1. **Course number and name**
   CMP 418 – Multicore Computing

2. **Credits and contact hours**
   3 credit hours, 3 contact hours

3. **Instructor's or course coordinator's name**
   Dr. Gerassimos Barlas

4. **Textbook, title, author, and year**
   G. Barlas. *Multicore and GPU Programming: An Integrated Approach*. 1st edition, Morgan Kaufmann, 2014.

   Other supplemental materials
   I. Foster. *Designing and Building Parallel Programs: Concepts and Tools for Parallel Software Engineering*, Addison-Wesley Pub Co, 1995, also available on-line at http://www.mcs.anl.gov/dbpp/
   Selected material from online sources:
   - MPI 3.0 specification
   - OpenMP 4.0 specification
   - NVidia's CUDA tutorials and reference material

5. **Specific course information**
   a. Brief description of content of the course (catalog description)
   Covers models of parallel computation and software development on multicore systems. Examines problem decomposition patterns including divide-and-conquer, geometric decomposition, task parallelism and pipelining. Covers program structure patterns such as master-worker, map-reduce and fork-join. Provides hands-on experience with high-performance multicore platforms, including both Central Processing Unit and Graphics Processing Unit architectures and state-of-the-art software tools.

   b. Prerequisites or co-requisites
   Prerequisites: CMP310/COE 381 (Operating Systems)

   c. Indicate whether a required, elective, or selected elective course in the program
   Selected Elective

6. **Specific goals for the course**
   a. Specific outcomes of instruction
   This course requires the student to demonstrate the following:
   1. Explain Flynn's taxonomy of computer architectures.
   2. Describe the most important contemporary multicore machine architectures.
   3. Calculate speedup and efficiency to measure the performance of a parallel algorithm.
   4. Use MPI to write a process-level parallel program.
   5. Transform a sequential program into a multi-threaded one using OpenMP compiler-directives.
   6. Design and implement programs running on GPUs by utilizing the CUDA platform.
   7. Employ different software design patterns (e.g. master-worker, pipelining, map-reduce, etc.) and data organization techniques (distributed data structures) for efficient utilization of multicore platforms.

b. Explicitly indicate which of the student outcomes listed in Criterion 3 or any other outcomes are addressed by the course

This course contributes in a significant way to the accomplishment of the following program outcomes:

Program outcome	Emphasis in this course
(a) an ability to apply knowledge of computing and mathematics appropriate to the discipline	◑
(b) an ability to analyze a problem, and identify and define the computing requirements appropriate to its solution	●
(c) an ability to design, implement, and evaluate a computer-based system, process, component or program to meet desired needs	◑
(d) an ability to function effectively on teams to accomplish a common goal	
(e) an understanding of professional, ethical, legal, security and social issues and responsibilities	
(f) an ability to communicate effectively with a range of audiences	
(g) an ability to analyze the local and global impact of computing on individuals, organizations, and society	
(h) recognition of the need for and an ability to engage in continuing professional development	○
(i) an ability to use current techniques, skills, and tools necessary for computing practice	●
(j) an ability to apply mathematical foundations, algorithmic principles, and computer science theory in the modeling and design of computer-based systems in a way that demonstrates comprehension of the tradeoffs involved in design choices	◑
(k) an ability to apply design and development principles in the construction of software systems of varying complexity	◑

Emphasis: ● High; ◑ Medium; ○ Low; Blank – Nothing Specific Expected

7. Brief list of topics to be covered

i. Parallel computers, taxonomies, architectures
ii. Metrics for a quantitative analysis of parallel algorithms
iii. Parallel program design methodologies
iv. Advanced Programming with Qt threads
v. Message-passing computing, MPI
vi. Programming with threads: OpenMP
vii. Programming GPUs using CUDA
viii. Case studies: searching and sorting algorithms, numerical algorithms, image processing