IT301 – Parallel Computing

Assignment – 2

Name: Niraj Nandish

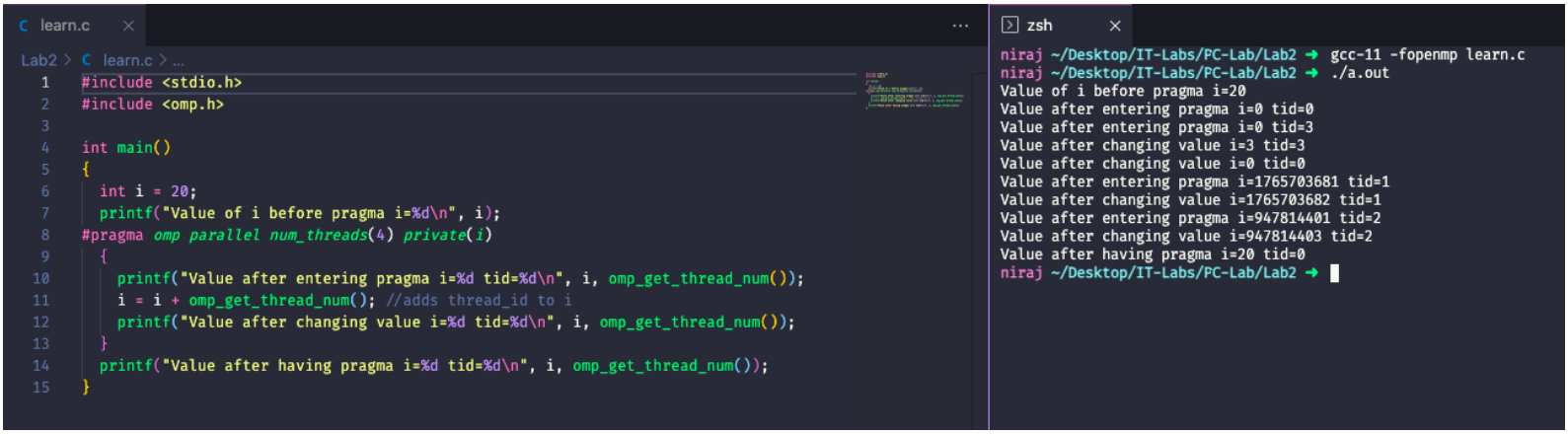
Roll No: 191IT234

1. Program 1
    a. Observation – The variable "x" is shared between all the threads and so it keeps incrementing without being reset to the initial value as all the threads are accessing the same variable (same address) and not a copy of it. But sometimes the value stays the same. This happens as there is no synchronization between the read and write of the values.

```
niraj ~/Desktop/IT-Labs/PC-Lab/Lab2 ➜ gcc-11 -fopenmp shared.c
niraj ~/Desktop/IT-Labs/PC-Lab/Lab2 ➜ ./a.out
Thread [1] value of x is 21
Thread [5] value of x is 24
Thread [4] value of x is 22
Thread [0] value of x is 26
Thread [3] value of x is 23
Thread [6] value of x is 27
Thread [2] value of x is 25
Thread [7] value of x is 28
niraj ~/Desktop/IT-Labs/PC-Lab/Lab2 ➜ export OMP_NUM_THREADS=4
niraj ~/Desktop/IT-Labs/PC-Lab/Lab2 ➜ ./a.out
Thread [0] value of x is 21
Thread [2] value of x is 23
Thread [3] value of x is 24
Thread [1] value of x is 22
```

```
niraj ~/Desktop/IT-Labs/PC-Lab/Lab2 ➜ export OMP_NUM_THREADS=16
niraj ~/Desktop/IT-Labs/PC-Lab/Lab2 ➜ ./a.out
Thread [4] value of x is 21
Thread [0] value of x is 31
Thread [3] value of x is 32
Thread [8] value of x is 24
Thread [9] value of x is 25
Thread [10] value of x is 26
Thread [11] value of x is 27
Thread [12] value of x is 28
Thread [13] value of x is 29
Thread [14] value of x is 30
Thread [1] value of x is 22
Thread [2] value of x is 33
Thread [7] value of x is 23
Thread [5] value of x is 34
Thread [6] value of x is 35
Thread [15] value of x is 36
niraj ~/Desktop/IT-Labs/PC-Lab/Lab2 ➜ ./a.out
Thread [1] value of x is 21
Thread [14] value of x is 35
Thread [15] value of x is 36
Thread [5] value of x is 24
Thread [2] value of x is 23
Thread [7] value of x is 26
Thread [4] value of x is 27
Thread [8] value of x is 28
Thread [9] value of x is 29
Thread [10] value of x is 30
Thread [11] value of x is 31
Thread [12] value of x is 32
Thread [13] value of x is 33
Thread [0] value of x is 34
Thread [3] value of x is 25
Thread [6] value of x is 22
niraj ~/Desktop/IT-Labs/PC-Lab/Lab2 ➜ ./a.out
Thread [3] value of x is 22
Thread [5] value of x is 23
Thread [4] value of x is 23
Thread [10] value of x is 27
Thread [9] value of x is 28
Thread [7] value of x is 32
Thread [11] value of x is 29
Thread [12] value of x is 30
Thread [8] value of x is 26
Thread [13] value of x is 31
Thread [14] value of x is 33
Thread [0] value of x is 34
Thread [2] value of x is 24
Thread [6] value of x is 25
Thread [15] value of x is 35
Thread [1] value of x is 21
```

2. Program 2
    a. Observation – As we can see here, private(x) initializes the value of x to 0 or a garbage value and performs the subsequent operations on the privately initialized x.



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

int main()
{
  int i = 20;
  printf("Value of i before pragma i=%d\n", i);
#pragma omp parallel num_threads(4) private(i)
  {
    printf("Value after entering pragma i=%d tid=%d\n", i, omp_get_thread_num());
    i = i + omp_get_thread_num(); //adds thread_id to i
    printf("Value after changing value i=%d tid=%d\n", i, omp_get_thread_num());
  }
  printf("Value after having pragma i=%d tid=%d\n", i, omp_get_thread_num());
}
```

```
niraj ~/Desktop/IT-Labs/PC-Lab/Lab2 ➜ gcc-11 -fopenmp learn.c
niraj ~/Desktop/IT-Labs/PC-Lab/Lab2 ➜ ./a.out
Value of i before pragma i=20
Value after entering pragma i=0 tid=0
Value after entering pragma i=0 tid=3
Value after changing value i=3 tid=3
Value after changing value i=0 tid=0
Value after entering pragma i=1765703681 tid=1
Value after changing value i=1765703682 tid=1
Value after entering pragma i=947814401 tid=2
Value after changing value i=947814403 tid=2
Value after having pragma i=20 tid=0
niraj ~/Desktop/IT-Labs/PC-Lab/Lab2 ➜
```

    b. Observation – As we can see here, firstprivate(x) initializes the value of x to the value it had when it encountered the parallel construct and performs the subsequent operations on the privately initialized x.

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

int main()
{
  int i = 20;
  printf("Value of i before pragma i=%d\n", i);
#pragma omp parallel num_threads(4) firstprivate(i)
  {
    printf("Value after entering pragma i=%d tid=%d\n", i, omp_get_thread_num());
    i = i + omp_get_thread_num(); //adds thread_id to i
    printf("Value after changing value i=%d tid=%d\n", i, omp_get_thread_num());
  }
  printf("Value after having pragma i=%d tid=%d\n", i, omp_get_thread_num());
}
```

```
niraj ~/Desktop/IT-Labs/PC-Lab/Lab2 ➜ gcc-11 -fopenmp learn.c
niraj ~/Desktop/IT-Labs/PC-Lab/Lab2 ➜ ./a.out
Value of i before pragma i=20
Value after entering pragma i=20 tid=0
Value after changing value i=20 tid=0
Value after entering pragma i=20 tid=1
Value after changing value i=21 tid=1
Value after entering pragma i=20 tid=3
Value after changing value i=23 tid=3
Value after entering pragma i=20 tid=2
Value after changing value i=22 tid=2
Value after having pragma i=20 tid=0
niraj ~/Desktop/IT-Labs/PC-Lab/Lab2 ➜
```

Since in both modes the variable was private, hence it makes no changes the original variable outside the parallel block.

3. Programming Exercise
    a. Observation – In the program we are parallelly computing the sum of a[i] & b[i] and storing the result in c[i]. We first set the number of threads and then ask the user how many elements should be present in the array. After that we split the array into chunks wherein each chunk is processed by each thread.
    b. Code

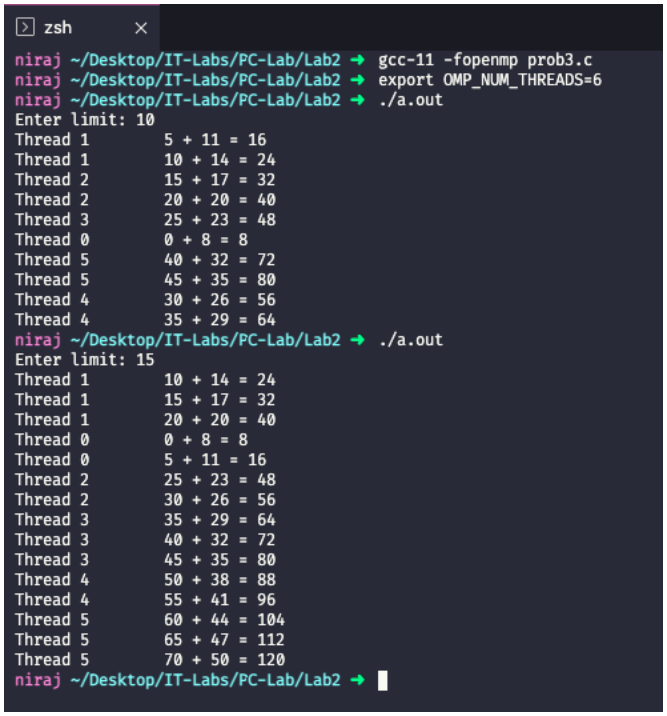
```c
#include <stdio.h>
#include <omp.h>

int main()
{
  int i, N, threadnum, numthreads, low, high;
  printf("Enter limit: ");
  scanf("%d", &N);
  int a[N], b[N], c[N];

  for (i = 0; i < N; i++)
  {
    a[i] = 5 * i;
    b[i] = 3 * i + 8;
  }

#pragma omp parallel default(shared) private(threadnum, numthreads, low, high, i)
  {
    int threadnum = omp_get_thread_num(), numthreads = omp_get_num_threads();
    int low = N * threadnum / numthreads, high = N * (threadnum + 1) / numthreads;

    for (i = low; i < high; i++)
    {
      c[i] = a[i] + b[i];
      printf("Thread %d\t%d + %d = %d\n", threadnum, a[i], b[i], c[i]);
    }
  }
}
```


zsh
niraj ~/Desktop/IT-Labs/PC-Lab/Lab2 ➜ gcc-11 -fopenmp prob3.c
niraj ~/Desktop/IT-Labs/PC-Lab/Lab2 ➜ export OMP_NUM_THREADS=6
niraj ~/Desktop/IT-Labs/PC-Lab/Lab2 ➜ ./a.out
Enter limit: 10
Thread 1 5 + 11 = 16
Thread 1 10 + 14 = 24
Thread 2 15 + 17 = 32
Thread 2 20 + 20 = 40
Thread 3 25 + 23 = 48
Thread 0 0 + 8 = 8
Thread 5 40 + 32 = 72
Thread 5 45 + 35 = 80
Thread 4 30 + 26 = 56
Thread 4 35 + 29 = 64
niraj ~/Desktop/IT-Labs/PC-Lab/Lab2 ➜ ./a.out
Enter limit: 15
Thread 1 10 + 14 = 24
Thread 1 15 + 17 = 32
Thread 1 20 + 20 = 40
Thread 0 0 + 8 = 8
Thread 0 5 + 11 = 16
Thread 2 25 + 23 = 48
Thread 2 30 + 26 = 56
Thread 3 35 + 29 = 64
Thread 3 40 + 32 = 72
Thread 3 45 + 35 = 80
Thread 4 50 + 38 = 88
Thread 4 55 + 41 = 96
Thread 5 60 + 44 = 104
Thread 5 65 + 47 = 112
Thread 5 70 + 50 = 120
niraj ~/Desktop/IT-Labs/PC-Lab/Lab2 ➜