



# KG REDDY

College of Engineering  
& Technology

New Age Engineering

## COURSE FILE

**Subject Name** : Programming for Problem Solving  
**Faculty Name** : Ms. Ashwini  
**Designation** : Assistant Professor  
**Regulation /** : R18  
**Course Code** : CS103ES  
**Year / Semester** : I / II  
**Department** : Computer Science and Engineering  
**Academic Year** : 2018-19

Signature of the Faculty

PRINCIPAL

KG Reddy College of Engineering & Technology  
Chilkur(V), Moinabad (M),  
R.R. Dist. Telangana.

## **1. Vision, Mission, Program Educational Objectives (PEOs),**

### **Program Outcomes (POs), Program Specific Outcomes (PSOs)**

#### **VISION**

To be recognized as a department of excellence by stimulating a learning environment in which students and faculty will thrive and grow to achieve their professional, institutional and societal goals.

#### **MISSION**

- .To provide high quality technical education to students that will enable life-long learning and build expertise in advanced technologies in Computer Science and Engineering.
- To promote research and development by providing opportunities to solve complex engineering problems in collaboration with industry and government agencies.
- To encourage professional development of students that will inculcate ethical values and leadership skills while working with the community to address societal issues.

## **PROGRAM EDUCATIONAL OUTCOMES(PEOs)**

**PEO 1:** Graduates will provide solutions to difficult and challenging issues in their profession by applying computer science and engineering theory and principles.

**PEO 2:** Graduates have successful careers in computer science and engineering fields or will be able to successfully pursue advanced degrees.

**PEO 3:** Graduates will communicate effectively, work collaboratively and exhibit high levels of professionalism, moral and ethical responsibility.

**PEO 4:** To develop the ability to understand and analyze engineering issues in a broader perspective with ethical responsibility towards sustainable development.

## **PROGRAMME OBJECTIVES (POs):**

**PO I: Engineering knowledge:** Engineering knowledge: Apply the knowledge of mathematics, science, engineering fundamentals and an engineering specialization to the solution of complex engineering problems.

**PO II: Problem analysis:** Identify, formulate, review research literature, and analyze complex engineering problems reaching substantiated conclusions using first principles of mathematics, natural sciences, and engineering sciences.

**PO III: Design/development of solutions:** Design solutions for complex engineering problems and design system components or processes that meet the specified needs with appropriate consideration for the public health and safety, and the cultural, societal, and environmental considerations.

**PO IV: Conduct investigations of complex problems:** Use research-based knowledge and research methods including design of experiments, analysis and interpretation of data, and synthesis of the information to provide valid conclusions.

**PO V: Modern tool usage:** Create, select, and apply appropriate techniques, resources, and modern engineering and IT tools including prediction and modeling to complex engineering activities with an understanding of the limitations.

**PO VI: The engineer and society:** Apply reasoning informed by the contextual knowledge to assess societal, health, safety, legal and cultural issues and the consequent responsibilities relevant to the professional engineering practice.

**PO VII: Environment and sustainability:** Understand the impact of the professional engineering solutions in the societal and environmental contexts, and demonstrate the knowledge of, and need for sustainable development.

**PO VIII: Ethics:** Apply ethical principles and commit to professional ethics and responsibilities and norms of the engineering practice.

**PO IX: Individual and team work:** Function effectively as an individual, and as a member or leader in diverse teams, and in multidisciplinary settings.

**PO X: Communication:** Communicate effectively on complex engineering activities with the engineering community and with society at large, such as, being able to comprehend and write effective reports and design documentation, make effective presentations, and give and receive clear instructions.

**PO XI: Project management and finance:** Demonstrate knowledge and understanding of the engineering and management principles and apply these to one's own work, as a member and leader in a team, to manage projects and in multidisciplinary environments.

**PO XII: Life-long learning:** Recognize the need for, and have the preparation and ability to engage in independent and life-long learning in the broadest context of technological change.

## **PROGRAMME SPECIFIC OUTCOMES (PSOs)**

### **PROGRAMME SPECIFIC OUTCOMES (PSO's)**

**PSO 1: Problem Solving Skills** – Graduate will be able to apply computational techniques and software principles to solve complex engineering problems pertaining to software engineering.

**PSO 2: Professional Skills** – Graduate will be able to think critically, communicate effectively, and collaborate in teams through participation in co and extra-curricular activities.

**PSO 3: Successful Career** – Graduates will possess a solid foundation in computer science and engineering that will enable them to grow in their profession and pursue lifelong learning through post-graduation and professional development.

## **2. Syllabus (University copy)**

**CS103ES/CS203ES: PROGRAMMING FOR PROBLEM SOLVING**

**B.Tech. I Year I Sem.**

	L	T	P	C
	3	1	0	4

**Course Objectives:**

- To learn the fundamentals of computers.
- To understand the various steps in program development.
- To learn the syntax and semantics of C programming language.
- To learn the usage of structured programming approach in solving problems.

**Course Outcomes:** The student will learn

- To write algorithms and to draw flowcharts for solving problems.
- To convert the algorithms/flowcharts to C programs.
- To code and test a given logic in C programming language.
- To decompose a problem into functions and to develop modular reusable code.
- To use arrays, pointers, strings and structures to write C programs.
- Searching and sorting problems.

**Unit - 1: Introduction to Programming**

Introduction to components of a computer system: disks, primary and secondary memory, processor, operating system, compilers, creating, compiling and executing a program etc.,

Number systems

Introduction to Algorithms: steps to solve logical and numerical problems. Representation of Algorithm, Flowchart/Pseudo code with examples, Program design and structured programming

Introduction to C Programming Language: variables (with data types and space requirements), Syntax and Logical Errors in compilation, object and executable code , Operators, expressions and precedence, Expression evaluation, Storage classes (auto, extern, static and register), type conversion, The main method and command line arguments

Bitwise operations: Bitwise AND, OR, XOR and NOT operators

Conditional Branching and Loops: Writing and evaluation of conditionals and consequent branching with if, if-else, switch-case, ternary operator, goto, Iteration with for, while, dowhile loops

I/O: Simple input and output with scanf and printf, formatted I/O, Introduction to stdin, stdout and stderr.

Command line arguments

**Unit - II: Arrays, Strings, Structures and Pointers:**

Arrays: one and two dimensional arrays, creating, accessing and manipulating elements of arrays

Strings: Introduction to strings, handling strings as array of characters, basic string functions available in C (strlen, strcat, strcpy, strstr etc.), arrays of strings

Structures: Defining structures, initializing structures, unions, Array of structures

Pointers: Idea of pointers, Defining pointers, Pointers to Arrays and Structures, Use of

Pointers in self-referential structures, usage of self referential structures in linked list (noimplementation) Enumeration data type

### **Unit - III: Preprocessor and File handling in C:**

Preprocessor: Commonly used Preprocessor commands like include, define, undef, if, ifdef, ifndef

Files: Text and Binary files, Creating and Reading and writing text and binary files, Appending data to existing files, Writing and reading structures using binary files, Random access using fseek, ftell and rewind functions.

### **Unit - IV: Function and Dynamic Memory Allocation:**

Functions: Designing structured programs, Declaring a function, Signature of a function, Parameters and return type of a function, passing parameters to functions, call by value, Passing arrays to functions, passing pointers to functions, idea of call by reference, Some C standard functions and libraries

Recursion: Simple programs, such as Finding Factorial, Fibonacci series etc., Limitations of Recursive functions

Dynamic memory allocation: Allocating and freeing memory, Allocating memory for arrays of different data types

### **Unit - V: Introduction to Algorithms:**

Algorithms for finding roots of a quadratic equations, finding minimum and maximum numbers of a given set, finding if a number is prime number, etc.

Basic searching in an array of elements (linear and binary search techniques),

Basic algorithms to sort array of elements (Bubble, Insertion and Selection sort algorithms),

Basic concept of order of complexity through the example programs

### **TEXT BOOKS:**

1. Byron Gottfried, Schaum's Outline of Programming with C, McGraw-Hill
2. B.A. Forouzan and R.F. Gilberg C Programming and Data Structures, Cengage Learning, (3<sup>rd</sup> Edition)

### **REFERENCE BOOKS:**

1. Brian W. Kernighan and Dennis M. Ritchie, The C Programming Language, Prentice Hall of India
2. R.G. Dromey, How to solve it by Computer, Pearson (16<sup>th</sup> Impression)
3. Programming in C, Stephen G. Kochan, Fourth Edition, Pearson Education.
4. Herbert Schildt, C: The Complete Reference, Mc Graw Hill, 4<sup>th</sup> Edition



### **3. COURSE OBJECTIVES, COURSE OUTCOMES AND TOPIC OUTCOMES**

#### **3.COURSE OBJECTIVES**

1. To learn the fundamentals of computers.
2. To understand the various steps in program development.
3. To learn the syntax and semantics of C programming language.
4. To learn the usage of structured programming approach in solving problems.

#### **Course Outcomes**

- CO1 Sketh the flowchart and write algorithm for solving problems.
- CO2 Translate the algorithms/flowcharts to C programs
- CO3 Test for a Code to given a logic in C programming language.
- CO4 Use arrays, pointers, strings and structures to write C programs and Searching and sorting problems.

**TOPIC OUTCOMES:**

S.NO	TOPIC	TOPIC OUTCOME
( At the end of the topic, the student will be able to)		
	<b>UNIT I</b>	
1	Introduction to components of a computer system: disks, primary and secondary memory, processor, operating system.	Understand the components of computer
2	Compilers, creating, compiling and executing a program etc.	Identify the compiling and executing process
3	Number systems.	Analyze the number system.
4	Introduction to Algorithms: steps to solve logical and numerical problems.	Understand the steps of algorithm.
5	Representation of Algorithm.	Understand the representation
6	Flowchart/Pseudo code with examples.	Draw the flowchart.
7	Program design and structured programming.	Learn what structure programming is.
8	Introduction to C Programming Language: variables.	Understand the basics of programming.
9	Syntax and Logical Errors in compilation, object and executable code.	Learn the syntax of language.
10	Operators, expressions and precedence, Expression evaluation.	Understand the operators. Evaluate the expression.
11	Storage classes (auto, extern, static and register), type conversion, The main method and command line arguments.	Student will be able to understand different storage classes like extern, auto, and static.
12	Bitwise operations: Bitwise AND, OR, XOR and NOT operators.	Describe the Bitwise AND, OR, XOR and NOT operators.
13	Conditional Branching and Loops: Writing and evaluation of conditionals and consequent branching with if, if-else, switch-case, ternary operator, goto.	Analyze the different conditionals and consequent branching with if, if-else, switch-case, ternary operator, goto.
14	Iteration with for, while, dowhile loops I/O: Simple input and output with scanf and printf.	Analyze different looping statement.
15	Formatted I/O, Introduction to stdin, stdout and stderr. Command line arguments.	Learn formatted i/o with command line argument.
	<b>UNIT II</b>	
16	Arrays: one and two dimensional arrays, creating, accessing	Learn array concept.

	and manipulating elements of arrays.	
17	Strings: Introduction to strings, handling strings as array of characters.	Understand string.
18	Basic string functions available in C (strlen, strcat, strcpy, strstr etc.).	Learn string function.
19	Structures: Defining structures, initializing structures.	Define the structure.
20	Unions.	Learn union.
21	Array of structures.	Understand array of structure
22	Pointers: Idea of pointers, Defining pointers.	Define pointers.
23	Pointers to Arrays and Structures	Demonstrate Pointers to Arrays and Structures.
24	Use of Pointers in self-referential structures	Use of Pointers in self-referential structures.
25	Usage of self-referential structures in linked list	Implement self-referential structures in linked list.
26	Enumeration data type	Understand Enumeration data type.
27	Arrays of strings.	Learn Arrays of strings.
	<b>UNIT III</b>	
28	Pre-processor: Commonly used Pre-processor commands like include,	Understand Commonly used Preprocessor commands like include
29	Define, undef, if,	Learn different pre-processor command.
30	Ifdef, ifndef	Learn different pre-processor command.
31	Files: text	Understand the file.
32	Binary files	Understand binary files.
33	Creating and Reading and writing text	Write text.
34	Binary files	Understand Binary files.
35	Appending data to existing files	Learn how to append data to existing file.
36	Writing and reading structures using binary files	Learn how to write & read structures using binary files
37	Random access using fseek	Use fseek function.
38	Ftell and rewind functions.	Understand the Ftell and rewind functions
	<b>UNIT IV</b>	
39	Functions: Designing structured programs, Declaring a function, Signature of a function.	Understand how to declare function.
40	Parameters and return type of a function.	Learn Parameters and return type of a function.
41	Passing parameters to functions.	Learn how to pass parameter to a function.
42	Call by value.	Differentiate between call by value and call by

		reference.
43	Passing arrays to functions.	Describe how to pass array to function.
44	Passing pointers to functions.	Learn how to Pass pointers to functions.
45	Idea of call by reference.	Differentiate between call by value and call by reference.
46	Some C standard functions and libraries.	Understand Some C standard functions and libraries.
47	Recursion: Simple programs.	Construct simple program.
48	Finding Factorial, Fibonacci series etc.	Write an program for Finding Factorial, Fibonacci series etc.
49	Limitations of Recursive functions.	Know the Limitations of Recursive functions.
50	Dynamic memory allocation: Allocating and freeing memory.	Learn Allocating and freeing memory function.
51	Allocating memory for arrays of different data types.	Learn Allocating memory for arrays of different data types.
<b>UNIT V</b>		
52	Algorithms for finding roots of a quadratic equations	Write an algorithm for finding roots of a quadratic equations.
53	Finding minimum and maximum numbers of a given set	Write an algorithm Finding minimum and maximum numbers of a given set.
54	Finding if a number is prime number	Write an algorithm Finding if a number is prime number.
55	Basic searching in an array of elements using linear search method.	Know the linear search method.
56	Basic searching in an array of elements using binary search method.	Learn the binary search method.
57	Basic algorithms to sort array of elements using bubble sort algorithms.	Learn bubble sort algorithms.
58	Basic algorithms to sort array of elements using selection sort algorithms.	Learn selection sort algorithms.
59	Basic algorithms to sort array of elements using insertion sort algorithms.	Learn insertion sort algorithms.
60	Basic concept of order of complexity through the example programs	Understand concept of order of complexity
61	<b>Topics Beyond Syllabi:</b> Implementation of Stacks and Queues using array and pointers	Able to write programs in data structures
62	<b>Topics Beyond Syllabi:</b> Software Development Process	Understand concepts of Software Development Process
63	<b>Gaps in the syllabus</b> Merge sort	Understand concepts of Merge Sort
64	<b>Gaps in the syllabus</b> Software Development Process	Understand concepts of Software Development Process

#### **4. COURSE PRE-REQUISITES**

- Knowledge of Components of Computer.
- Intelligence to complex problem-solving

## 5.COURSE INFORMATION SHEET (CIS)

### A. Course Description:

PROGRAMME: B. Tech. (Computer Science Engineering.)	DEGREE: B-TECH
COURSE: PROGRAMMING FOR PROBLEM SOLVING	YEAR: I SEM: II CREDITS: 4
COURSE CODE: CS103ES/CS203ES REGULATION: R18	COURSE TYPE: CORE
COURSE AREA/DOMAIN: PROGRAMMING	CONTACT HOURS: 3 (L+T)) hours/Week.
CORRESPONDING LAB COURSE CODE (IF ANY): CS108ES/CS208ES	LAB COURSE NAME: PROGRAMMING FOR PROBLEM SOLVING LAB

### B. Syllabus:

Unit	Details	Hours
I	<p><b>Introduction to components of a computer system:</b> disks, primary and secondary memory, processor, operating system, compilers, creating, compiling and executing a program etc., Number systems</p> <p><b>Introduction to Algorithms:</b> steps to solve logical and numerical problems. Representation of Algorithm, Flowchart/Pseudo code with examples, Program design and structured programming</p> <p><b>Introduction to C Programming Language:</b> variables (with data types and space requirements), Syntax and Logical Errors in compilation, object and executable code, Operators, expressions and precedence, Expression evaluation, Storage classes (auto, extern, static and register), type conversion, The main method and command line arguments</p> <p><b>Bitwise operations:</b> Bitwise AND, OR, XOR and NOT operators</p> <p><b>Conditional Branching and Loops:</b> Writing and evaluation of conditionals and consequent branching with if, if-else, switch-case, ternary operator, goto, Iteration with for, while, do-while loops</p> <p><b>I/O:</b> Simple input and output with scanf and printf, formatted I/O, Introduction to stdin, stdout and stderr. Command line arguments.</p>	24
II	<p><b>Arrays:</b> one and two dimensional arrays, creating, accessing and manipulating elements of arrays</p> <p><b>Strings:</b> Introduction to strings, handling strings as array of characters, basic string functions available in C (strlen, strcat, strcpy, strstr etc.), arrays of strings</p> <p><b>Structures:</b> Defining structures, initializing structures, unions, Array of structures</p> <p><b>Pointers:</b> Idea of pointers, Defining pointers, Pointers to Arrays and Structures, Use of Pointers in self-referential structures, usage of self-referential structures in linked list (no implementation), Enumeration data type</p>	10
III	<p><b>Preprocessor:</b> Commonly used Preprocessor commands like include, define, undef, if, ifdef, ifndef</p> <p><b>Files:</b> Text and Binary files, Creating and Reading and writing text and binary files, Appending data to existing files, Writing and reading structures using binary files, Random access using fseek, ftell and rewind functions.</p>	10
IV	<p><b>Functions:</b> Designing structured programs, Declaring a function, Signature of a function,</p>	10

	Parameters and return type of a function, passing parameters to functions, call by value, Passing arrays to functions, passing pointers to functions, idea of call by reference, Some C standard functions and libraries. <b>Recursion:</b> Simple programs, such as Finding Factorial, Fibonacci series etc., Limitations of Recursive functions <b>Dynamic memory allocation:</b> Allocating and freeing memory, Allocating memory for arrays of different data types.	
V	Algorithms for finding roots of quadratic equations, finding minimum and maximum numbers of a given set, finding if a number is prime number, etc. Basic searching in an array of elements (linear and binary search techniques), Basic algorithms to sort array of elements (Bubble, Insertion and Selection sort algorithms), Basic concept of order of complexity through the example programs	9
<b>Contact classes for syllabus coverage</b>		<b>63</b>
<b>Lectures beyond syllabus</b>		<b>2</b>
<b>Tutorial classes</b>		<b>0</b>
<b>Classes for gaps &amp; Add-on classes</b>		<b>2</b>
<b>Total No. of classes</b>		<b>67</b>

### C. GAPS IN THE SYLLABUS - TO MEET INDUSTRY/PROFESSION REQUIREMENTS:

SNO	DESCRIPTION	PROPOSED ACTIONS	PO
1	Software Development Process	Guest lecturer	2,3,5

### D. TOPICS BEYOND SYLLABUS / ADVANCED TOPICS:

SNO	Topic	PO
1	Software Development Process	2,3,5
2	Merge sort	2,3,5

### E. WEB SOURCE REFERENCES:

1	<a href="http://nptel.ac.in/video.php?subjectId=106105085">http://nptel.ac.in/video.php?subjectId=106105085</a>
2	<a href="http://freevidelectures.com/Course/2519/C-Programming-and-Data-Structures">http://freevidelectures.com/Course/2519/C-Programming-and-Data-Structures</a>
3	<a href="http://www.cosmolearning.com/courses/programming-and-data-structure-543/video-lectures/">http://www.cosmolearning.com/courses/programming-and-data-structure-543/video-lectures/</a>

4	<a href="https://www.cs.auckland.ac.nz/~jmor159/PLDS210/ds_ToC.html">https://www.cs.auckland.ac.nz/~jmor159/PLDS210/ds_ToC.html</a>
---	---

#### F. DELIVERY / INSTRUCTIONAL METHODOLOGIES:

<input checked="" type="checkbox"/> CHALK & TALK	<input checked="" type="checkbox"/> STUD. ASSIGNMENT	<input checked="" type="checkbox"/> WEB RESOURCES
<input checked="" type="checkbox"/> LCD/SMART BOARDS	<input checked="" type="checkbox"/> STUD. SEMINARS	<input type="checkbox"/> ADD-ON COURSES

#### G. ASSESSMENT METHODOLOGIES – DIRECT

<input checked="" type="checkbox"/> ASSIGNMENTS	<input checked="" type="checkbox"/> STUD. SEMINARS	<input checked="" type="checkbox"/> TESTS/MODEL EXAMS	<input checked="" type="checkbox"/> UNIV. EXAMINATION
<input checked="" type="checkbox"/> STUD.LAB PRACTICES	<input checked="" type="checkbox"/> STUD. VIVA	<input type="checkbox"/> MINI/MAJOR PROJECTS	<input type="checkbox"/> CERTIFICATIONS
<input type="checkbox"/> ADD-ON COURSES	<input type="checkbox"/> OTHERS		

#### H. ASSESSMENT METHODOLOGIES - INDIRECT

<input checked="" type="checkbox"/> ASSESSMENT OF COURSE OUTCOMES(BY FEEDBACK, ONCE)	<input checked="" type="checkbox"/> STUDENT FEEDBACK ON FACULTY (TWICE)
<input type="checkbox"/> ASSESSMENT OF MINI/MAJOR PROJECTS BY EXT. EXPERTS	<input type="checkbox"/> OTHERS

#### I. TEXT / REFERENCE BOOKS:

T/R	BOOK TITLE/AUTHORS/PUBLICATION
Text Book	Byron Gottfried, Schaum's Outline of Programming with C, McGraw-Hill
Text Book	B.A. Forouzan and R.F. Gilberg C Programming and Data Structures, Cengage Learning, (3rd Edition)
Reference Book	Brian W. Kernighan and Dennis M. Ritchie, The C Programming Language, Prentice Hall of India
Reference Book	R.G. Dromey, How to solve it by Computer, Pearson (16th Impression)
Reference Book	Programming in C, Stephen G. Kochan, Fourth Edition, Pearson Education.



Reference Book	Herbert Schildt, C: The Complete Reference, Mc Graw Hill, 4th Edition
----------------	---

## 6. Micro Lesson Plan

S.NO	TOPIC	SCHEDULED DATE	ACTUAL DATE
<b>UNIT -1 : INTRODUCTION TO PROGRAMMING</b>			
1	Introduction to components of a computer system: disks, primary and secondary memory, processor, operating system.	20-12-18	21/12/18
2	Compilers, creating, compiling and executing a program etc.	21/12/18	22/12/18
3	Number systems.	22/12/18	26/2/18
4	Introduction to Algorithms: steps to solve logical and numerical problems.	26/12/18	27/2/18
5	Representation of Algorithm.	27/12/18	2/01/19
6	Flowchart/Pseudo code with examples.	29/12/18	3/01/19
7	Program design and structured programming.	2/01/19	5/01/19
8	Introduction to C Programming Language: variables.	3/01/19	07/01/19
9	Syntax and Logical Errors in compilation, object and executable code.	4/01/19	09/01/19
10	Operators, expressions and precedence, Expression evaluation.	5/01/19	10/01/19
11	Storage classes (auto, extern, static and register), type conversion, The main method and command line arguments.	07/01/19	10/01/19
12	Bitwise operations: Bitwise AND, OR, XOR and NOT operators.	09/01/19	16/01/19
13	Conditional Branching and Loops: Writing and evaluation of conditionals and consequent branching with if, if-else, switch-case, ternary operator, goto.	10/01/19	17/01/19
14	Iteration with for, while, dowhile loops I/O: Simple input and output with scanf and printf.	16/01/19	21/01/19
15	Formatted I/O, Introduction to stdin, stdout and stderr. Command line arguments.	17/01/19	21/01/19
<b>UNIT -2 : ARRAYS, STRINGS, STRUCTURES AND POINTERS</b>			
16	Arrays: one and two dimensional arrays, creating, accessing and manipulating elements of arrays.	19/01/19	22/01/19
17	Strings: Introduction to strings, handling strings as array of characters.	21/01/19	23/01/19
18	Basic string functions available in C (strlen, strcat, strcpy, strstr etc.).	22/01/19	24/01/19
19	Structures: Defining structures, initializing structures.	23/01/19	28/01/19
20	Unions.	24/01/19	28/01/19

21	Array of structures.	28/01/19	30/01/19
22	Pointers: Idea of pointers, Defining pointers.	30/01/19	31/01/19
23	Pointers to Arrays and Structures	31/01/19	4/02/19
24	Use of Pointers in self-referential structures	01/02/19	5/02/19
25	Usage of self-referential structures in linked list	04/02/19	6/02/19
26	Enumeration data type	05/02/19	8/02/19
27	Arrays of strings.		
<b>UNIT – 3 : PREPROCESSOR AND FILE HANDLING IN C:</b>			
28	Preprocessor: Commonly used Preprocessor commands like include,	06/02/19	11/02/19
29	Define, undef, if,	08/02/19	11/02/19
30	Ifdef, ifndef	11/02/19	13/02/19
31	Files: text	12/02/19	14/02/19
32	Binary files	13/02/19	15/02/19
33	Creating and Reading and writing text	14/02/19	19/02/19
34	Binary files	15/02/19	21/02/19
35	Appending data to existing files	19/02/19	22/02/19
36	Writing and reading structures using binary files	21/02/19	22/02/19
37	Random access using fseek	22/02/19	25/02/19
38	Ftell and rewind functions.		
<b>UNIT – 4 :FUNCTION AND DYNAMIC MEMORY ALLOCATION:</b>			
39	Functions: Designing structured programs, Declaring a function, Signature of a function.	22/02/19	26/02/19
40	Parameters and return type of a function.	25/02/19	4/03/19
41	Passing parameters to functions.	26/02/19	6/03/19
42	Call by value.	04/03/19	08/03/19
43	Passing arrays to functions.	05/03/19	08/03/19
44	Passing pointers to functions.	06/03/19	13/03/19
45	Idea of call by reference.	08/03/19	14/03/19
46	Some C standard functions and libraries.	11/03/19	18/03/19
47	Recursion: Simple programs.	13/03/19	18/03/19
48	Finding Factorial, Fibonacci series etc.	14/03/19	20/03/19
49	Limitations of Recursive functions.	16/03/19	20/03/19
50	Dynamic memory allocation: Allocating and freeing memory.	18/03/19	21/03/19
51	Allocating memory for arrays of different data types.		
<b>UNIT – 5 : INTRODUCTION TO ALGORITHMS:</b>			
52	Algorithms for finding roots of a quadratic equations	20/03/19	25/03/19
53	Finding minimum and maximum numbers of a given set	21/03/19	25/03/19
54	Finding if a number is prime number	25/03/19	26/03/19
55	Basic searching in an array of elements using linear search method.	26/03/19	26/03/19
56	Basic searching in an array of elements using binary search method.	28/03/19	28/03/19

57	Basic algorithms to sort array of elements using bubble sort algorithms.		
58	Basic algorithms to sort array of elements using selection sort algorithms.		
59	Basic algorithms to sort array of elements using insertion sort algorithms.		
60	Basic concept of order of complexity through the example programs		
<b>Topics Beyond Syllabi:</b>			
61	Software Development Process		
62	Merge sort		
<b>Gaps in the syllabus</b>			
63	Software Development Process		

## 7. TEACHING SCHEDULE



<b>Subject</b>	<b>Programming for Problem Solving</b>					
<b>Text Books (to be purchased by the Students)</b>						
<b>Book 1</b>	Byron Gottfried, Schaum's Outline of Programming with C, McGraw-Hill					
<b>Book 2</b>	B.A. Forouzan and R.F. Gilberg C Programming and Data Structures, Cengage Learning, (3rd Edition)					
<b>Reference Books</b>						
<b>Book 3</b>	Programming in C, Stephen G. Kochan, Fourth Edition, Pearson Education.					
<b>Book 4</b>	Herbert Schildt, C: The Complete Reference, Mc Graw Hill, 4th Edition					
<b>Unit</b>	<b>Topic</b>	<b>Chapters Nos</b>				<b>No of classes</b>
		<b>Book 1</b>	<b>Book 2</b>	<b>Book 3</b>	<b>Book 4</b>	
<b>I</b>	Introduction to Programming	Ch1	Ch1			6
		Ch1				5
		Ch1	Ch2	Ch1	Ch2	5
		Ch3		Ch2	Ch12	5
		Ch4	Ch7	Ch7	Ch16	5
<b>II</b>	Arrays, Strings, Structures and Pointers:	Ch11		Ch10	Ch11	2
		Ch7	Ch4	Ch4	Ch8	2
		Ch12	Ch7	Ch7	Ch16	2
		Ch10	Ch9	Ch5	Ch11	2
		Ch14	Ch12			2
<b>III</b>	Preprocessor and File handling in C	Ch10	Ch10		Ch17	10
<b>IV</b>	Function and Dynamic Memory Allocation:	Ch9	Ch8	Ch5	Ch7	10
<b>V</b>	Introduction to Algorithms	Ch14				10
<b>Contact classes for syllabus coverage</b>						<b>63</b>
<b>Gaps in the Syllabus</b>						<b>2</b>
<b>Topics beyond the syllabus</b>						<b>2</b>
<b>Total No. of Classes</b>						<b>67</b>

**Department of Computer Science and Engineering**

**CO-PO Attainment**

Academic Year: 2018-2019

I B.Tech Semester-II

Section: CSE-B

Course Code: CS203ES

Course Name: Programming for Problem Solving

Course Instructor: Ashwini

**CO-PO & PSO Mapping:**

	PO1	PO2	PO3	PO4	PO5	PO6	PO7	PO8	PO9	PO10	PO11	PO12	PSO1	PSO2	PSO3
CO1	3	2			3			1				1	2		2
CO2	3												2		2
CO3	3				2								2		2
CO4	3							1					2		2

Attainment Levels: H: Substantial (High) M: Moderate (Medium) L: Slight (Low)

**Course Outcomes:** Students will be able to

- CO1 Write algorithms and to draw flowcharts for solving problems
- CO2 Convert the algorithms/flowcharts to C programs
- CO3 Code and test a given logic in C programming language.
- CO4 Use arrays, pointers, strings and structures to write C programs, searching and sorting problems.

K1-Remembering, K2-Understanding, K3-Appling, K4-Analyzing, K5-Evaluating, and K6-Creating

**Program Outcomes(POs)**

PO1	<b>Engineering knowledge:</b> Apply the knowledge of mathematics, science, engineering fundamentals and an engineering specialization to the solution of complex engineering problems.
PO2	<b>Problem analysis:</b> Identify, formulate, review research literature, and analyze complex engineering problems reaching substantiated conclusions using first principles of mathematics, natural science and engineering sciences.
PO3	<b>Design/development of solutions:</b> Design solutions for complex engineering problems and design system components or processes that meet the specified needs with appropriate consideration for the public health and safety, and the cultural, societal and environmental considerations.
PO4	<b>Conduct investigations of complex problems:</b> Use research based knowledge and research methods including design of experiments, analysis and interpretation of data, and synthesis of the information to provide valid conclusions.
PO5	<b>Modern tool usage:</b> Create, select and apply appropriate techniques, resources and modern engineering and IT tools including prediction and modeling to complex engineering activities with an understanding of the limitations.
PO6	<b>The engineer and society:</b> Apply reasoning informed by the contextual knowledge to assess societal, health, safety, legal and cultural issues and the consequent responsibilities relevant to the professional engineering practice.
PO7	<b>Environment sustainability:</b> Understand the impact of the professional engineering solutions in the societal and environmental contexts, and demonstrate the knowledge of, and need for sustainable development.
PO8	<b>Ethics:</b> Apply ethical principles and commit to professional ethics and responsibilities and norms of the engineering practice.
PO9	<b>Individual and team work:</b> Function effectively as an individual and as a member or leader in diverse teams, and in multidisciplinary settings.
PO10	<b>Communication:</b> Communicate effectively on complex engineering activities with the engineering community and with society at large, such as, being able to comprehend and write effective reports and design documentation, make effective presentations, and give and receive clear instructions.
PO11	<b>Project management and finance:</b> Demonstrate knowledge and understanding of the engineering and management principles and apply these to one's own work, as a member and leader in a team, to manage projects and in multidisciplinary environments.
PO12	<b>Lifelong learning:</b> Recognize the need for, and have the preparation and ability to engage in independent and lifelong learning in the broader context of technological change.

**Program Specific Outcomes (PSOs)**


PSO1	Problem Solving Skills– Graduate will be able to apply computational techniques and software principles to solve complex engineering problems pertaining to software engineering.
PSO2	Professional Skills– Graduate will be able to think critically, communicate effectively, and collaborate in teams through participation in co and extra-curricular activities.

PSO3

Graduates will possess a solid foundation in Electronics and Communications engineering that will enable them to grow in their profession and pursue lifelong learning through post-graduation and professional development.

  
Course Coordinator

  
Module Coordinator

  
Program Coordinator / HOD  
Head of the Department  
Humanities & Science  
K.G. Reddy College of Engg. & Tech.  
Chilukur, Moinabad, R.R. Dist. T.S.

[illegible]





[illegible]



Department of Computer Science and Engineering  
Overall CO Attainment

Academic Year: 2018-2019  
Course Code: CS203ES  
Course Instructor: Ashwini

I B.Tech Semester-II  
Section: CSE-B  
Course Name: Programming for Problem Solving

Overall Mid Attainment				
COs	Mid Examination-I Attainment %	Mid Examination-II Attainment %	Total Mid Attainment %	Attained Level
CO1	89.67		89.67	3
CO2	81.89		81.89	3
CO3		86.28	86.28	3
CO4		79.88	79.88	3
Overall Mid Attainment			84.43	3

Attainment Level	Threshold Value
H	3
M	2
L	1

Action Taken Report	
COs	Action Taken
CO2	

  
Course Coordinator

  
Module Coordinator

  
Program Coordinator / HOD  
Head of the Department  
Humanities & Science  
K.G. Reddy College of Engg. & Tech.  
Chilkur, Moinabad, R.R. Dist. T.S.

**CO Attainment for Semester End Examination**

**I B.Tech II Semester**

Academic Year: 2018-19

**Course Code: CS203ES**

**Course Name: Programming for Problem Solving**

Name of the Faculty: Ashwini

**Course Outcomes:** Students will be able to

- (C01) Write algorithms and to draw flowcharts for solving problems
- (C02) Convert the algorithms/flowcharts to C programs
- (C03) Code and test a given logic in C programming language.
- (C04) Use arrays, pointers, strings and structures to write C programs, searching and sorting problems.

Rationale:		
1	18QM1A0521	0
2	18QM1A0561	3
3	18QM1A0562	6
4	18QM1A0563	0
5	18QM1A0564	0
6	18QM1A0565	0
7	18QM1A0566	0
8	18QM1A0567	6
9	18QM1A0568	6
10	18QM1A0569	0
11	18QM1A0570	6
12	18QM1A0571	0
13	18QM1A0572	6
14	18QM1A0573	5
15	18QM1A0574	5
16	18QM1A0575	5
17	18QM1A0576	0
18	18QM1A0577	3
19	18QM1A0578	0
20	18QM1A0579	6
21	18QM1A0580	0
22	18QM1A0581	7
23	18QM1A0582	5
24	18QM1A0583	0
25	18QM1A0584	0
26	18QM1A0585	0
27	18QM1A0586	5
28	18QM1A0587	0
29	18QM1A0588	5
30	18QM1A0589	0
31	18QM1A0590	0
32	18QM1A0591	5
33	18QM1A0592	0
34	18QM1A0593	5
35	18QM1A0594	5
36	18QM1A0595	6
37	18QM1A0596	6
38	18QM1A0597	5
39	18QM1A0598	0
40	18QM1A0599	5
41	18QM1A05A0	0
42	18QM1A05A1	5
43	18QM1A05A2	6
44	18QM1A05A3	5
45	18QM1A05A4	5
46	18QM1A05A5	0
47	18QM1A05A6	5
48	18QM1A05A7	0
49	18QM1A05A8	6
50	18QM1A05A9	0
51	18QM1A05B0	7
52	18QM1A05B1	5
53	18QM1A05B2	5
54	18QM1A05B3	0
55	18QM1A05B4	5
56	18QM1A05B5	0
57	18QM1A05B6	0
58	18QM1A05B7	6
59	18QM1A05B8	0
60	18QM1A05B9	7
61	18QM1A05C0	5

Attainment Level	Threshold Value
H - 3	61% of students got $\geq 40\%$ marks.
M - 2	51% of students got $\geq 40\%$ marks.
L - 1	41% of student got $\geq 40\%$ of marks

Gap Analysis					
	Achieved Attainment %	Target Attainment %	Target in Level	Attainment in Level	Gap= Target in Level-Attainment in Level
	59.02	61.00	3	2	1

COs	Action Taken
CO1, CO2, CO3, CO4	Remedial class

Overall Attainment				
COs	Total Mid Examination Attainment %	Semester End Examination Attainment %	Total Attainment %	Attained Level
CO1	89.67	59.02	68.22	3
CO2	81.89	59.02	65.88	3
CO3	86.28	59.02	67.20	3
CO4	79.88	59.02	65.28	3
Average Attainment	84.43	59.02		3

Overall Attainment %	84.43
Overall Attained Level	3

Attainment Level	Threshold Value
H	3 61% of student got >=40% of marks
M	2 51% of student got >=40% of marks
L	1 41% of student got >=40% of marks

Academic Year: 2018-2019

I B. Tech Semester: II  
Course Name: Programming for Problem Solving  
Course Code: CS203ES

**CO-PO Attainment**

Section-CSE-B

		PO1		PO2		PO3		PO4		PO5		PO6		PO7		PO8		PO9		PO10		PO11		PO12		PS	
		Planned	Attained	Planned	Attained	Planned	Attained	Planned	Attained	Planned	Attained	Planned	Attained	Planned	Attained	Planned	Attained	Planned	Attained	Planned	Attained	Planned	Attained	Planned	Attained		
COs attainment %		3	2.05	2	1.36					3	2.05					1	0.68							1	0.68	2	
CO1 68.22		3	2.05	2	1.36					3	2.05					1	0.68							1	0.68	2	
CO2 65.88		3	1.98																							2	
CO3 67.20		3	2.02							2	1.34															2	
CO4 65.28		3	1.96													1	0.65									2	
Total		12	7.96	6	3.96					8	5.39					2	1.33							2	1.33	8	
Attainment %			66.64		66.22						67.81						66.75								68.22		
Attained Level			H		H						H						H								H		
Average		3.00	2.60	2.00	1.94					1.25	1.85					0.50	0.33							0.25	0.17	2.00	
Attainment Level																											

Attainment Level	Threshold Value
H	61% of students got >=40% marks.
M	51% of students got >=40% marks.
L	41% of student got >=40% of marks

# Unit 1

## 1. Introduction to Computer

### 1.1 Computer Systems

A computer is an electronic device which performs operations such as as an input, store the data, manipulate or process the data & produce the result as an output.

Main task performed by a computer

- Accept the data
- Process or manipulate the data
- Display or store the result in the form of human understanding
- Store the data, instructions and results.

### 1.2 Computing Environments

→ The arrangement of computer devices to solve a problem is said to be computing environment.

→ Computing environment is a collection of computers which are used to process & exchange the information to solve various types of computing problems.

## Types of Environment.

### ① Personal Computing Environment:

It is a stand alone machine. In this, the complete program resides on a stand alone machine & executed from the same machine.

e.g. laptop  
mobile.  
Printer

### ② Time Sharing Computing Environment.

Time sharing computing environment is a stand alone computer in which a single user can perform multiple operations at a time by using multitasking operating systems.

Here the processor time is divided among different tasks & this is called "Time sharing".

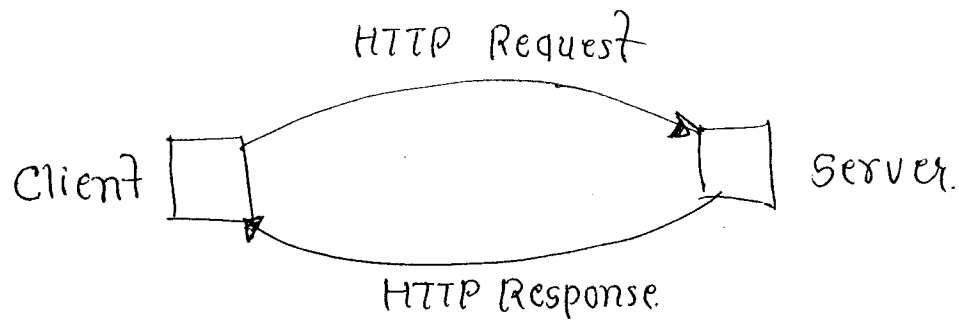
e.g. User can listen to music while writing something in a text editor.

### ③ Client Server Computing Environment.

It contains two machines. These both machines will exchange the information through an application.

In this environment client request for data & server provides data to the client.





#### ④ Distributed Computing

In the distributed computing environment

The complete functionality of a system is not on single computer but it is distributed among multiple computers.

In distributed computing environment the data is distributed among different systems and that data is logically related to each other.

#### ⑤ Grid Computing environment:

Grid computing is a collection of computers from different locations. All these computers work for a common problem.

A grid can be described as distributed collection of large number of computers working for a single application.

## Cluster Computing Environment

cluster computing is a collection of inter connected computers. These computer works together as a single system.

### 1.3 Computer languages:

languages are means of communication. Normally people interact with each other through a language. On the same pattern, communication with computer is carried out through a language. This language is understood by both user & the machine.

Computer language means an artificial language used to write instructions that can be translated into machine language & then executed by a computer.

languages are classified into two

- ① low level language.
- ② High level language.

low level language:

low level languages are designed to operate & handle the entire h/w & instructions set architecture of a computer directly, low-level languages are considered to be closer to computers.

They are two types of low-level languages.

- ① Machine language
- ② Assembly language.

① Machine language:

This language understood by the computer. It is machine dependent. It is difficult to learn & more difficult to write programs.

0101012001

② Assembly language:

The language where the machine code comprising of 0's & 1's are substituted by symbolic codes to improve their understanding.

It is the 1<sup>st</sup> step to improve programming structure.

It is also machine dependent language. Programmer must have a knowledge of the machine on which program will run. where instructions are given in English like word as ADD, SUM, MOV etc.

e.g.

add  $t_1, t_2, t_3$

04CB: 0000 0106 1100 1011

High level language:

High level language are computer independent & programming become quite easy & simple.

① Basic

② COBOL

③ FORTRAN

④ C.

⑤ C++

High level language not understood by the machine. So it need to translate by the translator into machine level. A translator is a SW which is used to translate high.

high level language as well as low level language into machine level language.

Three types of translator are there.

- 1) Compiler
- 2) Interpreter
- 3). Assembler.

## ~ Program development life cycle

When we want to develop a program using any programming language, we follow a sequence of steps. These steps are called phases in program development.

The phases are given below.

### ① Problem Definition.

Define the problem statement & decide the boundaries of the problem.

### 2) Problem Analysis :

Determine the requirement like variables, functions, etc. to solve the problem.

### ③ Algorithm Development

During this phase we develop a step by step procedure to solve the problem using the specification given in the previous phase.

### ④ Coding & Documentation:

construct actual program

### ⑤ Testing & Debugging:

During this phase, we check whether the code written in previous step is solving the specified problem or not.

### ⑥ Maintenance:

Program is actively used by the others. If users encounter any problem or wants any enhancements, then we need to repeat all the phases from the starting.

## 1.5 Algorithm & flowchart

Algorithm: An algorithm is defined as a finite set of steps that provide a chain of actions for solving a problem.

Categories of Algorithm:

① Sequence :

2) selection

3) Iteration.

① Sequence: steps described in an algorithm are performed successively one by one without skipping any step.

② Selection:

In algorithm there must be the procedure to handle operation failure occurring during execution.

• The selection of statements can be shown as follows.

if (condition)

statement-1

else

statement 2.

Iteration:

when we have to perform same action for a number of time. then iteration is useful.

e.g of algorithm.

write an algorithm to add two numbers entered by the user.

step1: start

step2: Declare variable num1, num2, & sum

step3: Read value num1 & num2.

step4: Add num1 & num2 & assign the result to sum.

$sum \leftarrow num1 + num2$

step5: Display sum.

step6: stop.



# Flowchart:

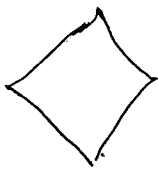
flowchart is a diagrammatic or pictorial representation of various steps involved in the algorithm.



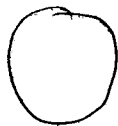
→ start / stop



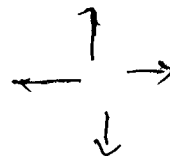
— loop



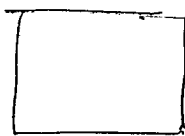
→ Decision



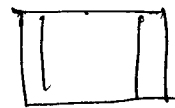
— connector



Data flow  
Direction  
lines.



Process

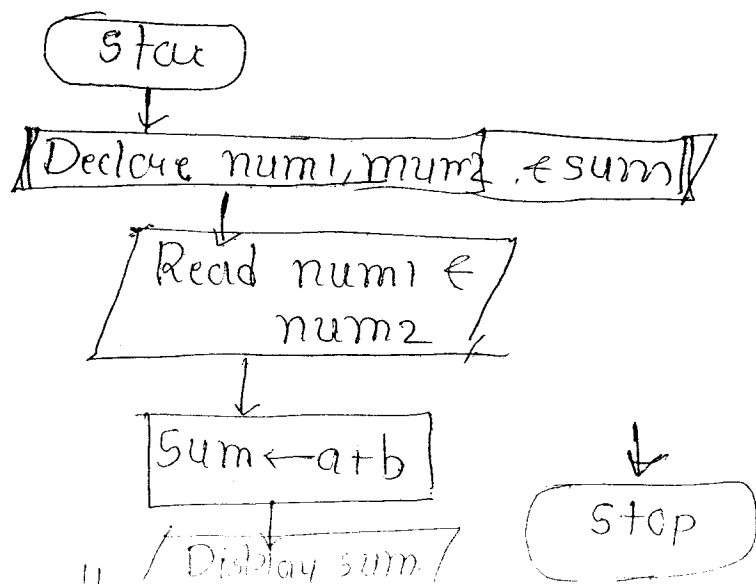


— Predefined  
Process



I/O

ex: Draw a flowchart to add two no. entered by the user.



## 1.6 Introduction to C language.

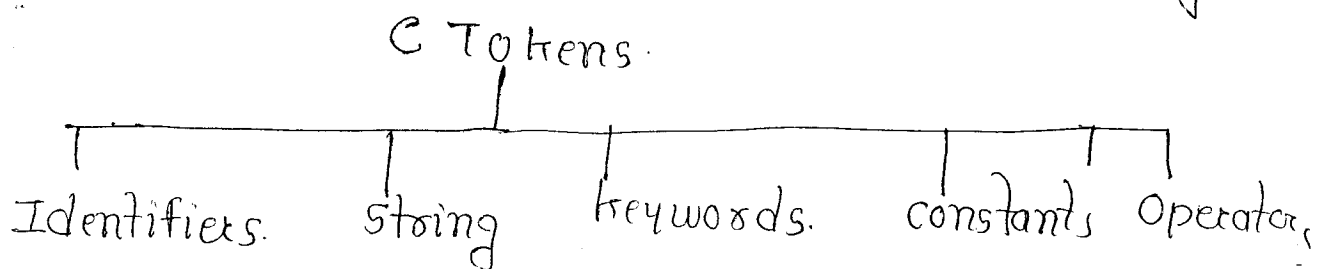
- C is a structured programming language.
- It is well suitable for developing application.
- C language is highly portable.

### History of 'C'

1960 - ALGOL-60 - International committee  
1963 - CPL - Cambridge university  
1967 - BCPL. Martine Richards.  
1970 B Ken Thomsen.  
1972 C Dennis Ritchie.

### C Tokens:

These are smallest individual unit, They may represent a single character or a group of characters which has a specific meaning.



## Identifiers:

The names that are given to the various program elements such as variables, functions, arrays, structures etc. These are user defined names.

## Keywords:

These are also called 'reserved words'. These keywords can be used only for their intended purpose. They can't be used as programmer defined identifiers.

C language supports nearly 32 keywords.

eg auto, Double, int, struct,  
Break, Else, etc

## Constants:

These are fixed values that will not change during the execution of prog.

1) Integer constant.

integer value numbers.

eg: 0, 1, 723, 01, 0723, 0x, 0x

## 2. Unsigned & long Integer Constants.

An unsigned integer constants can be identified by appending the letter  $U$

An long integer constant can be identified by appending the letter  $L$ , to the end of the const

An unsigned long integer constant can be specified by the letters  $UL$  to the end of const

eg      234U  
         0427UL  
         0x72FL

## 3: Floating point constant.

A floating pt. const is a decimal number that contains a decimal point.

eg. 0, 1, , 12.3

## 4. Character constants:

Character constant is a single character enclosed in a single quotation marks.

eg 'a', '1', '?'

## Variables:

A variable is an identifier that is used to represent a single data item. It changes its value during execution of the program.

## Operators:

An operator is a symbol that informs the computer to perform a particular task.

### ① Arithmetic Operators:

- + addition
- subtraction
- \* multiplication
- / Division
- % modular division.

### ② Relational Operators:

It is used to perform comparison between two values.

< less than

> greater than

<= less than or equal to

>= greater than or equal to

= exactly equal to

!= not equal to

## Logical Operator:

When we want to take a decision based on two or more conditions then we will use logical operators.

& logical AND

|| logical OR.

! logical NOT

## Assignment Operators:

This operator is used to assign a constant value the result of an expression to a variable. In this Right hand side expression is evaluated 1<sup>st</sup> & then the result is assigned to left hand side variable.

= assign RHS to LHS. value.

+= value of LHS add to value of RHS & assign back to the variable in LHS.

- = value of RHS variable will be subtracted from the value of LHS. & assign it back to the variable in LHS.

\* = value of LHS variable will be multiplied to the value of RHS & Assign it back to the variable in LHS.

## Increment / Decrement Operator

Increment operator increases the value by 1 & Decrement operator decreases the value of variable by 1.

## Bitwise Operator:

&	Bitwise logical AND.
	Bitwise logical OR.
^	Bitwise logical XOR.
<<	left shift
>>	Right shift.

## Conditional Operator: (?:)

It acts like a short hand version of if-else construction.

syntax: exp1 ? exp2 : exp3.

## Special Character:

- comma operator
- size of operator

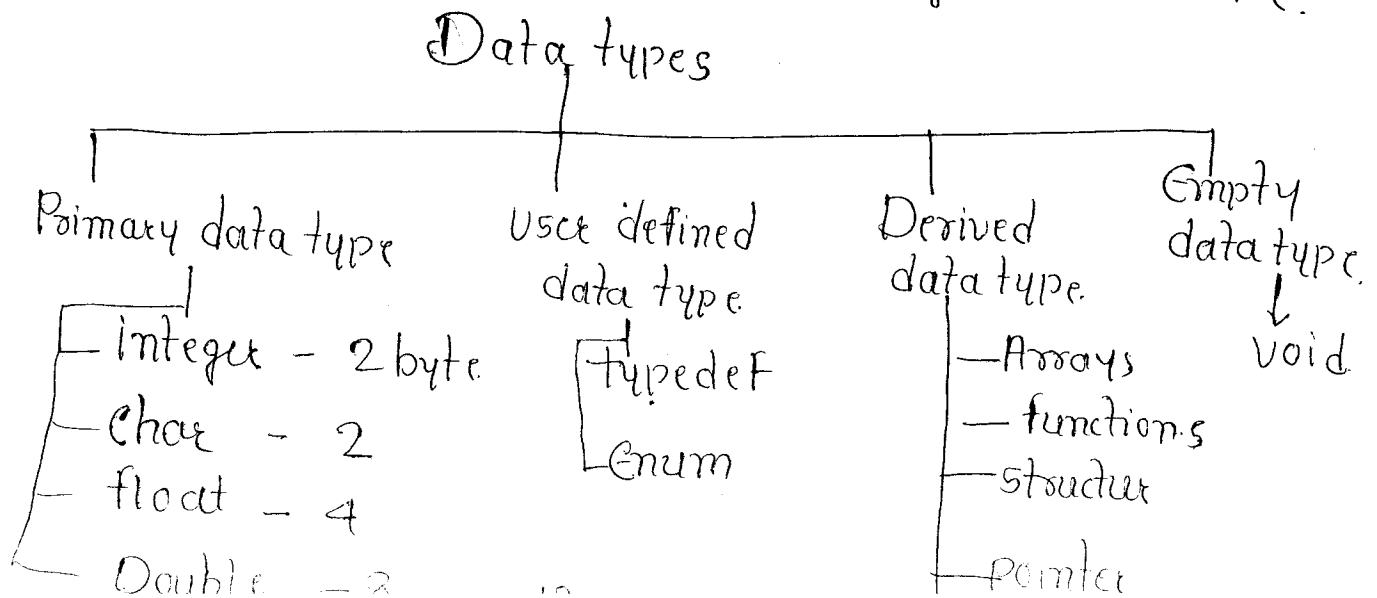
## 7. Precedence & Associativity

Precedence: The order in which operators are processed in a programming language.

Associativity: It specifies the order in which the operators are evaluated with the same precedence in a complex expression.

## Data types:-

It specifies the type of the data, the size of the data & about the range of the data.





## Expression:

An expression is a combination of variables, constants, operators & function call. It can be arithmetic, logical, & relational

e.g.  $\text{int } z = x + y$   
 $a > b$   
 $a == b$   
 $\text{func}(a, b)$

## 8. Statements:

① Selection Statement.

② ~~if~~ This statement changes the flow.

of execution depending on a given logical condition. Used to execute or skip some set of instructions based on given condition.

① if statement

syntax:  $\text{if (condition)}$

{

statement;

}

Rest of the prog;

if cond<sup>n</sup> is true the body of the if will execute otherwise control will come outside.

## ② If - else statement

if cond<sup>n</sup> true then if statement will execute otherwise else part will execute.

## ③ Nested if (Test expression)

```
    {  
    statements;  
else  
    {  
    statements;  
    }
```

## ③ Nested if-else

when series of decision are involved, we may have to use more than one if else statement in nested form.

Syntax:

```
if (condn)  
    {  
    if (cond)  
        statement  
    else  
        {  
            sta 2  
            {  
                else  
                {  
                    st 3
```

#### ④ switch-case.

It is used to select one options based on a given expression value,

Syntax:

```
switch (Variable)
{
    case value 1:
        Block 1;
        break;
    case value 2:
        Block;
        break;
    default:
        block;
}
```

## 1.9 loop control statement:

loop: A loop is defined as a block of statements which are repeatedly executed for certain number of times.

### ① for loop ;

The number of iteration of the loop is known before the loop is entered.

The 3 actions are mention into head of the loop. which are seperated by semicolon.

Syntax:

```
for (initialize expression; test condition;  
    { incrementation)
```

```
    statement-1;
```

```
    statement-2;
```

```
}
```

### ② while loop

It is a entry control loop statement.

Syntax:

```
Initialize expression;
```

```
while ( Test condn )
```

```
{
```

```
    Body of the loop;
```

### ③ "do-while loop":

To execute a part of program or code several times, we can use do-while loop of C lang.

Syntax:

```
Initialization Expression;  
do  
{  
    Body of the loop  
}  
while (Test condition);
```

### 10 Unconditional Control Statement:

a) break ;

It terminates the execution of the loop

Syntax:

```
Statement;  
break;
```

b) continue :

It is used to bypass the remainder of the current pass through a loop.

```
e.g Statement;  
continue;
```

## ② goto

It is used to branch unconditionally from one point to another in the program.

Syntax:

```
goto label;
```

```
==
```

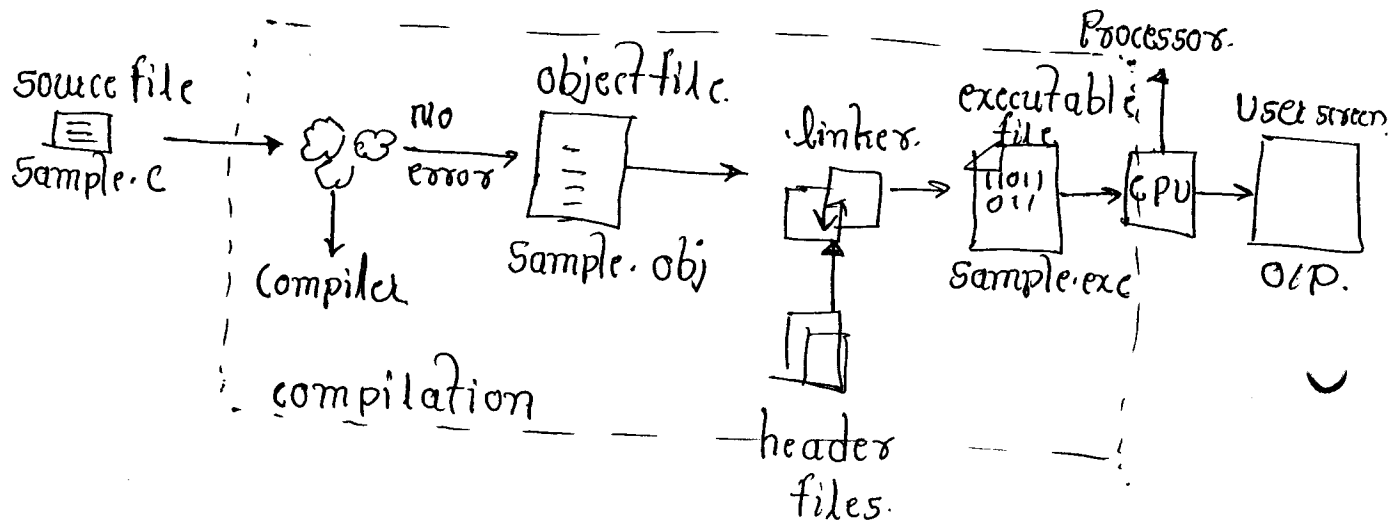
```
label;
```

```
statement;
```

Date 9/5/18

1) Execution process of a C program.

⇒ when we execute a C program it undergoes with following process.



→ The file which contains C prog instruction in high level language is said to be source code.

→ Every C prog source file is saved with .c extension.

eg sample.c

⇒ whenever we press alt+F9 the source file is submitted to the compiler. Compiler checks for the error, if there are any errors, it returns list of errors, otherwise generate object code in a file with name sample.obj. ← submit it to the linker.

linker combines the code from specified ~~folder~~ header file into the object file & generates executable files. as sample.exe.

⇒ To run the file, file has to be submit to CPU. Then CPU performs the task according to the instructions written in that program & place the result into User screen.

Algorithm example: (To add two numbers entered by the user)

step1: start.

step2: Declare variables num1, num2, & sum.

step3: Read values num1 & num2

step4: Add num1 & num2 & assign the result to sum

$$\text{sum} \leftarrow \text{num1} + \text{num2}$$

step5: Display sum

step6: Stop



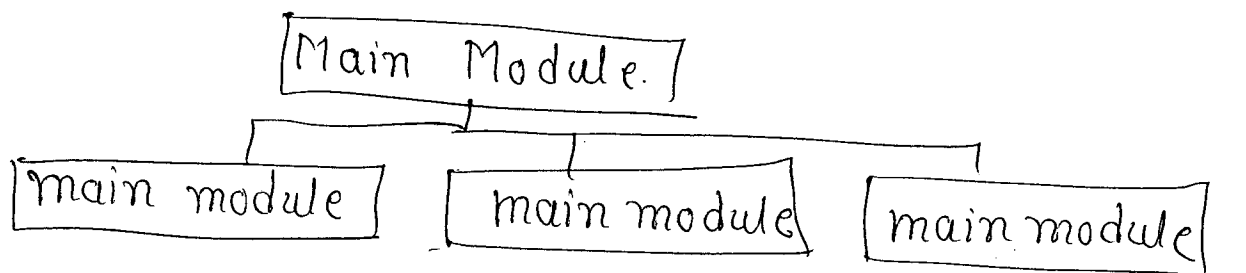
## Unit 2

### Functions:

#### 1) Designed Structure programs:

If the size of the problem to be solved is big then the complexity of problem increases & requires more number of instructions will make the problem big.

The big program is divided into smaller parts are called as modules.



### Structure chart

The main module is known as a calling module & the submodule is called module.

functions:

A function is a block of code that performs a specific task. It has name & it is reusable. It can return a value to calling program.

A computer program cannot handle all the tasks by itself. It request other program like entities called function in C.

## 2. User defined function:

A user can create their own functions for performing any specific task of program are called user defined functions.

To create & use these functions we have to know these three elements.

### ① function declaration:

The calling program should declare any function that is to be used later in the program. This is known as the function declaration or function prototype.

### ② Function Definition:

It consist of whole description & code of a function. It tells that what the funn

is doing & what are the i/p, o/p for that  
It consist of two parts.

① Function header : The 1<sup>st</sup> line of code

② Function body : whatever is written within `{}` is the body of the function.

③ Function call: In order to use the fun<sup>n</sup>. we need to invoke it at a required place in the program. This is known as the fun<sup>n</sup> call.

### 3. Inter function Communication.

A function is a selfcontained block of sub-program that perform a special task when it is called. Whenever a function is called to perform a specific task, then the called fun<sup>n</sup> perform that task & the result is return back to the calling function. The data flow between the calling & called fun<sup>n</sup> to perform a specific task is known as interfunction communication.

#### 4: Scope:

The scope of a variable is the portion of a program in which the variable may be visible or a violable.

There are three types of scope in which a variable can call.

- 1) Block scope
- 2) Function scope or local scope.
- 3) Global scope or file scope.

##### 1) Block Scope:

A variable is said to have block scope, if it is recognised only within the block where it is declared.

##### ② local scope:

A variable is said to have local scope or function scope, if it is recognised only within the function where it is declared.

##### ③ Global scope:

A variable is said to have global scope or file scope, if it is recognised within blocks & all the functions defined in a prog. A global variable can be declared outside of all funn within a file.

## 5. Storage Classes:

variable in C differ in behaviour. The behaviour depends on the storage class a variable may assume. From a compiler point of view, a variable name identifies some physical loc<sup>n</sup> within the computer where the string of bits representing the variable's value is stored.

There are four storage classes in C.

① Automatic storage class.

② Register storage class

③ Static storage class.

④ External storage class

① Automatic:

keyword: auto

: garbage value.

scope: local to the block in which it is defined.

② Register storage class.

storage is register, unpredictable default initial value.

scope is local to the block in which the variable is defined.

### ③ static storage class

Storage inside the memory, default initial value is 0, scope is local to the block in which the variable is defined.

Life is.

### ④ External Storage Class

Storage is in memory & default initial value is zero, scope life is global.

## 6. Recursion

Recursion is a process by which a function calls itself repeatedly, until some specified condition has been satisfied.

When function calls itself, a new set of local variable & parameters are allocated storage on the stack, & the function code is executed from the top with these new variable.

A recursive call does not make a new copy of the function. Only the values being operated upon are new.

As each recursive call returns, the old local variables & parameters are removed from the stack, & execution resumes immediately after the recursive call inside the function.

ex: void main()

```
{
    int n=5;
    fact(n);
}

int fact()
{
    if (n==0 || n==1)
        return 1;
    else
        return (n * fact(n-1));
}
```

## 7 : Array:

An array is defined as an ordered set of similar data items. All the data items of an array are stored in consecutive memory loc<sup>n</sup> in RAM. The elements of an array are of same data type & each item can be accessed using the same name.

### Declaration:

data type arrayname [n];

number of  
data items.

example     int a[5];  
               float x[10];

### Initialization

data type arrayname [size] =  
    { list of values };

## 2D Array:

An array consisting of two subscript is known as two-dimensional array



Declaration:

data type arrayname [row size] [colu size]

example:

int arr [3] [3].

## 8: Array Application:

① linear search:

It is a technique of searching an element in a list in sequence

Algorithm:

Step1: set up a flag to indicate "element not found."

Step2: Take the 1<sup>st</sup> element in the list.

Step3: If the element in the list is equal to the desired element

— set flag to "element found"

— Display the msg "element found in the list"

— Go to step 6.

Step4 If it is not the end of list.

— Take the next element in the list.

— Go to step 3

steps: if the flag is "element not found"

Display the message "element not found"

steps: End the Algorithm.

Time complexity  $O(n)$ .

## 3. Binary Search:

This algorithm is quicker than the linear search. However it cannot be applied on unsorted data structure. The binary search is based on the approach divide & conquer. The binary search starts by testing the data in the middle element of the array.

To implement binary search method, the elements must be in sorted order. Search is performed as follows.

- The key is compared with item in the middle pos<sup>n</sup> of an array.
- if it matches with item, return it & stop.
- if the key is less than mid pos<sup>n</sup> item,

then the item to be found must be in 1<sup>st</sup> half of array, otherwise it must be in second half of array.

→ Repeat the procedure for lower part of array until the element is found.

Date: 9/5/18.

Prog using function in C :

```
#include <stdio.h>
```

```
#include <conio.h>
```

```
int addition (int, int) ← funn decln.
```

```
int addition (int a, int b) ← function defn
```

```
{  
    int x;
```

```
    x = a + b;
```

```
    return (x);  
}
```

```
int main()
```

```
{  
    int z;
```

```
    z = addition (10, 3); ← function call.
```

```
    printf ("The result is %d", z);
```

```
    return 0;
```

```
}
```

Q10: 13

- why we have to use function.

⇒ 1) Writing functions avoids rewriting the same code over & over.

② Using function it becomes easier to write a prog. & keep track of what they are doing.

- Properties & advantage of userdefined function in C.

⇒ Properties:

- Every function has a unique name. This name is used to call function from main()
- A function performs a specific task.
- A function returns a value to the calling prog.

Advantages:

- function has topdown programming model. In this style of programming, the high level logic of the overall problem is solved first while the details of each lower

level functions is solved later.

- A C programmer can use functions written by others.
- Debugging is easier in function.
- It is easier to understand the logic involved in the program.
- Testing is easier.

• Difference between recursion & nonrecursion.

⇒ ① Recursive version of a prog. is slower than iterative version of a prog. due to overhead of maintaining stack.

② Recursive version of a prog. uses more memory than iterative version of a prog.

③ Sometimes, recursive version of a prog. is simpler to understand than iterative version of a prog.

## Unit III

### 1: Pointer:

Pointer is a special kind of variable used to store address of memory location through which indirect memory accessing can be performed when variables are declared memory is allocated to each variable. C provides data manipulation with address of variables therefore execution time is reduced.

Such concept is possible with special data type called pointer.

we know that memory is a collection of bits, in which eight bits constitute one byte. Each byte has its own unique location number called its address. To store this address we need a special kind of variable called pointer var.

To work with pointers we have two unary operators.

① Reference operator (&)

2) De-reference operator (\*)

## Reference Operator (&)

— called address operator, which gives the address of a variable, in which location the variable is resided in to the memory.

Syntax:  $\&$  variable-name.

For example: `int k=10;`

k  $\leftarrow$  variable name.

10
----

 $\leftarrow$  value  
2020      2021  $\leftarrow$  assumed address.

## Dereference Operator (\*)

This operator is used to get the value at a given address.

e.g.  $\ast(2020) = 10;$

The value at address 2020 is 10, this is k's value.

Syntax:  $\ast$  addressable-expression;

## 2. Pointer for Interfunction Communication:

⇒ Three types.

a) Downward Communication

b) Upward Communication.

c) Bi-direction Communication.

- Downward communication uses pass by value.
- Pass by address is used for both upward & bidirection communication.
- Pass by address can be effectively implemented by passing a pointer.

• Passing address:

- Exchange function: passing of two variables whose value are to be exchanged.
- call by value used in the following example. The data is exchanged in the called fun<sup>n</sup> but no changes to the calling program.



### 3 : function returning Pointer

A called function can also return a pointer

ex. pointer to the smaller of two variable  $a$  &  $b$ .

```
int * smaller (int * p1, int * p2);
```

```
int main(void)
```

```
{ int a;
```

```
  int b;
```

```
  int * p;
```

```
  scanf ("%d %d", &a, &b);
```

```
  p = smaller (&a, &b);
```

```
}
```

```
int * smaller (int * px, int * py)
```

```
{
```

```
  return (*px < *py ? px : py);
```

```
}
```

- when a pointer is returned, it must point to data in the calling function.
- when It is an error to return a pointer to a local variable in the called function.

## Pointer to Pointer:

A pointer to pointer is a form of multiple indirections, or a chain of pointer. Normally, a pointer contains the address of a variable. When we define a pointer to a pointer, the 1st pointer contains the address of the second pointer, which points to the location that contains the actual value as shown below

ex: `int **var;`

When a target value is indirectly pointed to by a pointer to pointer, accessing that value requires that the asterisk operator be applied twice as shown below in the example

```
#include <stdio.h>
```

```
int main()
```

```
{
```

```
    int var;
```

```
    int * ptr;
```

```
    int **pptr;
```

```
    var = 3000;
```

```
    ptr = &var;
```

```
    pptr = &ptr;
```

**Course Name: Programming for Problem Solving**  
**Year/Sem : I-I**

**Course code:CS103ES**  
**Regulation: R18**

**11. University previous question papers**

**R16**

Code No: 131AD

JAWAHARLAL NEHRU TECHNOLOGICAL UNIVERSITY HYDERABAD

B.Tech I Year I Semester Examinations, May/June - 2017

COMPUTER PROGRAMMING IN C

(Common to CE, ME, MCT, MMT, MIE, CEE, MSNT)

Time: 3 hours

Max. Marks: 75

**Note:** This question paper contains two parts A and B.

Part A is compulsory which carries 25 marks. Answer all questions in Part A.

Part B consists of 5 Units. Answer any one full question from each unit. Each question carries 10 marks and may have a, b, c as sub questions.

**PART- A****(25 Marks)**

- 1.a) What is the size of the double data type? Which conversion specifier is used? [2]
- b) Write program in C to interchange the two values without using third variable. [3]
- c) Distinguish between built – in and user – defined functions. [2]
- d) How one dimensional arrays are initialized? Give example. [3]
- e) List the dynamic memory handling functions used in 'C'. [2]
- f) List the advantages and disadvantages of using pointers. [3]
- g) Write the syntax for enumerated data type. Give example. [2]
- h) Give brief information about self referential structures. [3]
- i) Write the syntax for opening a file. Give example. [2]
- j) List the advantages of using files. [3]

**PART-B****(50 Marks)**

- 2.a) Write and explain the steps in writing a 'C' program.
  - b) Discuss about the various bitwise operators supported by Language 'C'. [5+5]
- OR**
- 3.a) Write 'C' program to print the Fibonacci sequence.
  - b) In what way a do – while loop differs from while loop. Explain. [5+5]
- 4.a) What is a function? What are its advantages? Explain various parameter passing techniques.
  - b) Write a 'C' program to search for an element by using Linear Search. [5+5]
- OR**
5. Why we need storage classes? List and explain the various storage classes present in language 'C'. [10]
  6. With the help of syntax and example program explain the various string handling functions. [10]
- OR**
- 7.a) Write in detail about the various dynamic memory allocation functions.
  - b) Write a program to accept a set of names and display them by using array of pointers. [5+5]

- 8.a) Write a 'C' program using functions to return the sum of two complex numbers passed as parameters.  
b) Write short notes on typedef. [5+5]

OR

- 9.a) Create a structure called **student** and the members of the structure are Stu\_Name, Stu\_Rno, M1, M2, M3. Create a **pointer variable** for the structure, store the values and fetch the values present in the structure student.  
b) In what way a Union differs from structures. [5+5]

- 10.a) Discuss in detail about the file positions functions.  
b) Write a 'C' program to count the number of words, white spaces and tab spaces present in a file. [5+5]

OR

- 11.a) Explain the file input and output functions with example programs.  
b) Distinguish between r, r+ and w, w+ modes of files. [5+5]

---ooOoo---

**R16**

Code No: 131AD

**JAWAHARLAL NEHRU TECHNOLOGICAL UNIVERSITY HYDERABAD****B.Tech I Year I Semester Examinations, December - 2017****COMPUTER PROGRAMMING IN C****(Common to CE, ME, MCT, MMT, AE, MIE, PTM, CEE, MSNT)****Time: 3 hours****Max. Marks: 75****Note:** This question paper contains two parts A and B.

Part A is compulsory which carries 25 marks. Answer all questions in Part A.  
Part B consists of 5 Units. Answer any one full question from each unit. Each question carries 10 marks and may have a, b, c as sub questions.

**PART- A****(25 Marks)**

- |      |  |     |
|------|--|-----|
| 1.a) | Explain rules of variables.                                    | [2] |
| b)   | Give the symbols of flow chart.                                | [3] |
| c)   | What are the limitations of recursion.                         | [2] |
| d)   | What are uses of binary search as compared with linear search. | [3] |
| e)   | Differentiate between gets() and scanf().                      | [2] |
| f)   | List the substrings that can be formed from the string "ABCD". | [3] |
| g)   | Differentiate between Union and Structure.                     | [2] |
| h)   | Explain about the typedef with an example.                     | [3] |
| i)   | What is the use of rewind( )?                                  | [2] |
| j)   | What is the impact of fclose( ) on buffered data.              | [3] |

**PART-B****(50 Marks)**

- |           |   |       |
|-----------|---|-------|
| 2.a)      | Explain the different types of bitwise operators are used in C.                       |       |
| b)        | Write an algorithm to find the greatest number among the three given numbers.         | [5+5] |
| <b>OR</b> |   |       |
| 3.a)      | Write a program to swap the two numbers without using a temporary variable.           |       |
| b)        | Explain the different types of operators are used in C.                               | [5+5] |
| 4.a)      | Write a program to implement the binary search by using functions.                    |       |
| b)        | Explain the two dimensional arrays for Inter Function Communication.                  | [5+5] |
| <b>OR</b> |   |       |
| 5.a)      | Write a program to generate Pascal's triangle by using an array.                      |       |
| b)        | What is recursion? Explain the different types of recursion.                          | [5+5] |
| 6.a)      | Explain the dynamic memory functions.   |       |
| b)        | Write a program to count the number of times a given character appears in the string. | [5+5] |
| <b>OR</b> |   |       |
| 7.a)      | Write a program to sort the names of employees alphabetically.                        |       |
| b)        | Explain the rules for pointer operations.   | [5+5] |

- 8.a) Write a program using pointer to structure to initialize the members in the structure. Use functions to print the student's information.  
b) Explain the Enumerated data types with an example. [5+5]

**OR**

- 9.a) Explain the command line arguments with an example.  
b) Write a program to define a structure for hotel that has members - name, address, grade, number of rooms and room charges. Write a function to print the names of a hotel in a particular grade. Also write a function to print names of a hotel that have room charges less than the specified value. [5+5]
- 10.a) Explain the different types of files are used in C.  
b) Write a program to print the records in reverse order. The file must be opened in binary mode. [5+5]

**OR**

- 11.a) Write a program to append a binary file at the end of another.  
b) Explain the function calls of fseek( ). [5+5]

---ooOoo---

Code No: 131AD

**R16**

**JAWAHARLAL NEHRU TECHNOLOGICAL UNIVERSITY HYDERABAD**

**B.Tech I Year I Semester Examinations, December - 2016**

**COMPUTER PROGRAMMING IN C**

**(Common to CE, ME, MCT, MMT, MIE, CEE, MSNT)**

**Time: 3 hours**

**Max. Marks: 75**

**Note:** This question paper contains two parts A and B.

Part A is compulsory which carries 25 marks. Answer all questions in Part A.

Part B consists of 5 Units. Answer any one full question from each unit. Each question carries 10 marks and may have a, b, c as sub questions.

**PART- A**

**(25 Marks)**

- 1.a) Define the term algorithm and how it is different from a flow chart. [2]
- b) Distinguish between if and switch statements. [3]
- c) Explain auto and register storage classes. [2]
- d) What is meant by type qualifier? [3]
- e) What is meant by array of pointers? When it will be useful? [2]
- f) Distinguish between string and character. [3]
- g) Explain how to define a union. [2]
- h) Explain the concept of enumerated type. [3]
- i) What is meant by a text file? [2]
- j) Discuss about rewind function? [3]

**PART-B**

**(50 Marks)**

2. Write a C program to find factorial of a given number N by using 'while' loop and 'do-while' loop. [10]

**OR**

3. What are the C operators? Explain their usage with suitable examples to each of them. [10]

- 4.a) Distinguish between iteration and recursion.

- b) What is meant by user defined functions? Explain with an example C Program. [5+5]

**OR**

- 5.a) Explain the binary search method.

- b) Perform bubble sort of the following numbers:  
20, 5, 30, 10, 65, 3, 90 [5+5]

- 6.a) What are the advantages and drawbacks of pointers?

- b) With a sample C program and explain about passing an array to a function. [5+5]

**OR**

7. What are the string manipulation functions? Explain their usage. [10]



8. Explain about self referential structures with a sample C program. [10]  
**OR**
- 9.a) Discuss the significance of preprocessor commands. [5+5]  
b) Explain about nested structures.
10. What is meant by state of a file? Write a C program to copy the contents of one text file to another text file. [10]  
**OR**
11. What is meant by binary file? Discuss about file positioning functions? [10]

---ooOoo---

Code No: 131AD

R16

JAWAHARLAL NEHRU TECHNOLOGICAL UNIVERSITY HYDERABAD

B.Tech I Year I Semester Examinations, May - 2018

COMPUTER PROGRAMMING IN C

(Common to CE, ME, MCT, MMT, AE, MIE, PTM, CEE, MSNT)

Time: 3 hours

Max. Marks: 75

Note: This question paper contains two parts A and B.

Part A is compulsory which carries 25 marks. Answer all questions in Part A.

Part B consists of 5 Units. Answer any one full question from each unit. Each question carries 10 marks and may have a, b, c as sub questions.

PART- A

(25 Marks)

- 1.a) Which statement is multi way selection statement? Why? [2]
- b) What is meant by time sharing environment? [2]
- c) Evaluate the value of the following when  $x=3.45$   
 $\text{floor}(x * 100 + 0.5) / 100$  [3]
- d) Can an assignment operator copy one array to another? Justify your answer. [2]
- e) Give the picture to show the memory configuration for the declaration  
`int (*a) [5];` [3]
- f) Differentiate between `strspn` and `strcspn`. [2]
- g) What is the need of `typedef` command? [3]
- h) Is macro an inline function? Justify your answer. [2]
- i) Contrast text files and binary files. [3]
- j) What is a system created stream? Give examples. [2]

PART-B

(50 Marks)

- 2.a) Write an algorithm to find LCM of two numbers. [5+5]
- b) Describe the process of program development. [5+5]
- OR
- 3.a) What are the three differences between the conversion codes for input formatting and output formatting? Explain them with examples. [5+5]
- b) What is the need of explicit type conversion in C? How to cast the data? [5+5]
- 4.a) Write a recursive function to solve towers of Hanoi problem and trace it with different input. [5+5]
- b) Discuss various storage classes of C. [5+5]
- OR
- 5.a) Write a function that copies a one-dimensional array of  $n$  elements into a two-dimensional array of  $k$ -rows and  $j$ -columns. The rows and columns must be a valid factor of the number of elements in the one-dimensional array i.e.,  $k * j = n$ . [5+5]
- b) Discuss the different ways of passing arrays as a parameter to a function. [5+5]

AG AG AG AG AG AG AG A

- 6.a) Discuss dynamic memory management in C.  
b) Explain in detail applications of pointers.

[5+5]

OR

7. Write a C program that converts a string representing a number in Roman numeral form to decimal form. [Follow regular convention for Roman numbers. Read string - parse it to convert in to decimal]  
Eg: Input: XL output: 40

[10]

- 8.a) How to pass a structure to a function? Give illustrations.  
b) Define a structure of arrays. Write code to read values in to this structure.

[5+5]

OR

- 9.a) Is enumeration a derived data type? Justify your answer with suitable example.  
b) Discuss any three types of preprocessor commands.

[5+5]

- 10.a) Compare formatting input/output functions with scanf and printf.  
b) What is the purpose of ungetc() function.

[5+5]

OR

11. Write a program to read two file names, append the first file content at the end of the second file content.

[10]

AG AG AG AG AG AG AG A

—ooOoo—

AG AG AG AG AG AG AG A

AG AG AG AG AG AG AG A

AG AG AG AG AG AG AG A

AG AG AG AG AG AG AG A

Code No: 132AD

R16

JAWAHARLAL NEHRU TECHNOLOGICAL UNIVERSITY HYDERABAD

B.Tech I Year II Semester Examinations, April - 2018

COMPUTER PROGRAMMING IN C

(Common to EEE, ECE, CSE, EIE, IT, ETM)

Time: 3 hours

Max. Marks: 75

Note: This question paper contains two parts A and B.

Part A is compulsory which carries 25 marks. Answer all questions in Part A.

Part B consists of 5 Units. Answer any one full question from each unit. Each question carries 10 marks and may have a, b, c as sub questions.

PART- A

(25 Marks)

- 1.a) What are flowchart symbols? Explain. [2]
- b) Explain the syntax of case statement with example. [3]
- c) What are limitations of recursion? [2]
- d) Explain the procedure for Binary search. [3]
- e) Give an example for pointers to pointers. [2]
- f) What are string functions? Explain. [3]
- g) Define structure with example. [2]
- h) What is self referential structure? Explain. [3]
- i) Write the concept of a file. [2]
- j) Explain about file status functions. [3]

PART-B

(50 Marks)

- 2.a) Write a C program to generate Fibonacci series.
- b) What are operators in C? Explain with example. [5+5]

OR

- 3.a) Explain about decision-making and branching.
- b) Describe the structure of a C program. [5+5]

- 4.a) Write a C program for Bubble sort.
- b) What are storage classes? Explain with examples. [5+5]

OR

- 5.a) What is recursion? Write a C program for factorial of n using recursion.
- b) Write a C program for addition of two matrices. [5+5]

- 6.a) Write a C program to count the number of vowels, consonants, digits and white-spaces in a string which is entered by the user.
- b) Explain about pointer arithmetic and arrays with example. [5+5]

OR

- 7.a) Write a C program to sort the 10 strings (entered by the user) in lexicographical order (dictionary order).
- b) Explain about arrays of pointers with example. [5+5]

AG AG AG AG AG AG AG A

8.a) Write a C program that takes two complex numbers as structures and adds them with the use of functions.

b) Explain about pointers to structures with example. [5+5]

OR

9.a) Explain about pre-processor commands. AG AG AG AG A  
b) Explain about unions and functions. [5+5]

10.a) Write a C program to copy the content from one file to another file using fseek( ) function.

b) Write a C program to copy the binary file from another file. [5+5]

OR

11.a) Write a C program read and write the content of the file using fprintf( ) and fscanf( ) functions. AG AG AG AG A  
b) Briefly explain about positioning functions in files. [5+5]

AG AG AG AG AG AG AG A

AG AG AG AG AG AG AG A

AG AG AG AG AG AG AG A

AG AG AG AG AG AG AG A

AG AG AG AG AG AG AG A

Code No: 121AF

**R15**

**JAWAHARLAL NEHRU TECHNOLOGICAL UNIVERSITY, HYDERABAD**

**B.Tech I Year Examinations, May - 2016**

**COMPUTER PROGRAMMING**

**(Common to CE, EEE, ME, ECE, CSE, EIE, IT, MCT, ETM, MMT, AE, AME, MIE, PTE, CEE, MSNT)**

**Time: 3 hours**

**Max. Marks: 75**

**Note:** This question paper contains two parts A and B.

Part A is compulsory which carries 25 marks. Answer all questions in Part A.

Part B consists of 5 Units. Answer any one full question from each unit.

Each question carries 10 marks and may have a, b, c as sub questions.

**PART- A**

**(25 Marks)**

- 1.a) Write brief notes on computer languages. [2]
- b) Discuss the significance of 'continue' statement with an example. [3]
- c) What is meant by type qualifiers? [2]
- d) Explain scope of a variable with an example. [3]
- e) What are the memory allocation functions? [2]
- f) What is meant by array of pointers? When it will be used. [3]
- g) Explain about positioning functions. [2]
- h) Discuss about bit fields. [3]
- i) Explain Enqueue operations. [2]
- j) What is meant by sorting? Give an example. [3]

**PART-B**

**(50 Marks)**

2. Write a C program to find factorial of a given number 'N' by using iteration and recursion separately. [10]
3. Explain switch statement. Explain its usage with a sample C-program. [10]
4. What are the storage classes in C? Explain their usage with a sample C-program. [10]
5. Explain inter function communication with a C-program. [10]
6. Explain pointer arithmetic with a sample C-program. [10]
7. What are the string manipulation functions? Explain their usage. [10]
8. What is meant by structure? Discuss with a C-program about operations on structures. [10]
9. What is meant by state of a file? Discuss about file status functions. [10]
10. Write a C-program for implementation of singly linked list. [10]
11. Write a C-program for implementing Dequeue operations. [10]

---ooOoo---

**R16**

Code No: 132AD

JAWAHARLAL NEHRU TECHNOLOGICAL UNIVERSITY HYDERABAD

B.Tech I Year II Semester Examinations, August/September - 2017

COMPUTER PROGRAMMING IN C

(Common to EEE, ECE, CSE, EIE, IT)

Time: 3 hours

Max. Marks: 75

**Note:** This question paper contains two parts A and B.

Part A is compulsory which carries 25 marks. Answer all questions in Part A.

Part B consists of 5 Units. Answer any one full question from each unit. Each question carries 10 marks and may have a, b, c as sub questions.

**PART- A****(25 Marks)**

- 1.a) Give the decimal equivalent of  $7AC_{(16)}$ . [2]
- b) What is ternary operator in C? Give example expression for its use. [3]
- c) List the advantages of functions. [2]
- d) How to declare and initialize a multi dimensional array? [3]
- e) What is the purpose of pointers? [2]
- f) List the input/output functions for strings. [3]
- g) What are the different type castings supported in C? [2]
- h) Give an example for nested structure. [3]
- i) Contrast fread and fscanf functions. [2]
- j) What is meant by opening a data file? How is this accomplished? [3]

**PART-B****(50 Marks)**

- 2.a) Why is C language known as middle-level language? [5]
  - b) Draw a flow chart to find average of 10 numbers. [5]
  - OR**
  - 3.a) What is associativity? Illustrate its application in expression evaluation. [5]
  - b) Which statement is a multi way selection statement? Why? [5]
  4. What is the difference between actual and formal parameters? With illustrative examples explain parameter passing techniques. [10]
  - OR**
  - 5.a) Write a recursive function to print Fibonacci sequence. [5]
  - b) Discuss the applications of arrays. [5]
  - 6.a) How to declare a pointer to a function? What is its use? [5]
  - b) What is the difference between calloc and malloc functions? [5]
  - OR**
  - 7.a) Write a program to display the location of a character 'T' in a given string. [5]
  - b) Give the signatures of getch, puts functions. [5]
-

- 8.a) Differentiate between structure and union? Give an example usage.  
b) What is the benefit of bitfield usage?

[5+5]

**OR**

9. Write a program using structures to search a name in a record of ten mobile subscribers and print the name, address, bill number and amount of the searched record.  
[10]
10. Write a program to copy contents of one file to another using file names passed as the command line arguments.  
[10]

**OR**

- 11.a) What are the modes in which files can be opened?  
b) Write a program to store students information (id, name, address, marks) into a file and print the information from the file.  
[5+5]

---ooOoo---

---