UNIVERSIDAD AUTÓNOMA DE MADRID

17823 - PROGRAMMING PROJECT

Information of the subject

Code - Course title: 17823 - PROGRAMMING PROJECT

Degree: 473 - Graduado/a en Ingeniería Informática
474 - Graduado/a en Ingeniería Informática y Matemáticas
722 - Graduado/a en Ingeniería Informática
734 - Graduado/a en Ingeniería Informática y Matemáticas (2019)

Faculty: 350 - Escuela Politécnica Superior

Academic year: 2020/21

1. Course details

1.1. Content area

Programación y estructuras de datos

1.2. Course nature

Compulsory

1.3. Course level

Grado (EQF/MECU 6)

1.4. Year of study

474 - Graduado/a en Ingeniería Informática y Matemáticas: 2
473 - Graduado/a en Ingeniería Informática: 1
734 - Graduado/a en Ingeniería Informática y Matemáticas (2019): 2
722 - Graduado/a en Ingeniería Informática: 1

1.5. Semester

First semester o Second semester

1.6. ECTS Credit allotment

6.0

1.7. Language of instruction

Español, English

1.9. Recommendations

In order to take this course and to take the maximum advantage from it, you will need a capacity to: read and understand

Spanish and English texts (English texts if the course is offered in English), search and evaluate bibliographic references, write technical reports in Spanish (viz. English if the course is offered in English).

We recommend that before taking this course you take and pass the course **Programming** I (Programación I) and the **Software Seminar** (Seminario-Taller de Software), or some equivalent course. The former provides the knowledge and basic programming skills that you will need in order to follow this course, while the latter will give you the basic knowledge and working skill in operating systems, development tools, and collaboration systems that you will use in this course.¡

PPROG is part of the Programming and Data Structure unit (Programación y Estructuras de Datos) which, together with the courses *Programación I* y *Programación II* constitutes the *Programming* subject. PPROG complements and unifies the knowledge and the skills that you have acquired in the other courses of this module.

PPROG creates a basis for the courses *Proyecto de Análisis y Diseño de Software*, *Proyecto de Autómatas y Lenguajes*, *Proyecto de Sistemas Informáticos* y *Proyecto de Ingeniería del Software*.

1.10. Minimum attendance requirement

The course proposes two alternative itineraries, the first requiring mandatory class attendance, the other not requiring it. You will have to choose one itinerary at the beginning of the course and comply with the requirements corresponding to the itinerary that you have chosen. Such requirements are published below in this Syllabus (see Section 4).

ITINERARY WITH MANDATORY CLASS ATTENDANCE

If you choose this itinerary, you will be required to attend at least 85% of the classes.

ITINERARY WITHOUT MANDATORY CLASS ATTENDANCE

If you choose this itinerary, class attendance will not be mandatory, although it is recommended. However, even with this itinerary you will be required to submit deliverables and attend evaluation sessions on the dates indicated at the beginning of the course (see Section 4).

Students will be able to change itinerary with the authorization of the teacher; the teacher might set a time limit to changes of itinerary, allowing it only for a certain number of weeks at the beginning of the course. In this case, the time limit will be communicated in the first classes.

Independently of the itinerary, you will have to attend the evaluation sessions set for your itinerary.

1.11. Subject coordinator/s

Simone Santini

https://autoservicio.uam.es/paginas-blancas/

1.12. Competences and learning outcomes

1.12.1. Competences

B4 Conocimientos básicos sobre el uso y programación de los ordenadores, sistemas operativos, bases de datos y programas informáticos con aplicación en ingeniería.
C3 Capacidad para comprender la importancia de la negociación, los hábitos de trabajo efectivos, el liderazgo y las habilidades de comunicación en todos los entornos de desarrollo de software.
C4 Capacidad para elaborar el pliego de condiciones técnicas de una instalación informática que cumpla los estándares y normativas vigentes.
C5 Conocimiento, administración y mantenimiento sistemas, servicios y aplicaciones informáticas.
C6 Conocimiento y aplicación de los procedimientos algorítmicos básicos de las tecnologías informáticas para diseñar soluciones a problemas, analizando la idoneidad y complejidad de los algoritmos propuestos.
C7 Conocimiento, diseño y utilización de forma eficiente los tipos y estructuras de datos más adecuados a la resolución de un problema.
C14 Conocimiento y aplicación de los principios fundamentales y técnicas básicas de la programación paralela, concurrente, distribuida y de tiempo real.

1.12.2. Learning outcomes

After performing the course students will be able to create a complex program working as part of a team.

1.12.3. Course objectives

This course is part of the module "Programming and Data Structures" and relates to the subject "Programming". Its main objective is to provide students with the skills to create a complex program working as part of a team. For this, the student must use knowledge and skills acquired in the first year grades before or during the development of the subject.

This objective is closely related to following competence grade:

9. Ability to solve problems with initiative, decision making, autonomy and creativity. Ability to communicate and transmit knowledge, abilities and skills of the profession to the degree in Computer Engineering.

The **general objectives** to be achieved with this subject are:

G1. Work together and effectively to achieve a software product.

G2. Write the documentation of a software project.

G3. Design, implement and integrate the modules of a program.

G4. Verify and validate a program.

The **specific objectives** for each learning unit are:

Unit 1. Teamwork

1.1) Promote team commitment and cooperation.

1.2) Develop a reasonable plan time and resources.

1.3) Coordinate to achieve a common objective.

1.4) Integrate code from different authors.

Unit 2. Coding and Documentation

2.1) Programming according to a coding style guide.

2.2) Document the code.

2.3) Write technical documentation.

Unit 3. Testing

3.1) Design test cases.

3.2) Writing a test plan document.

3.3) Apply a test plan to a program.

Unit 4. Modules and libraries

4.1) Work in the program in a modular way

4.2) Locate and identify existing libraries.

4.3) Use third-party libraries.

Unit 5. Project

5.1) Implement a software project based on a requirements document and a partial preliminary design.

5.2) Integrate concepts and skills acquired in previous units which resulted in a complex computer program composed of several modules.

1.13. Course contents

The contents of this course are built around a programming project that will be implemented using the ANSI C programming language and the GNU development tools (make, gcc and gdb, mainly). Each student is free to choose their development environment, but the delivered project must have a Makefile developed by the students with an ASCII editor in order to allow to compile, link and deploy the programs handed out directly from command line.

The project development starts with a technical specification and a preliminary modular-design. Details of modules will be gradually specified in the content-units according to their goals. The subject contents are as follows:

Detailed Program

1. Introduction to the subject

1.1. Subject presentation: rules and topics list

1.2. Project presentation: requirements and iterative approach

2. Coding and Documentation

2.1. Using the GNU development environment: make, gcc and gdb

2.2. Coding style guides

2.3. Technical documentation with Doxygen

2.4. Modular programming concepts: modules and libraries

3. Introduction to verification and validation

3.1. Execution traces

3.2. Software Testing

4. Implementation and project management

4.1. Introduction to management and project planning

4.2. Completion of a project iteratively and incrementally following phases of design, implementation, debugging, integration and testing

1.14. Course bibliography

1. Kernighan, Brian W y Pike, Rob. **The Practice of Programming**. Addison-Wesley. ISBN: 020161586X. 1999. Cat: INF/C6110/KER (Texto completo en línea) (Disponible también en castellano)
2. Kernighan, Brian W. y Ritchie, Dennis M. **The C programming language**. Prentice Hall. ISBN: 0131103628. 1988. Cat: INF/C6140-C/KER (Texto completo en línea) (Disponible también en castellano)

2. Teaching-and-learning methodologies and student workload

2.2. List of training activities

		Number of Hours	Percentage
In-class work	Theoretical lectures	0h (0%)	54h (35%)
	Practical lectures	48h (22%)	
	Final test	6h (4%)	
Out-of-class work	Practical activities in the lab(**2.5hx14weeks**)	35h (23%)	96h (65%)
	Regulated individual work and study	30h (20%)	
	Unregulated individual work and study to prepare for the final test	31h (21%)	
Total work load in hours: 25 hours x 6 ECTS		**150h**	

3. Evaluation procedures and weight of components in the final grade

3.1. Regular assessment

The work of each student within this course is evaluated in terms of a set of deliverables, where each deliverable can be scored in terms of a collection of intermediate submissions. The deliverables are the result of the team work of a group of students. An individual test will determine the contribution of each student to the overall work of the team for each deliverable, as well as individual achievement achieved. Table "list of evaluation activities" displays the information both for those students who regularly attend the lectures (regular attendance) and for those students who do not (non regular attendance).

General considerations

1. The student must be present in each of the **evaluation sessions**, required to take the corresponding part into consideration, whether she follows the route with compulsory attendance or without it.
2. The detection of **plagiarism** in any of the activities developed during the course, either related to theory or practice, will penalize both the students who copy and the students who are copied. The copy penalty implies the application of the EPS internal regulations.

Deliverables

3. Deliverables will be evaluated following a **correction rubric** common to all groups, which will be published along with the statements.
4. Deliverables are graded from **0 to 10**, being considered as acceptable if the score is **equal or greater than 4,5**.
5. To pass the subject it is mandatory to present **all the deliverables** on time, according to the scheduling, and pass all of them.
6. The evaluation of each deliverable may involve an interview with each team that will serve to assess its realization, as well as the knowledge took into account for it. To evaluate the final iteration it could be required presentation of the final project in classroom.

Project

7. The description of the project will be published in Moodle at the beginning of the course. The **project requirements** will be common for all the teams, but, exceptionally, could be changed for the development of interdisciplinary projects in collaboration with other faculties and departments.

Final individual exam

8. Students can attend the final individual exam if an only if they successfully deliver the **final product**.
9. The exam will take up to **3 hours** approximately.

Team composition

10. Activities will take place in teams that may increase its size from the first iterations to the last.
11. The teams' composition will be established with students on the same itinerary at the beginning of each iteration, according to the established criteria at the beginning of the course.
12. Team **restructuring** will only be conducted under the authorization and supervision of the teacher. If a team's restructuration cannot be done due to a lack of available students (i.e., students with no team assignment), that team will disappear, and its members will be assigned to other teams.
13. The **final mark** of a student reassigned to a particular team will be computed with her marks in both the initial and final teams. In the final team a reassigned student could make use of code developed by her in the original team.

3.1.1. List of evaluation activities

Final score (FS) – regular attendance itinerary	**FS = Max (RFS ; NRFS)** where **RFS = 0.8*SD + 0.2*SIT** **NRFS = 0.5*SD + 0.5*SIT** where SD denotes an average of the scores corresponding to the deliverables associated to the project development iterations (as it will be specified at the beginning of the academic year), and SIT denotes the score of the individual test. In order to pass the course it is mandatory to obtain: a. SF >= 5 points b. SD >= 5 points c. SIT >= 4,5 points d. The score of each deliverable >= 4.5 points (The score for no submitted deliverables is 0 points. If a student does not attend an exam/test, then the related score is 0 points.) In other cases, the final score is computed as: **FS = 0.8*Min(4.5 ; SD) +0.2*Min(4.5 ; SIT)** If FS is equal to 0, the student will obtain the mark "No Evaluable".
Final score (FS) – non regular attendance itinerary	**FS = 0.5*SD+0.5*SIT** where SD denotes an average of the scores corresponding to the deliverables associated to the project development iterations (as it will be specified at the beginning of the academic year), and SIT is the score of the individual test. In order to pass the course it is mandatory to obtain: a. SF >= 5 points b. SD >= 5 points

	c. SIT >= 5 points d. The score of each deliverable >= 4.5 points (The score for no submitted deliverables is 0 points. If a student does not attend an exam/test, then the related score is 0 points.) In other cases the final score will be computed as: **FS: 0.5*Min(4.5 ; SD) +0.5*Min(4.5 ; SIT)** If FS is equal to 0, the student will obtain the mark “No Evaluable”.

3.2. Resit

In addition to those considerations for the regular assessment, students must take into account:

1. In order to take extraordinary exam, students must have submitted all required deliverables and have a score greater than or equal to 5 points in the calculation of SD.
2. Deliverables that have not been assessed before will be assessed the week before the individual test of the extraordinary examination.

3.2.1. List of evaluation activities

The same as regular assessment

4. Proposed workplan

The following schedule and some of the referred contents may slightly change during the course. Moreover, the submission deadlines for the pre-project and project deliverables are not specified. The corresponding date adjustments will be established at the beginning of the course and conveniently communicated in the classroom and via the Moodle platform.

Week	Tasks	Classroom working hours	Home working hours
1	Introduction to the course and the project to develop iteratively. Start to work in the project	3	5
2	Review of GNU development tools (gcc and make). Work in the project	3	5
3	Basics on programming style and technical documentation (Doxygen). Work in the project	3	5
4	Basics of work group (teamwork, collaborative tools and meetings). Work in the project	3	5
5	Modules and libraries. Work in the project	3	5
6	Execution traces.	3	5

Week	Tasks	Classroom working hours	Home working hours
	Work in the project		
7	Introduction to project planning and management. Work in the project	3	5
8	Introduction to software testing. Work in the project	3	5
9-14	Work in the project.	18	30
15	Final project review.	3	5
	Ordinary individual exam.	3	12
	Extraordinary individual exam.	3	12