Course unit title:	Programming in Unix-Like Environments
Course unit code:	INS500
Type of course unit: **(Compulsory/Optional)**	Optional
Level of course unit: **(First, second or third cycle)**	Master (2nd Cycle)
Year of study:	Foundation Year
Semester when the unit is delivered:	1
Number of ECTS credits allocated:	6
Name of lecturer(s):	TBA

Learning outcomes of the course unit:

Upon successful completion of this course students should be able to:

- Explain the basic Linux concepts.
- Write intermediate to advanced level C++ code.
- Employ memory effectively in the context of computer programs.
- Use stand-alone debuggers to find persistent errors in source code.

Mode of delivery:	Face- to- face
Prerequisites and co-requisites:	None
Recommended optional program components:	None

Course Contents:

Objective:

To introduce the concepts of Unix-like operating systems, demonstrate the concepts of the C++ programming language, further the understanding of memory manipulation of personal computers, and introduce tools and techniques of debugging.

Description:

Introduction:

Linux basic concepts – file and directory management, line and screen editors, changing file and directory attributes, differentiation between binary and text files, creating text files from the command line, familiarization with the system files of Linux, basic c-shell scripting.

Overview of C++:
Makefiles, constant and variable types, expressions and operators, flow control structures, function definitions, header files, separate compilation and linking, understanding the compiling process, understanding the linking process.

Object-oriented programming with C++:
Class definitions, inheritance, encapsulation, polymorphism and operator overloading using C++ structures, comparison between object-based and object-oriented programming.

Memory management:
Pointers (referencing and dereferencing), memory models to accommodate pointers, pointer arithmetic, argument passing (by value and by reference), pointers to pointers, pointers as arrays, pointers to objects, abstract data structures, file processing, dynamic object creation and deletion.

Command line compilation and debugging:
Basic debugging using cout statements. Using debuggers: getting to know gdb, using gdb to identify errors, breakpoints and watches.

Recommended or required reading:	Savitch, W. (2006). Problem Solving with C++. Boston, MA: Addison-Wesley Oualline, S. (2003). Practical C++ Programming. Beijing, China: O' Reilly Press. Welsh, M., Dalheimer, M. K., Dawson, T., & Kaufman, L. (2005). Running Linux. Beijing, China: O' Reilly Press. Johnson, M. K., & Troan, E. W. (2008). Linux Application Development. Boston, MA: Addison-Wesley. Stallman, R. M., Pesch, R. H., & Shebs, S. (2002). Debugging with GDB: The GNU Source-Level Debugger Boston, MA: Free Software Foundation. Matloff, N. (2008). The Art of Debugging with GDB, DDD. San Francisco, CA: No Starch Press.
Planned learning activities and teaching methods:	Class Instruction 42 Hours Consultation 30 Hours

Assessment methods and criteria:	Examinations 50% Assignments / Class Participation 50% 100%
Language of instruction:	English
Work placement(s):	No
Place of Teaching:	Regular Classroom European University Cyprus, Nicosia Computer Laboratory European University Cyprus, Nicosia