1. Write a OpenMP program to sort an array on n elements using both sequential and parallel mergesort(using Section). Record the difference in execution time.

COMMAND:

- **Create**: gedit prg1.c
- **Compile**: gcc -fopenmp prg1.c -o prg1
- **Run**: export OMP_NUM_THREADS=4 && ./prg1

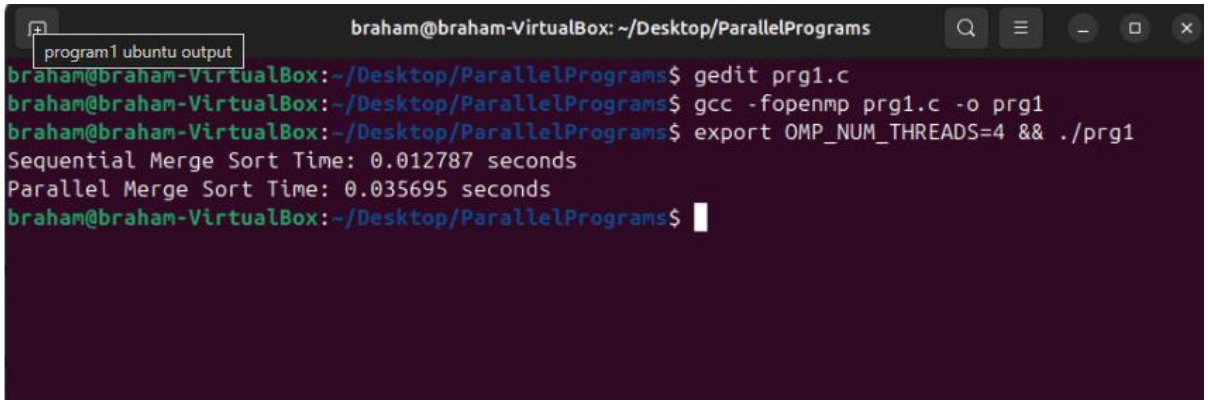
PROGRAM:

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

void merge(int arr[], int l, int m, int r) {
  int i = l, j = m + 1, k = 0;
  int temp[r - l + 1];

  while (i <= m && j <= r) {
    if (arr[i] <= arr[j])
      temp[k++] = arr[i++];
    else
      temp[k++] = arr[j++];
  }
  while (i <= m) temp[k++] = arr[i++];
  while (j <= r) temp[k++] = arr[j++];

  for (i = l, k = 0; i <= r; i++, k++)
    arr[i] = temp[k];
}
```

```
void sequentialMergeSort(int arr[], int l, int r) {
    if (l < r) {
        int m = (l + r) / 2;
        sequentialMergeSort(arr, l, m);
        sequentialMergeSort(arr, m + 1, r);
        merge(arr, l, m, r);
    }
}

void parallelMergeSort(int arr[], int l, int r) {
    if (l < r) {
        int m = (l + r) / 2;

        #pragma omp parallel sections
        {
            #pragma omp section
            parallelMergeSort(arr, l, m);

            #pragma omp section
            parallelMergeSort(arr, m + 1, r);
        }

        merge(arr, l, m, r);
    }
}

int main() {
    int n = 100000;
```

```
    int arr1[n], arr2[n];

    for (int i = 0; i < n; i++) {
        arr1[i] = rand() % 1000;
        arr2[i] = arr1[i];
    }

    double start, end;

    start = omp_get_wtime();
    sequentialMergeSort(arr1, 0, n - 1);
    end = omp_get_wtime();
    printf("Sequential Merge Sort Time: %f seconds\n", end - start);

    start = omp_get_wtime();
    parallelMergeSort(arr2, 0, n - 1);
    end = omp_get_wtime();
    printf("Parallel Merge Sort Time: %f seconds\n", end - start);

    return 0;
}
```

OUTPUT:



2. Write an OpenMP program that divides the Iterations into chunks containing 2 iterations, respectively (OMP_SCHEDULE=static,2). Its input should be the number of iterations, and its output should be which iterations of a parallelized for loop are executed by which thread. For example, if there are two threads and four iterations, the output might be the following:
a. Thread 0 : Iterations 0 –– 1
b. Thread 1 : Iterations 2 –– 3

COMMAND:

- **Create**: gedit prg2.c
- **Compile**: gcc -fopenmp prg2.c -o prg2
- **Run**: export OMP_NUM_THREADS=4 && ./prg2

PROGRAM:

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

int main() {
  int n;
  printf("Enter number of iterations: ");
  scanf("%d", &n);

  int thread_start[100], thread_end[100];
  int i;

  for (i = 0; i < 100; i++) {
    thread_start[i] = -1;
    thread_end[i] = -1;
  }

  #pragma omp parallel for schedule(static,2)
```

```
    for (i = 0; i < n; i++) {
        int tid = omp_get_thread_num();

        if (thread_start[tid] == -1)
            thread_start[tid] = i;
        thread_end[tid] = i;
    }

    for (i = 0; i < 100; i++) {
        if (thread_start[i] != -1) {
            printf("Thread %d : Iterations %d -- %d\n", i, thread_start[i], thread_end[i]);

        }
    }

    return 0;
}
```

OUTPUT: