**Course Specification**

(**CS 313** Programming Language III)

|  |  |
| --- | --- |
| *University:* | Helwan University |
| *Faculty:* | Faculty of Computers & Information |
| *Department:* | ***Computer science*** |

**1. Course Data**

|  |  |
| --- | --- |
| **Code:** | **CS 313** |
| **Course title:** | **Programming Language III** |
| **Level:** | 2 |
| **Specialization:** | Computer Science |
| **Credit hours:** | 3 hours |
| **Number of learning units (hours):** | (3) theoretical 2) practical |

**2. Course Objective**

This course builds on the foundation provided by preliminary courses in introduction to computers and programming, to introduce the fundamental concepts of data structures and the algorithms that proceed from them. Topics include recursion, fundamental data structures (including stacks, queues, linked lists, hash tables, trees, and graphs), and the basics of algorithmic analysis. At the end of the semester, if time permits, and if a course in Object Oriented Programming (typically in C++) is at least going in parallel with this course in the same semester, the underlying philosophy of object-oriented programming can be applied to some data structure, e.g. stacks.

**3. Intended Learning Outcomes:**

**A- Knowledge and Understanding:**

A6. Describe the Modeling Problems.

A21. Apply the principles of Object-Oriented Programming.

* **Intellectual Skills**

B3. Develop Analytical Skills.

B5. Collaborate Modeling and Simulation.

B8. Gather and assess relevant information, using abstract ideas to

interpret it effectively.

B9. Design and implement Programming methods.

B24. Represent Data structures.

**C- Professional and Practical Skills**

C1. Choose the appropriate Programming Language.

C9. Design computer-based systems.

C18. Design of web pages based on the principles of human-computer

interactions.

**D- General and Transferable Skills**

D3. Use different Problem Solving techniques.

D5. Follow Creative Thinking.

D11. Clarify Ideas formulation and presentation.

D13. Practice Designing skills in software projects.

D14. Practice Engineering skills for software development.

**4. Course contents**

|  |  |  |  |
| --- | --- | --- | --- |
| **Topic** | **No. of hours** | **Lecture** | **Tutorial/ Practical** |
| Review of elementary programming concepts and C programming language. | 3 | 1 | Review of elementary programming concepts and C programming language. |
| **Software engineering:**  Software validation, testing fundamentals, including test plan creation and test case generation. | 3 | 1 | **Software engineering:**  Software validation, testing fundamentals, including test plan creation and test case generation. |
| Encapsulation and information hiding; separation of behavior and implementation; basic built-in data structures, one-, two-, and three-dimensional arrays. | 3 | 1 | Encapsulation and information hiding; separation of behavior and implementation; basic built-in data structures, one-, two-, and three-dimensional arrays. |
| **Fundamental data structures:**  Stacks; queues; linked lists; hash tables; trees; graphs. | 3 | 1 | **Fundamental data structures:**  Stacks; queues; linked lists; hash tables; trees; graphs. |
| **Recursion:**  The concept of recursion; recursive mathematical functions; simple recursive procedures; divide-and-conquer strategies; recursive backtracking; implementation of recursion; writing recursive algorithms in iterative fashion; when and when not to use recursion; | 6 | 2 | **Recursion:**  The concept of recursion; recursive mathematical functions; simple recursive procedures; divide-and-conquer strategies; recursive backtracking; implementation of recursion; writing recursive algorithms in iterative fashion; when and when not to use recursion; |
| **Basic algorithmic analysis**:  Asymptotic analysis of upper and average complexity bounds; identifying differences among best, average, and worst case behaviors; big “O,” little “o,” omega, and theta notation; standard complexity classes; empirical measurements of performance; time and space tradeoffs in algorithms; using recurrence relations to analyze recursive algorithms | 6 | 2 | **Basic algorithmic analysis**:  Asymptotic analysis of upper and average complexity bounds; identifying differences among best, average, and worst case behaviors; big “O,” little “o,” omega, and theta notation; standard complexity classes; empirical measurements of performance; time and space tradeoffs in algorithms; using recurrence relations to analyze recursive algorithms |
| **Fundamental computing algorithms**:  O(N log N) sorting algorithms; hash tables, including collision-avoidance strategies; binary search trees; representations of graphs; depth- and breadth-first traversals | 6 | 2 | **Fundamental computing algorithms**:  O(N log N) sorting algorithms; hash tables, including collision-avoidance strategies; binary search trees; representations of graphs; depth- and breadth-first traversals |
| **Object-oriented programming:**  Object-oriented design; encapsulation and information hiding; classes; separation of behavior and implementation; class hierarchies; inheritance. | 6 | 2 | **Object-oriented programming:**  Object-oriented design; encapsulation and information hiding; classes; separation of behavior and implementation; class hierarchies; inheritance. |
| **Algorithmic strategies**:  Brute-force algorithms; greedy algorithms; divide-and-conquer; backtracking; branch-and-bound; heuristics; pattern matching and string/text algorithms. | 6 | 2 | **Algorithmic strategies**:  Brute-force algorithms; greedy algorithms; divide-and-conquer; backtracking; branch-and-bound; heuristics; pattern matching and string/text algorithms. |
| Review of elementary programming concepts and C programming language. | 3 | 1 | Review of elementary programming concepts and C programming language. |
| **Software engineering:**  Software validation, testing fundamentals, including test plan creation and test case generation. | 3 | 1 | **Software engineering:**  Software validation, testing fundamentals, including test plan creation and test case generation. |
| Encapsulation and information hiding; separation of behavior and implementation; basic built-in data structures, one-, two-, and three-dimensional arrays. | 3 | 1 | Encapsulation and information hiding; separation of behavior and implementation; basic built-in data structures, one-, two-, and three-dimensional arrays. |
| **Fundamental data structures:**  Stacks; queues; linked lists; hash tables; trees; graphs. | 3 | 1 | **Fundamental data structures:**  Stacks; queues; linked lists; hash tables; trees; graphs. |

**Mapping contents to ILOs**

|  |  |  |  |  |
| --- | --- | --- | --- | --- |
| Topic | Intended Learning Outcomes (ILOs) | | | |
| Knowledge and understanding | Intellectual Skills | Professional and practical skills | General and Transferable skills |
| Review of elementary programming concepts and C programming language. | A6,A21 | B3 | C1,C9 | D3 |
| **Software engineering** | A6,A21 | B8 | C1,C18 | D5 |
| Encapsulation and information hiding; | A21 | B3,B8 | C1,C9 | D5,D3 |
| **Fundamental data structures** | A21 | B8 | C1,C18 | D3 |
| **Recursion** | A6 | B8 | C4 | D5 |
| **Basic algorithmic analysis** | A6 | B8,B9 | C9 | D5 |
| **Fundamental computing algorithms** |  | B24 | C18 | D3,D5 |
| **Object-oriented Programming** |  | B5,B5,B8 | C9 | D5,D14 |
| **Algorithmic strategies**. |  | B9,B24 | C1 | D11,D3,D5,D13 |

**5. Teaching and Learning Methods**

Class Lectures

Highly lab-based courses

Exercises

**6. Teaching and Learning Methods for students with limited capability**

Using data show

e-learning management tools

**7. Students Evaluation**

**a) Used Methods**

* Written exams to assess concepts related to Data Structures.
* Computer programs submitted in labs. (4 of them along the semester should be adequate).
* Written exercises solved in labs.

**b) Time**

Assessment 1: Test 1 Week 4

Assessment 2: Test 2 Week 7

Assessment 3: Midterm Exam Week 10

Assessment 4: Practical Exam Week 14

Assessment 5: final written exam Week 16

**c) Grades Distribution**

Mid-term Examination 20 %

Final-Year Examination 50 %

Semester Work 20 %

Practical Exam 10%

Total 100%

Any formative only assessments

**List of Books and References**

**a) Notes**

Course Notes

- Handouts

**b) Mandatory Books**

- Kruse, R.L., C.L. Tondo, and B.P. Leung, *Data structures and program design in C*. 2nd ed. 1997, Upper Saddle River, N.J.: Prentice Hall. xvi, 671 p.

**c) Suggested Books**

- Langsam, Y., M. Augenstein, and A.M. Tenenbaum, *Data structures using C and C++*. 2nd ed. 1996, Upper Saddle River, N.J.: Prentice Hall. xvi, 672 p.

- Sedgewick, R., *Algorithms in C*. 3rd ed. 1998, Reading, Mass.: Addison-Wesley. <v. 1-2 in 5 >.

- Dale, N.B., *C++ plus data structures*. 4th ed. 2006, Sudbury, MA: Jones and Bartlett Publishers.

**d) Other publications**

**Course Coordinator:**  Dr. Hala Abdel-Gelil

**Chairman of the Department:** Prof. dr. Iraqy Khalifa