

**Course:** Parallel and Distributed Programming

**Module: PDP**

**Semester:** Semester II

**Assignment Number: 2 OpenMP and MPI C programming**

**Date of Title Issue:** 27th November 2024

**Assignment Deadline: 20**th. December 2024

**Assignment Weighting:** 20/50

**Please state the assignment title / brief. Please specify details such as:** Answer the questions on the accompanying sheet.

**Learning Outcomes**

**Please state the programme and related module learning outcomes that this assignment is assessing.** 1,2,4, 5,6

**Assessment Criteria**

Please state the assessment criteria applied to this assignment, such as:

* Correctness of the work.
* Presentation, including compliance with the specified file format.
* Evidence of critical thinking and analysis.
* Originality, quality and thoroughness of the work.
* Research correct academic approach.
* Proper treatment of sources.

*Academic Dishonesty: All of your assignments need to represent your own effort. Assignments should be done without consultation with other students and you should not share your source code with others. Any assignment submitted that is essentially the same, as someone else’s will not be accepted****. ALL matching assignments will receive 0 credits.***

Your task is to search for a **palindrome string of length (n)** in a two-dimensional matrix of random single characters between a-z, the idea of the solution is to parallelize the search between all possible threads and count each occurrence. The matrix size is 1000x1000, you can input the number you want to search or hard code it, search right to left, up to down and diagonally up to down. Test your code with all possible threads incrementing one by one as sample results below.

**Make sure to include timing to your code**, you need to submit two solutions using MPI and OpenMP:

1. **OpenMP C file and results of execution. 50%**

2. **Repeat part 1 using MPI C file and results of execution. 50%**

**Code below can be used to generate the matrix:**

**for(i = 0;i < ROWS;i++)**

**for(j = 0;j < COLUMNS;j++)**

**a[i][j] = (rand() % 26) + 'A';**

**Hint: start your testing with a small matrix, 10x10 to make sure it is working, then scale up to 1000x1000.**

**You can use function for the search or string libraries.**

**Example of execution output:**

Palindrome search

﻿115308 palindromes of size 3 found in 0.004378 s. using 1 threads.

4394 palindromes of size 4 found in 0.003098 s. using 1 threads.

4420 palindromes of size 5 found in 0.003105 s. using 1 threads.

182 palindromes of size 6 found in 0.012899 s. using 1 threads.

\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*

115308 palindromes of size 3 found in 0.008779 s. using 2 threads.

4394 palindromes of size 4 found in 0.003641 s. using 2 threads.

4420 palindromes of size 5 found in 0.003185 s. using 2 threads.

182 palindromes of size 6 found in 0.003517 s. using 2 threads.

\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*

115308 palindromes of size 3 found in 0.003589 s. using 3 threads.

4394 palindromes of size 4 found in 0.003143 s. using 3 threads.

4420 palindromes of size 5 found in 0.013208 s. using 3 threads.

182 palindromes of size 6 found in 0.008625 s. using 3 threads.

\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*

115308 palindromes of size 3 found in 0.008677 s. using 4 threads.

4394 palindromes of size 4 found in 0.017414 s. using 4 threads.

4420 palindromes of size 5 found in 0.003351 s. using 4 threads.

182 palindromes of size 6 found in 0.003495 s. using 4 threads.

\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*

115308 palindromes of size 3 found in 0.003005 s. using 5 threads.

4394 palindromes of size 4 found in 0.002990 s. using 5 threads.

4420 palindromes of size 5 found in 0.003169 s. using 5 threads.

182 palindromes of size 6 found in 0.003276 s. using 5 threads.

\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*

115308 palindromes of size 3 found in 0.015719 s. using 6 threads.

4394 palindromes of size 4 found in 0.003616 s. using 6 threads.

4420 palindromes of size 5 found in 0.003430 s. using 6 threads.

182 palindromes of size 6 found in 0.003543 s. using 6 threads.

\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*

115308 palindromes of size 3 found in 0.003260 s. using 7 threads.

4394 palindromes of size 4 found in 0.003472 s. using 7 threads.

4420 palindromes of size 5 found in 0.003414 s. using 7 threads.

182 palindromes of size 6 found in 0.003308 s. using 7 threads.

\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*

115308 palindromes of size 3 found in 0.003278 s. using 8 threads.

4394 palindromes of size 4 found in 0.003190 s. using 8 threads.

4420 palindromes of size 5 found in 0.003299 s. using 8 threads.

182 palindromes of size 6 found in 0.014600 s. using 8 threads.

\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*