Universidad Nacional Mayor de San Marcos
School of Computer Science
Syllabus of Course
Academic Period 2018-II

1. **Code and Name:** CS1100. Introduction to Computer Science (Mandatory)
2. **Credits:** 4
3. **Hours of theory and Lab:** 2 HT; 4 HL; (15 weeks)
4. **Professor(s)**

   Meetings after coordination with the professor
5. **Bibliography**

[Bro11] J. Glenn Brookshear. *Computer Science: An Overview*. Addison-Wesley, 2011.

[Gut13] John V Guttag. . *Introduction To Computation And Programming Using Python*. MIT Press, 2013.

[Zel10] John Zelle. *Python Programming: An Introduction to Computer Science*. Franklin, Beedle & Associates Inc, 2010.

6. **Information about the course**

   (a) **Brief description about the course** This is the first course in the sequence of introductory courses to Computer Science.This course is intended to cover the concepts outlined by the Computing Curricula IEEE-CS/ACM 2013. Programming is one of the pillars of Computer Science; any professional of the area, will need to program to materialize their models and proposals. This course introduces participants to the fundamental concepts of this art. Topics include data types, control structures, functions, lists, recursion, and the mechanics of execution, testing, and debugging.

   (b) **Prerrequisites:** None

   (c) **Type of Course:** Mandatory

   (d) **Modality:** Face to face

7. **Specific goals of the Course**

   - Introduce the fundamental concepts of programming.
   - Develop the ability of abstraction using programming language

8. **Contribution to Outcomes**

   a) An ability to apply knowledge of mathematics, science. (**Usage**)

   b) An ability to design and conduct experiments, as well as to analyze and interpret data. (**Usage**)

   a) An ability to apply knowledge of mathematics, science. (**Usage**)

   b) An ability to design and conduct experiments, as well as to analyze and interpret data. (**Usage**)

9. **Competences (IEEE)**

   C1. An intellectual understanding and the ability to apply mathematical foundations and computer science theory.⇒ **Outcome a**

   C2. Ability to have a critical and creative perspective in identifying and solving problems using computational thinking. ⇒ **Outcome b**

   C1. An intellectual understanding and the ability to apply mathematical foundations and computer science theory.⇒ **Outcome a**

C2. Ability to have a critical and creative perspective in identifying and solving problems using computational thinking.
⇒ **Outcome b**

10. List of topics

1. History
2. Basic Type Systems
3. Fundamental Programming Concepts
4. Basic Analysis
5. Fundamental Data Structures and Algorithms
6. Algorithms and Design
7. Development Methods

11. Methodology and Evaluation
Methodology:

Theory Sessions:
The theory sessions are held in master classes with activities including active learning and roleplay to allow students to internalize the concepts.

Lab Sessions:
In order to verify their competences, several activities including active learning and roleplay will be developed during lab sessions.

Oral Presentations:
Individual and team participation is encouraged to present their ideas, motivating them with additional points in the different stages of the course evaluation.

Reading:
Throughout the course different readings are provided, which are evaluated. The average of the notes in the readings is considered as the mark of a qualified practice. The use of the UTEC Online virtual campus allows each student to access the course information, and interact outside the classroom with the teacher and with the other students.
Evaluation System:

12. Content

<table>
<tr><td colspan="2">Unit 1: History (5)</td></tr>
<tr><td colspan="2">Competences Expected: C4</td></tr>
<tr><td>Learning Outcomes</td><td>Topics</td></tr>
<tr><td>• Identify significant continuing trends in the history of the computing field [Familiarity]
• Identify the contributions of several pioneers in the computing field [Familiarity]
• Discuss the historical context for several programming language paradigms [Familiarity]
• Compare daily life before and after the advent of personal computers and the Internet [Assessment]</td><td>• Prehistory, the world before 1946
• History of computer hardware, software, networking
• Pioneers of computing
• History of the Internet</td></tr>
<tr><td colspan="2">Readings : [Bro11], [Gut13], [Zel10]</td></tr>
</table>

Unit 2: Basic Type Systems (2)	
Competences Expected: C1	
Learning Outcomes	**Topics**
• For both a primitive and a compound type, informally describe the values that have that type [Familiarity] • For a language with a static type system, describe the operations that are forbidden statically, such as passing the wrong type of value to a function or method [Familiarity] • Describe examples of program errors detected by a type system [Familiarity] • For multiple programming languages, identify program properties checked statically and program properties checked dynamically [Usage] • Use types and type-error messages to write and debug programs [Usage] • Define and use program pieces (such as functions, classes, methods) that use generic types, including for collections [Usage]	• A type as a set of values together with a set of operations – Primitive types (e.g., numbers, Booleans) – Compound types built from other types (e.g., records, unions, arrays, lists, functions, references) • Association of types to variables, arguments, results, and fields • Type safety and errors caused by using values inconsistently given their intended types
Readings : [Gut13], [Zel10]	

Unit 3: Fundamental Programming Concepts (9)

Competences Expected: C1

Learning Outcomes	Topics
• Analyze and explain the behavior of simple programs involving the fundamental programming constructs variables, expressions, assignments, I/O, control constructs, functions, parameter passing, and recursion. [Assessment] • Identify and describe uses of primitive data types [Familiarity] • Write programs that use primitive data types [Usage] • Modify and expand short programs that use standard conditional and iterative control structures and functions [Usage] • Design, implement, test, and debug a program that uses each of the following fundamental programming constructs: basic computation, simple I/O, standard conditional and iterative structures, the definition of functions, and parameter passing [Usage] • Write a program that uses file I/O to provide persistence across multiple executions [Usage] • Choose appropriate conditional and iteration constructs for a given programming task [Familiarity] • Describe the concept of recursion and give examples of its use [Assessment] • Identify the base case and the general case of a recursively-defined problem [Familiarity]	• Basic syntax and semantics of a higher-level language • Variables and primitive data types (e.g., numbers, characters, Booleans) • Expressions and assingments • Simple I/O including file I/O • Conditional and iterative control structures • Functions and parameter passing • The concept of recursion

Readings : [Gut13], [Zel10]

Unit 4: Basic Analysis (2)

Competences Expected: C1,C5

Learning Outcomes	Topics
• Explain what is meant by "best", "expected", and "worst" case behavior of an algorithm [Familiarity] • In the context of specific algorithms, identify the characteristics of data and/or other conditions or assumptions that lead to different behaviors [Familiarity] • State the formal definition of big O [Familiarity] • Use big O notation formally to give asymptotic upper bounds on time and space complexity of algorithms [Usage] • Use big O notation formally to give expected case bounds on time complexity of algorithms [Usage]	• Differences among best, expected, and worst case behaviors of an algorithm • Big O notation: formal definition • Complexity classes, such as constant, logarithmic, linear, quadratic, and exponential • Big O notation: use • Analysis of iterative and recursive algorithms

Readings : [Gut13], [Zel10]

Unit 5: Fundamental Data Structures and Algorithms (8)

Competences Expected: C1,C2,C5

Learning Outcomes	Topics

• Implement basic numerical algorithms [Usage]
• Implement simple search algorithms and explain the differences in their time complexities [Assessment]
• Be able to implement common quadratic and O(N log N) sorting algorithms [Usage]
• Describe the implementation of hash tables, including collision avoidance and resolution [Familiarity]
• Discuss the runtime and memory efficiency of principal algorithms for sorting, searching, and hashing [Familiarity]
• Discuss factors other than computational efficiency that influence the choice of algorithms, such as programming time, maintainability, and the use of application-specific patterns in the input data [Familiarity]
• Explain how tree balance affects the efficiency of various binary search tree operations [Familiarity]
• Solve problems using fundamental graph algorithms, including depth-first and breadth-first search [Usage]
• Demonstrate the ability to evaluate algorithms, to select from a range of possible options, to provide justification for that selection, and to implement the algorithm in a particular context [Assessment]
• Describe the heap property and the use of heaps as an implementation of priority queues [Familiarity]
• Solve problems using graph algorithms, including single-source and all-pairs shortest paths, and at least one minimum spanning tree algorithm [Usage]
• Trace and/or implement a string-matching algorithm [Usage]
	• Simple numerical algorithms, such as computing the average of a list of numbers, finding the min, max,
	• Sequential and binary search algorithms
	• Worst case quadratic sorting algorithms (selection, insertion)
	• Worst or average case O(N log N) sorting algorithms (quicksort, heapsort, mergesort)
	• Hash tables, including strategies for avoiding and resolving collisions
	• Binary search trees
	  – Common operations on binary search trees such as select min, max, insert, delete, iterate over tree
	• Graphs and graph algorithms
	  – Representations of graphs (e.g., adjacency list, adjacency matrix)
	  – Depth- and breadth-first traversals
	• Heaps
	• Graphs and graph algorithms
	  – Shortest-path algorithms (Dijkstra's and Floyd's algorithms)
	  – Minimum spanning tree (Prim's and Kruskal's algorithms)
	• Pattern matching and string/text algorithms (e.g., substring matching, regular expression matching, longest common subsequence algorithms)

Readings : [Gut13], [Zel10]

Unit 6: Algorithms and Design (9)

Competences Expected: C1,C2,C5

Learning Outcomes	Topics
• Discuss the importance of algorithms in the problem-solving process [Familiarity] • Discuss how a problem may be solved by multiple algorithms, each with different properties [Familiarity] • Create algorithms for solving simple problems [Usage] • Use a programming language to implement, test, and debug algorithms for solving simple problems [Usage] • Implement, test, and debug simple recursive functions and procedures [Usage] • Determine whether a recursive or iterative solution is most appropriate for a problem [Assessment] • Implement a divide-and-conquer algorithm for solving a problem [Usage] • Apply the techniques of decomposition to break a program into smaller pieces [Usage] • Identify the data components and behaviors of multiple abstract data types [Usage] • Implement a coherent abstract data type, with loose coupling between components and behaviors [Usage] • Identify the relative strengths and weaknesses among multiple designs or implementations for a problem [Assessment]	• The concept and properties of algorithms – Informal comparison of algorithm efficiency (e.g., operation counts) • The role of algorithms in the problem-solving process • Problem-solving strategies – Iterative and recursive mathematical functions – Iterative and recursive traversal of data structures – Divide-and-conquer strategies • Fundamental design concepts and principles – Abstraction – Program decomposition – Encapsulation and information hiding – Separation of behaivor and implementation

Readings : [Gut13], [Zel10]

Unit 7: Development Methods (1)

Competences Expected: C2

Learning Outcomes	Topics
• Construct and debug programs using the standard libraries available with a chosen programming language [Familiarity]	• Modern programming enviroments – Code search – Programming using library components and their APIs

Readings : [Gut13], [Zel10]