**Course Contents and Lecture Schedule**

| Lesson No/Session No | Topics | No. of hours |
| --- | --- | --- |
| **UNIT - I** | | |
| **Algorithms and Flowcharts** | | |
|  | The meaning of algorithms, flowcharts and their need | 55 |
|  | Writing algorithms | 55 |
|  | Drawing flowcharts | 55 |
|  | Writing algorithms and drawing flowcharts for simple exercises | 55 |
| **Constants, Variables and Data Types** | | |
|  | Character set, C tokens, keywords, Identifiers, structure of C program, Writing a C Program | 55 |
|  | Constants, variables, data types, Declaration of variables, declaration of storage classes, Assigning values to variables ,defining symbolic constants, declaring a variable as constant, declaring a variable as volatile, overflow and underflow of data | 55 |
| **Operators and Expressions** | | |
|  | Arithmetic operators, relational operators, logical operators, assignment operator | 55 |
|  | Increment and decrement operator, conditional operator, bitwise operators, comma  operator, special operators | 55 |
|  | Arithmetic expressions, evaluation of expressions, precedence of arithmetic operators,  Type conversions in expressions, operator precedence and associativity, mathematical functions | 55 |
| **UNIT - II** | | |
| **Managing Input and Output Operations** | | |
|  | The *scanf()* function for input operations, *printf()* function for output operations | 55 |
|  | Reading a character, writing a character (the *getchar() & putchar()* functions), The  address operator(*&*), formatted input using format specifiers | 55 |
|  | formatted output using format specifiers, Writing simple complete C programs | 55 |
| **Decision Making and Branching** | | |
|  | Decision making with *if* statement, simple if statement, the *if..else* statement,  programming examples | 55 |
|  | Nesting of *if..else* statements, the *else..if* ladder, programming examples | 55 |
|  | The *switch* statement, the ***?*:** operator, the *goto* statement, the break statement,  programming examples | 55 |
| **Decision Making and Looping** | | |
|  | While statement, the do..while statement, programming examples | 55 |
|  | The for statement, nested loops, jumps in loops, the continue statement, programming examples | 55 |
| **UNIT - III** | | |
| **Arrays** | | |
|  | The meaning of an array, one dimensional arrays declaration, initialization, reading , writing and manipulation, programming examples | 55 |
|  | Two dimensional arrays, declaration, initialization, reading , writing and manipulation, programming examples | 55 |
|  | More Examples |  |
|  | Multidimensional arrays, dynamic arrays, programming examples | 55 |
| **Strings** | | |
|  | Declaring and initializing string variables, reading string from terminal, writing string to  screen, programming examples | 55 |
|  | Arithmetic operations on characters, putting strings together, comparison of two strings,  programming examples | 55 |
|  | String handling functions, table of strings, other features of strings, programming  Examples | 55 |
| **UNIT - IV** | | |
| **User Defined Functions** | | |
|  | Need for user defined functions, a multi function program, elements of User defined  functions, defining functions | 55 |
|  | Return values and their types, function calls, function declaration, category of functions,  no arguments and no return values, programming examples | 55 |
|  | Arguments but no return values, arguments with return values, no arguments with return  value, functions that return multiple values, programming examples | 55 |
|  | Nesting of functions, recursion, passing arrays to functions, passing string to functions, programming examples | 55 |
|  | The scope, visibility and life time of variables, programming examples | 55 |
| **Structures and Unions** | | |
|  | Defining a structure, declaring structure variables, accessing structure members, structure  initialization, programming examples | 55 |
|  | copying and comparing structure variables, operations on individual members,  programming examples | 55 |
|  | array of structures, structures within structures, structures and functions, programming  examples | 55 |
|  | Unions, programming examples | 55 |
| **UNIT - V** | | |
| **Pointers** | | |
|  | Understanding pointers, accessing the address space of a variable, declaring and  initialization pointer variables, accessing a variable through its pointer | 55 |
|  | Chain of pointers, pointer expressions, pointers and arrays, pointer and character strings,  array of pointers, programming examples | 55 |
|  | Pointer as function arguments, functions returning pointers, pointers to functions, pointers  and structures, programming examples | 55 |
| **Dynamic Memory Allocation** | | |
|  | Dynamic memory allocation, allocating a block of memory: m*alloc,* allocating multiple  blocks of memory: *calloc,* programming examples | 55 |
|  | Releasing the used space: *Free*, altering the size of a block: r*ealloc*, programming  Examples | 55 |
| **File Management in C and Additional features** | | |
|  | Defining and opening a file, closing a file, input/output operations on files, programming  examples  Macro substitution, files inclusion, compiler control directives, ANSI additions, programming exercises | 55 |
|  | error handling during I/O operations, random access files, command line arguments,  programming examples | 55 |
|  | Macro substitution, files inclusion, programming exercises | 55 |
|  | Compiler control directives, ANSI additions, programming exercises | 55 |