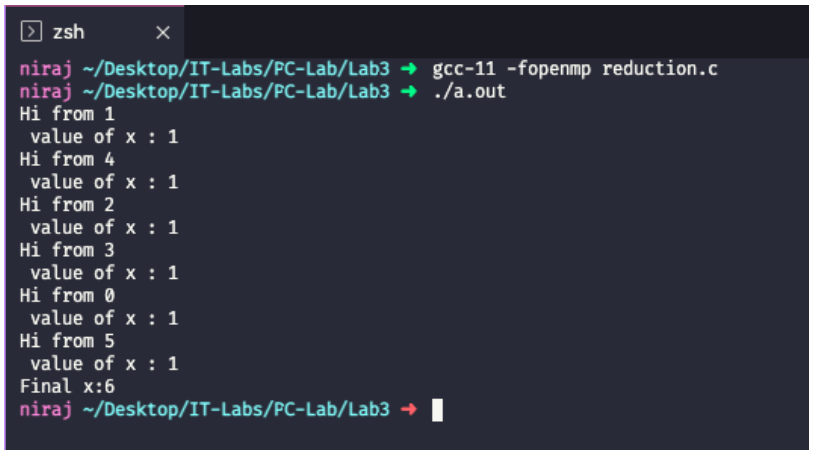
IT301 – Parallel Computing

Assignment – 3

Name: Niraj Nandish

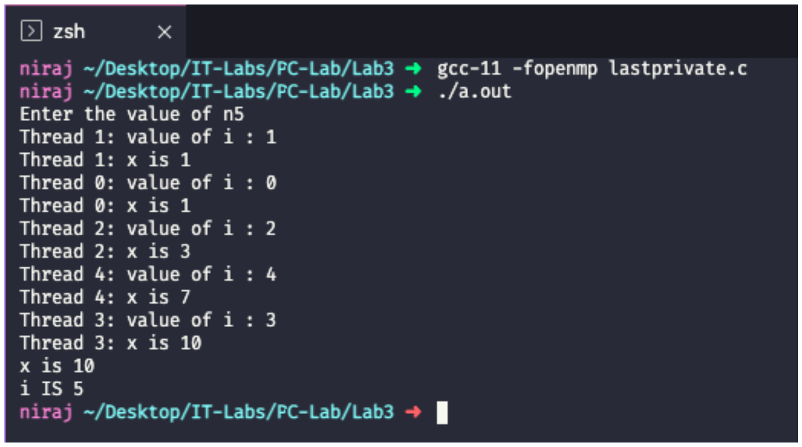
Roll No: 191IT234

1. Program 1
   a. Observation – The variable "x" has a value of 0 before it encounters the parallel construct. Since we are using the **reduction** clause with operator "+", each thread will then get a private copy of "x" with initialized value of 0. The threads then perform the subsequent operations on the privately initialized "x". After all threads have finished their operations, the original "x" outside the parallel construct is updated with the final values of "x" from each thread using the respective operator, in this case they are summed up. Therefore, the final value is 6, 0 is the initial value and 6 threads each having final value of "x" as 1.



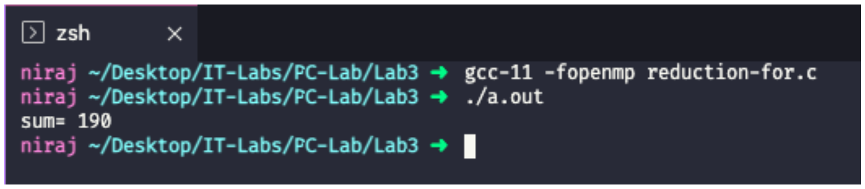
```
niraj ~/Desktop/IT-Labs/PC-Lab/Lab3 ➜ gcc-11 -fopenmp reduction.c
niraj ~/Desktop/IT-Labs/PC-Lab/Lab3 ➜ ./a.out
Hi from 1
 value of x : 1
Hi from 4
 value of x : 1
Hi from 2
 value of x : 1
Hi from 3
 value of x : 1
Hi from 0
 value of x : 1
Hi from 5
 value of x : 1
Final x:6
niraj ~/Desktop/IT-Labs/PC-Lab/Lab3 ➜ 
```

2. Program 2
    a. Observation – When **lastprivate(i)** is used, the value of the variable "i" is updated to the final value "i" of the last thread that performs operations on it inside the parallel construct. Unlike **firstprivate** where the value is taken in from the main region to the parallel region, **lastprivate** transfers the value from the parallel region to the main region. In the image below, the variable "i" has the value 5 in the last thread and so that value is transferred after all threads have completed their operations and updated into the original variable "i" outside the parallel region.



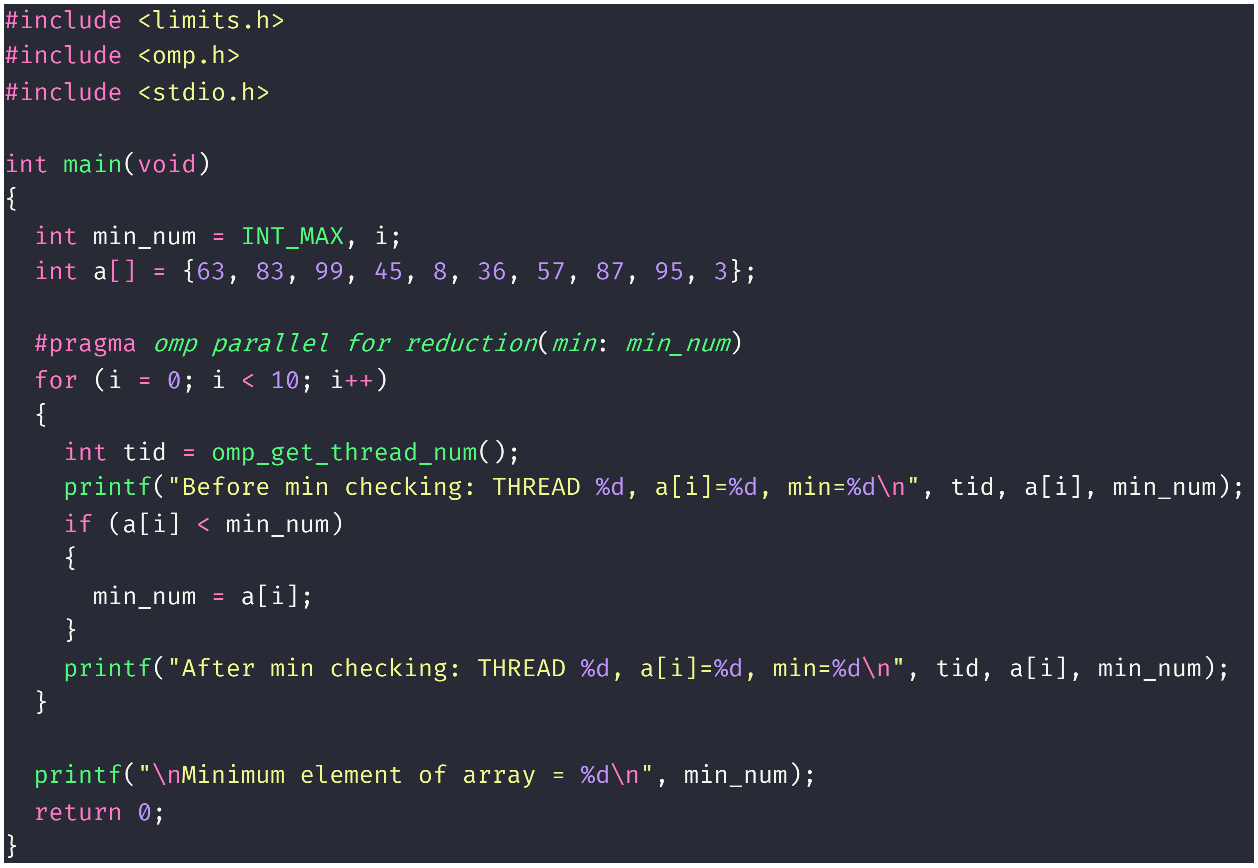
```
niraj ~/Desktop/IT-Labs/PC-Lab/Lab3 ➜ gcc-11 -fopenmp lastprivate.c
niraj ~/Desktop/IT-Labs/PC-Lab/Lab3 ➜ ./a.out
Enter the value of n5
Thread 1: value of i : 1
Thread 1: x is 1
Thread 0: value of i : 0
Thread 0: x is 1
Thread 2: value of i : 2
Thread 2: x is 3
Thread 4: value of i : 4
Thread 4: x is 7
Thread 3: value of i : 3
Thread 3: x is 10
x is 10
i IS 5
niraj ~/Desktop/IT-Labs/PC-Lab/Lab3 ➜ 
```

3. Program 3



```
niraj ~/Desktop/IT-Labs/PC-Lab/Lab3 ➜ gcc-11 -fopenmp reduction-for.c
niraj ~/Desktop/IT-Labs/PC-Lab/Lab3 ➜ ./a.out
sum= 190
niraj ~/Desktop/IT-Labs/PC-Lab/Lab3 ➜ 
```

4. Programming Exercise
    a. Observation – We first initialize the variable "min_num" with the maximum value of an **int** variable i.e. 2147483647. When the code encounters the parallel construct, each thread creates its own private copy of "min_num" with an initialized value of maximum value of **int** variable. Each thread then finds the minimum number of the array values it comes across. After all the threads have finished their operations, the final values of "min_num" in each thread & the value in the original "min_num" are reduced with the **min** operator and assigned back to the "min_num" variable outside the parallel construct.
    b. Code

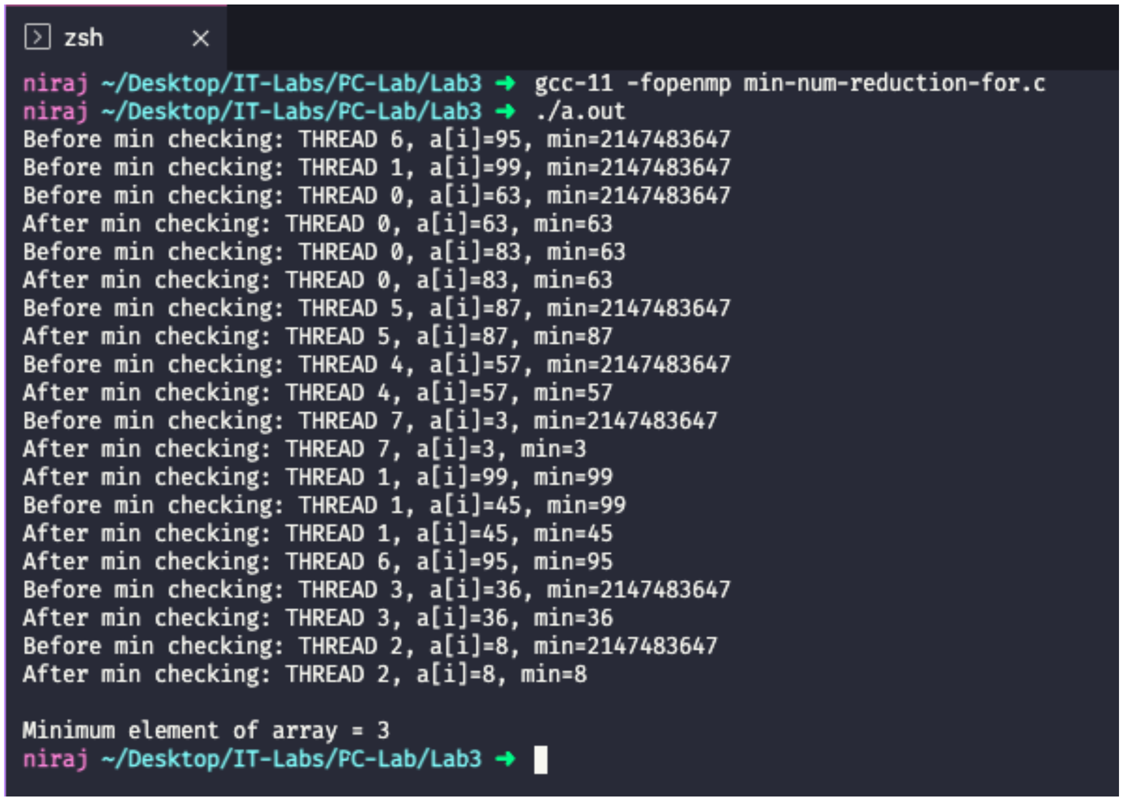


```c
#include <limits.h>
#include <omp.h>
#include <stdio.h>

int main(void)
{
  int min_num = INT_MAX, i;
  int a[] = {63, 83, 99, 45, 8, 36, 57, 87, 95, 3};

  #pragma omp parallel for reduction(min: min_num)
  for (i = 0; i < 10; i++)
  {
    int tid = omp_get_thread_num();
    printf("Before min checking: THREAD %d, a[i]=%d, min=%d\n", tid, a[i], min_num);
    if (a[i] < min_num)
    {
      min_num = a[i];
    }
    printf("After min checking: THREAD %d, a[i]=%d, min=%d\n", tid, a[i], min_num);
  }

  printf("\nMinimum element of array = %d\n", min_num);
  return 0;
}
```



```
niraj ~/Desktop/IT-Labs/PC-Lab/Lab3 ➜ gcc-11 -fopenmp min-num-reduction-for.c
niraj ~/Desktop/IT-Labs/PC-Lab/Lab3 ➜ ./a.out
Before min checking: THREAD 6, a[i]=95, min=2147483647
Before min checking: THREAD 1, a[i]=99, min=2147483647
Before min checking: THREAD 0, a[i]=63, min=2147483647
After min checking: THREAD 0, a[i]=63, min=63
Before min checking: THREAD 0, a[i]=83, min=63
After min checking: THREAD 0, a[i]=83, min=63
Before min checking: THREAD 5, a[i]=87, min=2147483647
After min checking: THREAD 5, a[i]=87, min=87
Before min checking: THREAD 4, a[i]=57, min=2147483647
After min checking: THREAD 4, a[i]=57, min=57
Before min checking: THREAD 7, a[i]=3, min=2147483647
After min checking: THREAD 7, a[i]=3, min=3
After min checking: THREAD 1, a[i]=99, min=99
Before min checking: THREAD 1, a[i]=45, min=99
After min checking: THREAD 1, a[i]=45, min=45
After min checking: THREAD 6, a[i]=95, min=95
Before min checking: THREAD 3, a[i]=36, min=2147483647
After min checking: THREAD 3, a[i]=36, min=36
Before min checking: THREAD 2, a[i]=8, min=2147483647
After min checking: THREAD 2, a[i]=8, min=8

Minimum element of array = 3
niraj ~/Desktop/IT-Labs/PC-Lab/Lab3 ➜
```