Birla Institute of Technology and Science, Pilani, Hyderabad Campus
Department of Computer Sc. and Information Systems
Second Semester 2020-2021, Course Handout (Part-II)
CS F111 (Computer Programming)

Date: 25th July 2023

In addition to Part-I (General Handout for all courses appended to the time table) this portion gives further specific details regarding the course:

Course Number	: CS F111
Course Title	: Computer Programming
Instructor-In-Charge	: Narasimha Bolloju (narsi.bolloju@hyderabad.birs-pilani.ac.in)
Instructors	: Aneesh Sreevallabh, Pragati Shrivastava, Jay Kamlesh Dave, Apurba Das, Prajna Devi Upadhyay, Aruna Malapati

Scope and Objective of the Course:
This is an introductory course to computers and programming in ‘C’. This course uses a problem-solving focused approach to teach computer programming. The course starts with an introduction to computational thinking which is applicable to solving problems across disciplines. The course elaborates on the process of creating or developing algorithms and/or flowcharts for solving different types of problems through computer programming. The course covers programming constructs in C including data types, variables, operators, input/output, decision making, loops, arrays, functions, structures, dynamic memory allocations, file handling. Students will get hands on experience through homework, and programming exercises in the laboratory. Towards the end of the course, students will be introduced to the concepts of programming at large and object-oriented programming.

Text Book:
T1: Problem solving and program design in C, 7th edition, Jeri Hanly and Elliot Koffman, Pearson, 2014.

Reference Books:
R1: The C Programming Language, Kernighan and Ritchie, 2nd Edition, Pearson, 2015.
R2: Programming in ANSI C, E Balaguruswamy, Mc Graw Hill, 8th Edition 2019.
R3: Let us C, Yaswanth Kanethkar, BPB Publications, 16th Edition, 2017.

Lecture Plan:

Week #	Learning objectives	Topics (Reference)
1	• understand and practice writing instructions and to get familiar with C programming language • develop problem-solving skills and ability to approach complex problems in a systematic and algorithmic manner • recognize the core components of computational thinking	**Introduction to programming**: Writing instructions, Computational thinking (decomposition, pattern recognition, abstraction, and algorithm design), Introduction to C, C language elements (lecture notes, ch 1, ch 2)
2	• become familiar with the general form of a C program and the basic elements in a program including the differences between the data types int, double, and char • understand how to write assignment statements to change the values of variables • learn how C evaluates arithmetic expressions and how to write them in C • learn how to read data values into a program and to display results	**Overview of C:** program structure, identifiers, variable declarations and data types, assignment statement, arithmetic expressions, formatting numbers (ch 2)
3	• learn about functions and how to use them to write programs with separate modules • use flowcharts as a system documentation tool • understand how control flows between function main and other functions and pass information to functions using input arguments • understand the concepts of function prototypes and scope.	**Top-down design with functions:** library functions, functions with and without arguments, function prototypes and scope (ch 3)
4	• understand the purpose of conditional statements in decision-making. • learn how to write if-else statements and nested if-else structures. • use comparison operators (e.g., ==, >, <, >=, <=, !=) in conditionals. • apply logical operators (e.g., &&, ||, !) for complex conditions.	**Selection structures**: Control structures, conditions, comparison operators, logical operators, if statement, nested if statements, switch statement (ch 4)
5	• understand the concept of loops and their importance in repetitive tasks. • learn to write for, while, and do-while loops and know when to use each. • master the use of loop control statements (e.g., break, continue). • avoid common pitfalls such as infinite loops.	**Repetition and Loops:** repetition in programs, counting loops, while statement, for loop, conditional loops, nested loops, do-while statement (ch 5)
6	• understand the concept of pointers and their role in memory manipulation. • learn how to declare, initialize, and dereference pointers. • understand the relationship between pointers and arrays. • use pointers for dynamic memory allocation and deallocation.	**Pointers and modular programming:** pointers and the indirect operator, functions with output parameters, scope of names (ch 6)
7	• understand the concept of arrays and their use for storing multiple elements of the same type. • learn to declare, initialize, and access elements in arrays. • Work with multidimensional arrays and strings in C. • handle common array operations like sorting and searching.	**Arrays:** declaring and referencing arrays, array subscripts, loops for sequential access, searching and sorting arrays, parallel arrays and enumerated types, multi-dimensional arrays (ch 7)

8	• understand the concept of strings and their use for storing and manipulating text • work with strings in C and associated functions. • handle common string operations like matching, and manipulation	**Strings:** string basics, string library functions, string comparisons, arrays of points, character operations, conversions (ch 8)
9	• understanding the concept of recursion and its applications. Write simple recursive functions for basic problems (e.g., factorial, Fibonacci sequence). • learn to identify recursive patterns in problems. • apply the divide-and-conquer approach to break complex problems into simpler sub-problems.	**Recursion:** the nature of recursion, tracing a recursive function, recursive math functions, problem solving with recursion (ch 9)
10	• understand the concept of structures and unions in C. • learn to declare and define structures and unions. • access members of structures and unions using dot (.) and arrow (->) operators. • comprehend the differences between structures and unions.	**Structure and union types:** user-defined data types, structure data type, problem solving with structure types, parallel arrays, arrays of structures, union types (ch 10)
11	• learn about streams in C and their relationship to files and standard input and output devices • understand how scanf, fscanf and printf, fprintf are used to read and write characters to text files • learn about binary files and understand the differences between binary and text files	**Text and binary processing:** input and output files, binary files, search a database, differences between binary and text files (ch 11)
12	• learn to use dynamic memory allocation functions (e.g., malloc, calloc, free). • define what dynamic data structures are and their significance in programming. • learn the concepts of stacks, queues and trees as dynamic data structures with specific access patterns	**Dynamic data structures:** pointers, dynamic memory allocation, linked lists, stacks, queues, binary trees (ch 13)
13	• learn how procedural and data abstraction helps the programmer • learn how to create your own personal library with a separate header file and implementation file and to understand what should be stored in each file • learn the core principles of OOP: Encapsulation, Inheritance, Polymorphism, and Abstraction. • Understand the benefits of using OOP in terms of code organization, reusability, and maintainability.	**Programming at large:** using abstraction to manage complexity, personal libraries, arguments to main function (ch 12); **Object-oriented programming** : core principles and benefits of OOP, introduction to C++ (ch 15)

Evaluation:

Component	**Duration**	**Weightage (%)**	**Date & Time**	**Nature of Component**
Class participation and homework	NA	20%	Throughout the semester	Open book
Lab work	NA	20%	Throughout the semester	Open book
Mid-semester test	1.5 Hrs.	25%	10/10 - 4.00 - 5.30PM	Closed book
Comprehensive exam	3 Hrs.	35%	11/12 AN	Closed book

Make-up-Policy:
Make-up will be strictly granted on prior permissions and on justifiable grounds only for the Mid-Semester test and Comprehensive exam. No make-up will be available for other evaluation components.

Course Notices:
All notices pertaining to this course will be posted on the Google Classroom.

Chamber Consultation Hour:
Will be announced on the Google Classroom.

Academic Honesty and Integrity Policy:
Academic honesty and integrity are to be maintained by all the students throughout the semester and no type of academic dishonesty is acceptable.

Instructor-In-Charge
CS F111