Exam

NAME:

Instructions: Write your solutions with a pen *clearly* and *succinctly* directly on this booklet. You may not use extra resources such as books, notepads or portable electronic devices.

Please sign below to confirm that you understand that, according to the regulations of the Faculty of Informatics, violation of the above instructions can lead to you failing this exam.

Signature: ____________________

Grades

Sec. #	you got	out of
1		6
2		12
3		8
4		7
5		4
total		37

Choose the correct answer (only one is correct): /12

1 OpenMP

Questions	Answers
1. OpenMP is	☐ a programming language ☐ a parallelization library ☐ a set of compiler directives accompanied with a runtime library that allows you to easily parallelize a code on shared memory architectures ☐ a utility that allows you to parallelize your code on distributed memory architectures
2. Assuming OMP_NUM_THREADS=8, when an OpenMP parallel region ("omp parallel") is encountered by the master thread,	☐ a set of 7 additional threads will be created and will execute in parallel the following block of code along with the master thread ☐ a set of 8 additional threads will be created and will execute in parallel the following block of code along with the master thread ☐ a set of as many threads as the number of available cores will be created and will execute in parallel the following block of code ☐ no threads will be created until an "omp for" directive is encountered
3. If an "omp barrier" directive is encountered in an OpenMP parallel region	☐ a single random thread will execute the next block of code ☐ all the threads will execute the next block of code one after the other sequentially ☐ the master thread only will execute the following block of code ☐ the execution will not proceed further until all the threads of the parallel region reach the barrier
4. If a variable is declared as "firstprivate" in an OpenMP parallel region, this means that	☐ a copy of the variable will be placed by the compiler in the L2 cache of each core ☐ the variable will be copied from the master thread to each individual thread of the parallel region ☐ a variable of the same type will be created in each individual thread of the parallel region ☐ the variable will be private for the first subsequent for loop but shared for the rest

Questions	Answers
5. When an “omp for” / "omp do" directive precedes a for / do loop, then	☐ the iteration space of the following for / do loop will be partitioned across all the threads of the current OpenMP parallel region ☐ a new thread will be created for each iteration of the loop ☐ you can safely assume that the first iteration of the loop will always be executed by the master thread ☐ each thread of the parallel region will execute the whole loop
6. The Amdahl’s law states that the execution time T(N) of a parallel application with N threads	☐ will always be equal to T(1)/T(N), where T(1) is the execution time with a single thread ☐ will be proportional to the time it takes for a main memory request to be served ☐ will always be bound by the fraction of the application that remained strictly serial ☐ cannot be predicted

2 MPI

Questions	Answers
1. MPI is useful	☐ only for shared memory systems ☐ only for distributed memory systems ☐ both for shared memory and distributed memory systems ☐ only for Cray systems
2. What is the last MPI routine that all processes must call in an MPI program?	☐ MPI_Abort ☐ MPI_Finalize ☐ MPI_Init ☐ MPI_Initialized
3. When an MPI_Send routine returns, which of the following is always true?	☐ The receiver has received all of the message ☐ The variable that has been sent (the send buffer) is safe to be modified by the caller ☐ The send was a "synchronous" send ☐ The receiver must have called MPI_Recv
4. The MPI_Send routine	☐ is always synchronous ☐ is always buffered ☐ might be synchronous or buffered, depending on the implementation ☐ is an example of non-blocking communication
5. A message can be sent and will be received	☐ between different communicators if the sender and receiver ranks are the same in both communicators ☐ between different communicators if the sender and receiver tags are the same for both communicators ☐ between different communicators if the sender and receiver tags are the same and the ranks are the same in each communicator ☐ only within the same communicator
6. Wildcarding allows	☐ the sender to broadcast a messages to all processes instead of just one ☐ the sender to send a message to the next receiving process ☐ the receiver to receive from any source or with any tag ☐ two messages sent from one rank to another rank to overtake each other

Questions	Answers
7. With which MPI routine can I determine my left and right neighbours in a Cartesian topology?	☐ MPI_Cart_coords ☐ MPI_Cart_create ☐ MPI_Cart_shift ☐ MPI_Cart_create
8. If I have a total of n processes, what function can I call that will create a sensible division of these processors in a Cartesian grid?	☐ MPI_Dims_create ☐ MPI_Cart_rank ☐ MPI_Cart_coords ☐ MPI_Graph_create
9. All MPI programs must contain a call to MPI_Init. This routine must be called before all other MPI routines, with one exception:	☐ MPI_Abort ☐ MPI_Initialized ☐ MPI_Finalize ☐ MPI_Comm_size
10. MPI_Reduce will	☐ send chunks of an array from the root process to other processes ☐ send a single piece of data from the root process to all other processes ☐ compute an operation on elements from different processes and store the result on the root process ☐ compute an operation on elements from different processes and store the result on all processes
11. If you are using MPI with OpenMP, how should you initialize MPI?	☐ MPI_Init ☐ MPI_Init_thread ☐ MPI_Funneled ☐ MPI_Initialize_OpenMP
12. MPI_Barrier:	☐ Stops all MPI processes in a communicator until all reach the barrier ☐ Completes all communications within a communicator ☐ Ensures it is safe to use a sent buffer ☐ Finalizes an MPI session

3 I/O

Choose the correct answer (only one is correct): /8

Questions	Answers
1. Parallelizing I/O	☐ Will lead to perfect scalability of the code ☐ Helps reducing the total computational time ☐ Does not matter, it is always a minor part of the execution ☐ It is not possible
2. Writing files in parallel	☐ Is possible only using MPI ☐ Always improves performance, independently from the file system ☐ Is always safe if implemented with MPI ☐ Always improves performance, if implemented with MPI
3. A binary file written with MPI	☐ Can be read by a non MPI program ☐ Can be read only using the same number of MPI tasks it was written with ☐ Follows a precise standard ☐ Can be read only by a C/C++ program
4. MPI non blocking I/O	☐ Is not supported by MPI ☐ Cannot be combined to collective I/O ☐ Is supported since it avoids deadlocks ☐ Can improve the performance by overlapping I/O to computation
5. MPI collective I/O	☐ Improves performance because all MPI tasks can write at the same time to disk ☐ Improves performance since only one MPI task write files ☐ Improves performance since MPI tasks do not communicate ☐ Improves performance since the number I/O operations is optimized
6. HDF5 files	☐ Are portable but needs HDF5 library to be read ☐ Are portable and can be easily read by any program ☐ Are portable and can be easily read only by Fortran programs ☐ Are portable and their size is always smaller than any other file format

Questions	Answers
7. HDF5 provides	☐ An OpenMP based library to do parallel I/O ☐ An MPI based library to do parallel I/O ☐ Does not support parallel I/O ☐ A CUDA based library to do parallel I/O
8. Strong scalability test:	☐ Measures the scalability of a MPI code ☐ Measures how performance changes increasing the problem size together with the number of parallel tasks ☐ Measures how performance changes increasing the problem size and keeping the number of parallel tasks the same ☐ Measures the scalability of a parallel code for a specific problem size with increasing the number of parallel tasks

4 CUDA

Questions	Answers
1. In CUDA terminology, when a kernel is launched, the set of threads that run on the same SMX is called a	☐ thread grid ☐ thread set ☐ thread gang ☐ thread block
2. A reduction between n threads in the same thread block can be optimally performed with complexity	☐ constant O(1) ☐ linear O(n) ☐ logarithmic O($log(n)$) ☐ quadratic O(n^2)
3. If host and gpu access managed memory simultaneously the following occurs	☐ a segmentation fault ☐ performance slowdown ☐ undefined behavior ☐ managed memory ensures ensures that all reads and writes are correctly ordered
4. To make the host code wait for all kernels to finish before proceeding call	☐ __syncthreads() ☐ cudaDeviceSynchronize() ☐ MPI_Barrier() ☐ nothing: kernel launches return when the kernel has finished
5. A race condition may occur when	☐ multiple threads access an address with at least one write ☐ multiple threads read the same location in device memory ☐ host and device write to the same managed memory ☐ all of the above
6. Thread A can reliably read a memory update by thread B if	☐ thread B was in the same thread block and __syncthreads() has been called ☐ thread A and thread B both access the memory using atomics (e.g. atomicAdd()) ☐ thread B was in another kernel that finished before thread A's kernel launched ☐ all of the above
7. Memory reads for a kernel are fastest from	☐ host memory ☐ shared memory ☐ hard disk ☐ device memory

5 OpenACC

Questions	Answers
1. Currently, OpenACC	☐ Is supported only by NVIDIA ☐ Is supported by various compilers ☐ Is supported only by PGI ☐ Is supported only on CRAY systems
2. When you declare a data region	☐ You have to ensure that the needed arrays are properly created and initialized on the GPU ☐ You are sure that the compiler will copy all the needed arrays to the GPU ☐ All the instructions until the end of the data region will be executed on the GPU ☐ The compiler creates a CUDA kernel
3. The "parallel" directive	☐ Executes the first following loop on the GPU assigning one iteration per thread ☐ Parallelizes the first loop that follows on the CPU ☐ Distributes the work that follows among all the available threads ☐ Creates a kernel that is executed by all gangs redundantly
4. Assuming A is an array of 128 real number elements, then the directive "acc data create(A[0:128])" in C or "acc data create(A)" in Fortran	☐ Will allocate an array of 128 "real" elements on the main memory of the CPU (host) ☐ Will allocate an array of 128 "real" elements on the main memory of the GPU ☐ Will allocate an array of 128 "real" elements on the shared memory of the GPU ☐ Will allocate an array of 128 "real" elements on the main memory of the GPU and copy the values stored to the CPU