**COMSATS University Islamabad, Lahore Campus** 

**Lab Examination FALL 2025**

| Course Title: Parallel and Distributed Computing Course Code: | | | | CSC334 | Credit Hours: | 3(2,1) |
| --- | --- | --- | --- | --- | --- | --- |
| Course Instructor: Akhzar Nazir Programme Name: | | | | BAI | | |
| Semester: 6th | Batch: | SP23-BAI | Section: A | Date: | 27/10/2025 | |
| **Time Allowed: 2:30 Hours Maximum Marks:** | | | | | **100** | |
| **Important Instructions / Guidelines:**  ● Be precise and to the point while answering any question.  ● Show all your work, as partial credits will be given. You will be graded not only on the correctness of your answer, but also on the clarity with which you express it. Please be neat.  ● **Good luck!** | | | | | | |

**Questions: *CLO: <4>*** *Implement parallel programming algorithms using appropriate parallel and distributed computing paradigms.* ***:<Applying>***

**Duration:** 2:30 hours

**Format:** Open Book, Open Internet

**Allowed:** AI tools for syntax/coding help

**Prohibited:** Chatting, emailing, or collaborating with classmates

**Dataset (Download from Kaggle or any other source)**

**A folder named images\_dataset/ containing 4 subfolders (e.g., cats, dogs, flowers, cars) with 20–30 images each.**

**Download Images from the given link**

**https://drive.google.com/drive/folders/1f0dg1u-xUu4e\_ZsWU-IzdDlYRLGuHIGY?usp=sharing Tasks**

**Task 1: Sequential Preprocessing (20 marks)**

**Write a Python script sequential\_process.py that:**

**●** Reads all images (using Pillow or OpenCV or any other library)

**●** Resizes each to 128x128

● And add water mark in image

**●** Saves them into output\_seq/ keeping class folders intact

● Prints total execution time

**Example Output:**

Sequential Processing Time: 18.24 seconds

**Task 2: Parallel Processing (30 marks)**

**Write parallel\_process.py that performs the same operation, but in parallel using multiprocessing or ThreadPoolExecutor.**

Requirements:

● Use at least 2, 4, and 8 worker threads/processes

**●** Save results in output\_parallel/

● Measure and print time for each configuration

● Display a simple speedup table

**Output Format:**

Workers | Time (s) | Speedup

-------- | -------- | -------

1 | 18.24 | 1.00x

4 | 6.12 | 2.98x

8 | 4.88 | 3.73x

**Task 3: Simulated Distributed Task (30 marks)**

**Simulate a distributed environment using multiprocessing.Manager() or queue:**

● Divide total images equally among 2 “nodes” (processes)

● Each node works on its subset and reports its completion time

● Master process aggregates total time and prints the summary

● No need to arrange 2 systems, manage it in one machine and simulate it like 2 machines logically **Output example:**

Node 1 processed 40 images in 5.8s

Node 2 processed 40 images in 6.0s

Total distributed time: 6.1s

Efficiency: 2.99x over sequential

**Task 4 — Short Report (20 marks)**

**Create a small file report.txt or report.pdf containing: [10 Marks] Upload on Github with Readme File [10 Marks]**

● Table comparing sequential, parallel, and distributed times

● Best number of workers (explain why)

**●** One paragraph discussing:

“How parallelism improved performance and what bottlenecks still exist.”

**Submission Format (Same is for GitHub)**

**Zip your folder as <rollno>\_exam.zip with structure:**

**├── sequential\_process.py**

**├── parallel\_process.py**

**├── distributed\_sim.py**

**├── report.txt / report.pdf**

**└── sample\_output/**

**Submit to Google Classroom + Github before the deadline.**

**Hints**

**● Use os.cpu\_count() to check core count.**

**● Measure time using time.perf\_counter().**

**● For Mac/AMD systems, use CPU parallelism only (no CUDA required). ● Use multiprocessing.Pool for best performance.**

**● You can test locally using small images.**