MODULE DESCRIPTION FORM

<table>
<tr><td colspan="3">Module Information
معلومات المادة الدراسية</td></tr>
</table>

Module Title	**Programming Fundamentals II**	Module Delivery
Module Type	Core	☒ Theory
Module Code	**CyS1213**	☐ Lecture
ECTS Credits	7	☒ Lab
SWL (hr/sem)	**175**	☒ Tutorial ☒ Practical ☒ Seminar

Module Level	UGI	Semester of Delivery	2
Administering Department	Cybersecurity	**College**	College of Computer Science & Information Technology
Module Leader	Ali Kareem Abdul Raheem	**e-mail**	alialmujab@uowa.edu.iq
Module Leader's Acad. Title	Lecturer	**Module Leader's Qualification**	Ph.D.
Module Tutor	Ali Kareem Abdul Raheem	**e-mail**	alialmujab@uowa.edu.iq
Peer Reviewer Name	Nabil Sadiq Abdul abbas	**e-mail**	nabeel@uowa.edu.iq
Scientific Committee Approval Date	5-3-2026	**Version Number**	V1

<table>
<tr><td colspan="4">Relation with other Modules
العلاقة مع المواد الدراسية الأخرى</td></tr>
</table>

Prerequisite module	CYS1103	Semester	1
Co-requisites module	Non	Semester	…

Department Head Approval   **Dean of the College Approval**

	Module Aims, Learning Outcomes and Indicative Contents أهداف المادة الدراسية ونتائج التعلم والمحتويات الإرشادية
Module Objectives أهداف المادة الدراسية	The following are some key aims and benefits of studying Programming Fundamentals II: 1. Problem Solving: Teach students how to analyze problems and develop algorithms to solve them. Emphasize problem-solving techniques, algorithm design, and decomposition of complex problems into smaller, manageable parts. 2. Debugging and Testing: Teach students how to debug and test their programs to identify and fix errors. Explore techniques for error detection, debugging tools, and strategies for writing effective test cases. 3. Data Structures: Introduce students to fundamental data structures such as arrays, stacks, queues, structures. Explore their properties, implementation, and usage in solving programming problems. 4. Modular Programming: Introduce the concept of modular programming, including the use of functions, parameter passing, and code reuse. Emphasize the importance of modular design and writing reusable and maintainable code. 5. Programming Best Practices: Introduce students to programming best practices and coding standards, including code documentation, naming conventions, code formatting, and code optimization techniques. 6. Files Input and Output: Teach students how to interact with the user and handle input/output operations, including reading from and writing to files, standard input/output, and error handling. 7. Introduction to Object-Oriented Programming (OOP): Introduce the principles and concepts of OOP, including classes.
Module Learning Outcomes مخرجات التعلم للمادة الدراسية	The following are some common learning outcomes for an Programming Fundamentals II : 1. Effective Code Writing: Write clear, well-structured, and readable code that follows coding standards and best practices, including proper indentation, meaningful variable names, and appropriate comments. 2. Use of Data Structures: Apply appropriate data structures, such as arrays, to store and manipulate data effectively in programming problems. 3. Modular Design and Reusability: Design and implement modular programs by breaking them into reusable functions or methods, facilitating code reuse, improving maintainability, and promoting good software engineering practices. 4. Debugging and Testing Skills: Use debugging techniques and tools to identify and fix errors in programs. Develop effective test cases and perform testing to ensure program correctness and reliability. 5. Understand basic software development principles: Students should have an awareness of software development principles such as code organization, code reusability, and modularity. They should be able to write clear, readable, and maintainable code following coding conventions and best practices. 6. Collaborate effectively in programming projects: Students should demonstrate the ability to work collaboratively in a programming project,

	effectively communicating with team members, sharing code, and using version control systems.
Indicative Contents المحتويات الإرشادية	The indicative contents of an Programming Fundamentals II module have a list of common topics that shown below : 1- Modular Programming: **[25 hrs]** Functions and procedures, Scope and lifetime of variables, Parameter passing mechanisms. 2- Data Structures: **[25 hrs]** Arrays, Strings and lists, Structures, Stacks and queues. 3- Input and Output: **[15 hrs]** Standard input/output, Reading from and writing to files, Error handling and exception handling. 4- Debugging and Testing: Common types of programming errors, Debugging techniques and tools. **[20 hrs]**

Learning and Teaching Strategies استراتيجيات التعلم والتعليم	
Strategies	To teaching an Programming Fundamentals II module, various strategies can be employed to facilitate effective learning and engagement. Here are some learning and teaching strategies commonly used in Programming Fundamentals II module: 1- Lectures: Delivering lectures to present theoretical concepts, principles, and foundational knowledge of Programming Fundamentals II. Lectures can include visual aids, examples, and demonstrations to enhance understanding. 2- Interactive Discussions: Encourage students to actively participate in discussions by asking questions, sharing their thoughts, and engaging in peer-to-peer learning. Discussions can focus on challenging concepts, real-world applications, or case studies related to Programming Fundamentals II. 3- Hands-on Lab Sessions: Conduct practical lab sessions where students can gain hands-on experience with Programming Fundamentals II, commands, and programming exercises. These sessions provide an opportunity to reinforce theoretical concepts and develop practical skills. 4- Group Projects: Assign group projects that involve designing, implementing, and evaluating components of an Programming Fundamentals II. Group projects promote teamwork, problem-solving, and practical application of operating system concepts. 5- Online Resources and Tutorials: Provide access to online resources, tutorials, and interactive learning materials related to Programming

Fundamentals II. This allows students to explore additional content, reinforce their understanding, and self-assess their progress.

6- Assessments and Feedback: Use a variety of assessment methods such as quizzes, assignments, projects, and exams to evaluate students' understanding of Programming Fundamentals II concepts. Provide timely and constructive feedback to help students improve their knowledge and skills.

Student Workload (SWL)			
الحمل الدراسي للطالب محسوب لـ 15 اسبوعا			
Structured SWL (h/sem) الحمل الدراسي المنتظم للطالب خلال الفصل	78	Structured SWL (h/w) الحمل الدراسي المنتظم للطالب أسبوعيا	5
Unstructured SWL (h/sem) الحمل الدراسي غير المنتظم للطالب خلال الفصل	97	Unstructured SWL (h/w) الحمل الدراسي غير المنتظم للطالب أسبوعيا	6
Total SWL (h/sem) الحمل الدراسي الكلي للطالب خلال الفصل	175		

Module Evaluation					
التقييم التكويني للطالب					
		Time/Number	Weight (Marks)	Week Due	Relevant Learning Outcome
Formative assessment	Quizzes	5	10% (10)	2, 4, 5, 7, 9	1, 2, 3, 4
	Projects	1	10% (10)	10	All
	Lab execution	10	10% (10)	2, 3, 4, 5, 6, 7, 8, 9, 10, 11	All
	H. W.	5	5% (5)	2, 4, 5, 7, 9	1, 2, 3, 4
	Weekly Class Activity	5	5% (5)	2, 3, 4, 5, 6, 7, 8, 9, 10, 11	All
	Formative assessment	40%(40)			All
Summative assessment	Midterm Exam	2hr	10%(10)	7	All
	Final Exam	3hr	50% (50)	16	
Total assessment		100% (100 Marks)			

Delivery Plan (Weekly Syllabus)
المنهاج الاسبوعي النظري

	Material Covered
Week 1	Functions
Week 2	Function Types
Week 3	The concept of Recursion
Week 4	Array
Week 5	1D array
Week 6	2D array (Matrix)
Week 7	Array of Characters (Strings)
Week 8	String Processing
Week 9	Midterm Exam
Week 10	Arrays and functions
Week 11	Structures
Week 12	Array of structures and Nested Structures
Week 13	Stack and Queue
Week 14	Pointers
Week 15	Files
Week 16	Preparatory week before the final Exam

Delivery Plan (Weekly Lab. Syllabus)
المنهاج الاسبوعي للمختبر

	Material Covered
Week 1	Writing Codes using Functions
Week 2	Writing Codes using Function Types
Week 3	Writing Codes using The concept of Recursion
Week 4	Writing Codes using Arrays
Week 5	Writing Codes using 1D arrays
Week 6	Writing Codes using 2D array (Matrix)s
Week 7	Writing Codes using Array of Characters (Strings)
Week 8	Writing Codes using String Processing

Week 9	Midterm Exam
Week 10	Writing Codes using Arrays and functions
Week 11	Writing Codes using Structures
Week 12	Writing Codes using Array of structures and Nested Structures
Week 13	Writing Codes using Stack and Queue
Week 14	Pointers
Week 15	Files

Learning and Teaching Resources
مصادر التعلم والتدريس

	Text	Available in the Library?
Required Texts	C++: The Complete Reference, Fourth Edition, Herbert Schildt.	Yes
Recommended Texts	The C++ Programming Language , Third Edition , Bjarne Stroustrup.	Yes
Websites	https://stackoverflow.com/	

Grading Scheme
مخطط الدرجات

Group	Grade	التقدير	Marks %	Definition
Success Group (50 - 100)	**A - Excellent**	امتياز	90 - 100	Outstanding Performance
	B - Very Good	جيد جدا	80 - 89	Above average with some errors
	C - Good	جيد	70 - 79	Sound work with notable errors
	D - Satisfactory	متوسط	60 - 69	Fair but with major shortcomings
	E - Sufficient	مقبول	50 - 59	Work meets minimum criteria
Fail Group (0 – 49)	**FX – Fail**	راسب (قيد المعالجة)	(45-49)	More work required but credit awarded
	F – Fail	راسب	(0-44)	Considerable amount of work required

Note: Marks Decimal places above or below 0.5 will be rounded to the higher or lower full mark (for example a mark of 54.5 will be rounded to 55, whereas a mark of 54.4 will be rounded to 54. The University has a policy NOT to condone "near-pass fails" so the only adjustment to marks awarded by the original marker(s) will be the automatic rounding outlined above.