1. Write a program to sort an array on n elements using both sequential and parallel merge sort. Record the difference in execution time.
2. Write the program to calculate the value of area under a curve using trapezoidal rule. Use Critical directive to parallelize the code. Record the execution time in both serial and parallel versions.
3. Write the program to calculate the value of area under a curve using trapezoidal rule. Use reduction clause to parallelize the code. Record the execution time in both serial and parallel versions.
4. Estimate the value of pi using:

   Parallelize the code by removing loop carried dependency and record both serial and parallel execution times.
5. Write a program to sort n elements using odd-even transposition sort. Record serial and parallel execution times.
6. Write an OpenMP program that determines the default scheduling of parallel for loops. 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:
   Thread 0 : Iterations 0 -- 1
   Thread 1 : Iterations 2 – 3

7. Write a program to calculate n Fibonacci numbers using Parallel Directive. Demonstrate elimination of the race condition using Schedule directive.
8. Write a program to find the prime numbers from 1 to n employing parallel for directive. Record both serial and parallel execution times.