**The final exam test**

**Block 1**

1. Describe the ways computers execute programs in parallel computing
2. Describe the main challenges of parallel programming compared to traditional sequential programming
3. Describe three Python libraries used for parallel programming
4. Write about two types of memory organization in parallel computing
5. Describe the shared memory in parallel computing
6. Describe the distributed memory in parallel computing
7. Write about various features in the distributed memory
8. Describe low-performance computing cluster
9. Describe high-performance computing cluster
10. Write about multiprocessing on a GPU

**Block 2**

1. Explain the concepts of deadlock and race condition in the context of parallel programming
2. Characterize processes and threads in parallel computing
3. Write about a global lock in multithreaded Python programs
4. Describe the RLock object of the threading module
5. Write about the Deadlock bad thread synchronization situation
6. Describe Mpi4py in parallel computing
7. Describe strategies to optimize communication in an MPI program written with mpi4py
8. Describe how Python's asyncio module supports asynchronous programming
9. Define Numba and explain its primary purpose in Python programming
10. Discuss the use of Numba for customizing NumPy universal functions

**Block 3**

1. Write a Python script using the threading module for parallel computing
2. Use Python's multiprocessing module to write a program that calculates the sum of a list of numbers
3. Write a Python program using multiprocessing where multiple processes increment a shared counter
4. Write a Python code that deals with the deadlock situation
5. Write an example of a deadlock in a Python multithreading application and modify the code to resolve it
6. Write a Python code for matrix multiplication with multiprocessing library
7. Write a Numba-optimized function to perform element-wise addition of two large NumPy arrays
8. Write a Numba function that uses GPU acceleration to multiply two matrices
9. Provide a Numba-optimized function that includes error handling
10. Create a Numba-optimized function that operates on multi-dimensional NumPy arrays