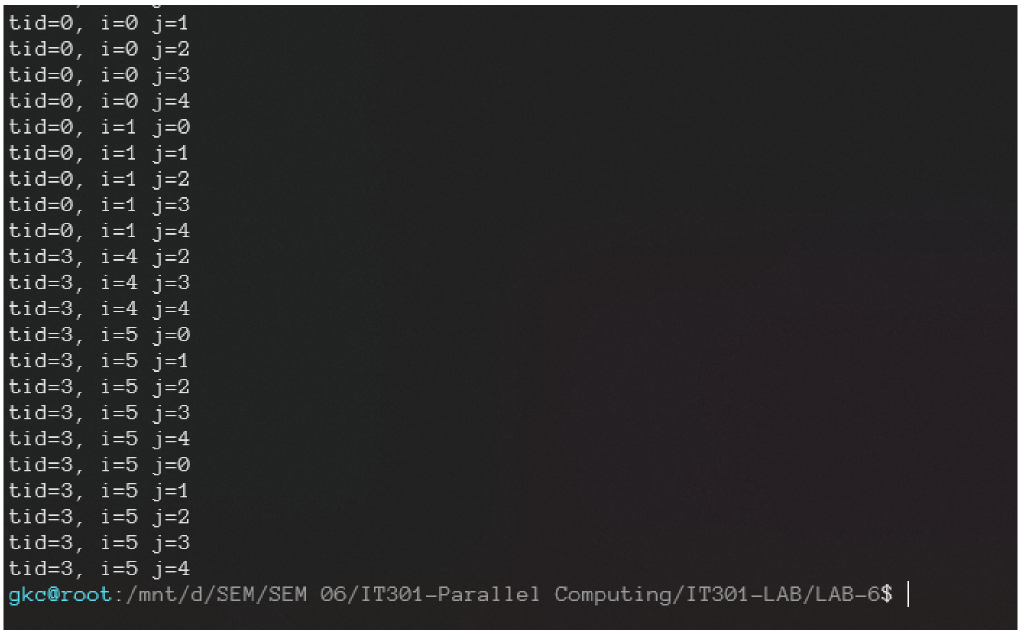
```
tid=1, i=4 j=2
tid=1, i=4 j=3
tid=1, i=4 j=4
tid=1, i=4 j=0
tid=1, i=4 j=1
tid=1, i=4 j=2
tid=1, i=4 j=3
tid=1, i=4 j=4
tid=1, i=5 j=0
tid=1, i=5 j=1
tid=1, i=5 j=2
tid=1, i=5 j=3
tid=1, i=5 j=4
tid=1, i=5 j=0
tid=1, i=5 j=1
tid=1, i=5 j=2
tid=1, i=5 j=3
tid=1, i=5 j=4
gkc@root:/mnt/d/SEM/SEM 06/IT301-Parallel Computing/IT301-LAB/LAB-6$
```

With collapse(2)

Here, combining two for loops (variable i and k), each thread does 3 iterations. Like tid = 0 does from i = 0, k = 0 to i = 1 to k = 0 then tid = 1 does i = 1, k= 1 to i = 2, k = 1 and so on



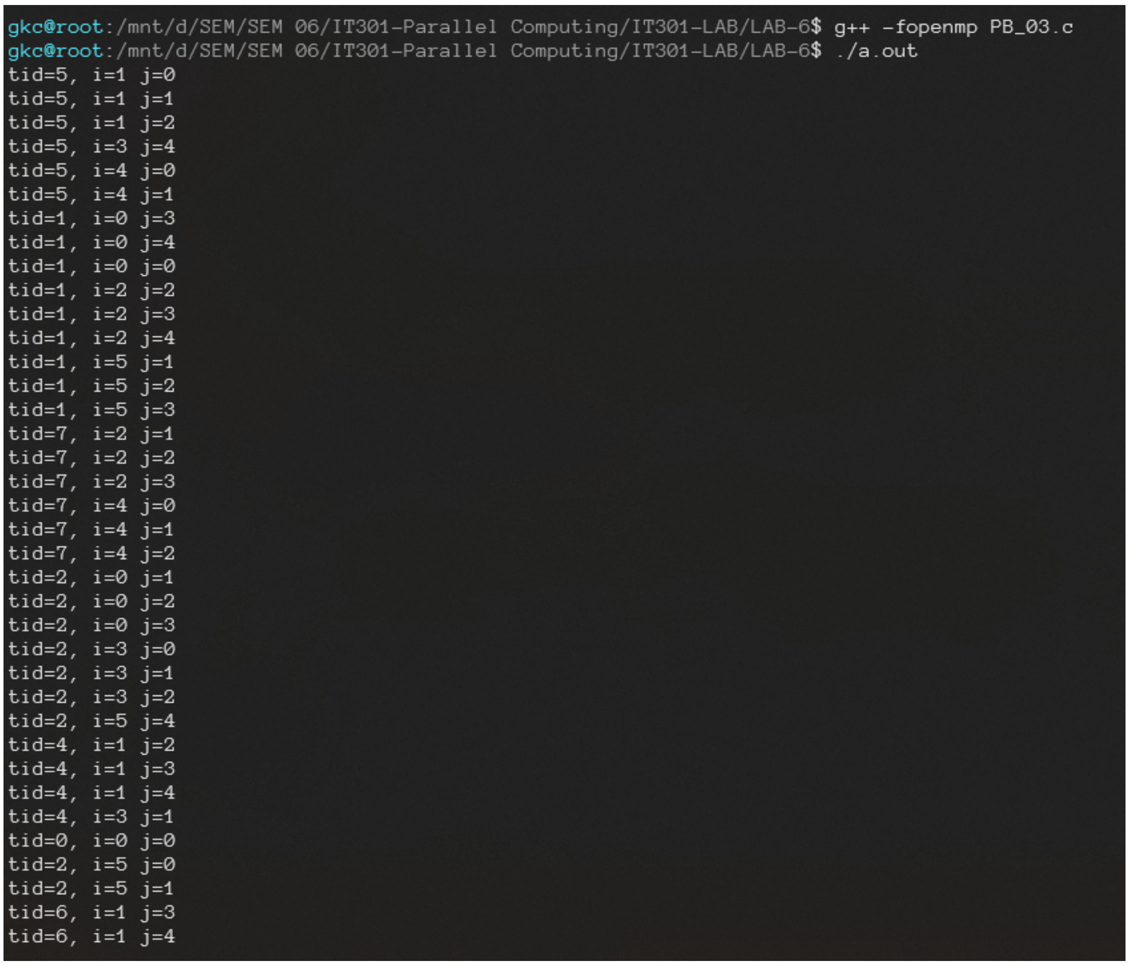
```
gkc@root:/mnt/d/SEM/SEM 06/IT301-Parallel Computing/IT301-LAB/LAB-6$ ./a.out
tid=1, i=1 j=0
tid=1, i=1 j=1
tid=1, i=1 j=2
tid=1, i=1 j=3
tid=1, i=1 j=4
tid=1, i=2 j=0
tid=1, i=2 j=1
tid=1, i=2 j=2
tid=1, i=2 j=3
tid=1, i=2 j=4
tid=1, i=2 j=0
tid=1, i=2 j=1
tid=1, i=2 j=2
tid=1, i=2 j=3
tid=1, i=2 j=4
tid=3, i=4 j=0
tid=3, i=4 j=1
tid=2, i=3 j=0
tid=2, i=3 j=1
tid=2, i=3 j=2
tid=2, i=3 j=3
tid=2, i=3 j=4
tid=2, i=3 j=0
tid=2, i=3 j=1
tid=2, i=3 j=2
tid=2, i=3 j=3
tid=2, i=3 j=4
tid=2, i=4 j=0
tid=2, i=4 j=1
tid=2, i=4 j=2
tid=2, i=4 j=3
tid=2, i=4 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=0 j=0
```



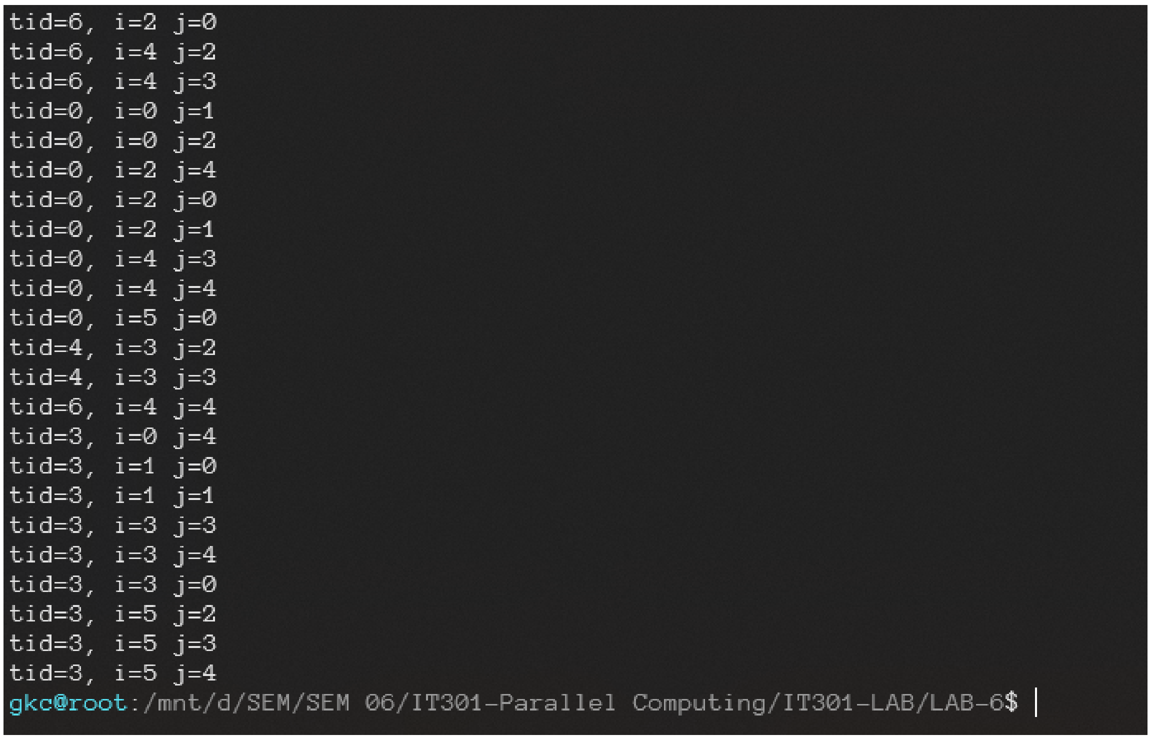
```
tid=0, i=0 j=1
tid=0, i=0 j=2
tid=0, i=0 j=3
tid=0, i=0 j=4
tid=0, i=1 j=0
tid=0, i=1 j=1
tid=0, i=1 j=2
tid=0, i=1 j=3
tid=0, i=1 j=4
tid=3, i=4 j=2
tid=3, i=4 j=3
tid=3, i=4 j=4
tid=3, i=5 j=0
tid=3, i=5 j=1
tid=3, i=5 j=2
tid=3, i=5 j=3
tid=3, i=5 j=4
tid=3, i=5 j=0
tid=3, i=5 j=1
tid=3, i=5 j=2
tid=3, i=5 j=3
tid=3, i=5 j=4
gkc@root:/mnt/d/SEM/SEM 06/IT301-Parallel Computing/IT301-LAB/LAB-6$
```

With collapse(3)

Here, combining all 3 loops, (variables i, k and j) each thread does 3 iterations. Example: - tid = 0 does iteration from i =0, k = 0, j = 0 to i=0,k=0,j=2 and then tid = 1 does iteration from i=0, k=0,j=3 to i=0, k=1,j=0 and so on



```
gkc@root:/mnt/d/SEM/SEM 06/IT301-Parallel Computing/IT301-LAB/LAB-6$ g++ -fopenmp PB_03.c
gkc@root:/mnt/d/SEM/SEM 06/IT301-Parallel Computing/IT301-LAB/LAB-6$ ./a.out
tid=5, i=1 j=0
tid=5, i=1 j=1
tid=5, i=1 j=2
tid=5, i=3 j=4
tid=5, i=4 j=0
tid=5, i=4 j=1
tid=1, i=0 j=3
tid=1, i=0 j=4
tid=1, i=0 j=0
tid=1, i=2 j=2
tid=1, i=2 j=3
tid=1, i=2 j=4
tid=1, i=5 j=1
tid=1, i=5 j=2
tid=1, i=5 j=3
tid=7, i=2 j=1
tid=7, i=2 j=2
tid=7, i=2 j=3
tid=7, i=4 j=0
tid=7, i=4 j=1
tid=7, i=4 j=2
tid=2, i=0 j=1
tid=2, i=0 j=2
tid=2, i=0 j=3
tid=2, i=3 j=0
tid=2, i=3 j=1
tid=2, i=3 j=2
tid=2, i=5 j=4
tid=4, i=1 j=2
tid=4, i=1 j=3
tid=4, i=1 j=4
tid=4, i=3 j=1
tid=0, i=0 j=0
tid=2, i=5 j=0
tid=2, i=5 j=1
tid=6, i=1 j=3
tid=6, i=1 j=4
```



```
tid=6, i=2 j=0
tid=6, i=4 j=2
tid=6, i=4 j=3
tid=0, i=0 j=1
tid=0, i=0 j=2
tid=0, i=2 j=4
tid=0, i=2 j=0
tid=0, i=2 j=1
tid=0, i=4 j=3
tid=0, i=4 j=4
tid=0, i=5 j=0
tid=4, i=3 j=2
tid=4, i=3 j=3
tid=6, i=4 j=4
tid=3, i=0 j=4
tid=3, i=1 j=0
tid=3, i=1 j=1
tid=3, i=3 j=3
tid=3, i=3 j=4
tid=3, i=3 j=0
tid=3, i=5 j=2
tid=3, i=5 j=3
tid=3, i=5 j=4
gkc@root:/mnt/d/SEM/SEM 06/IT301-Parallel Computing/IT301-LAB/LAB-6$
```