4TH SEMESTER

SYLLABUS

DESIGN AND ANALYSIS OF ALGORITHMS			
Course code:	**IPCC21IS42**	**Credits:**	4
L: T:P:	3:0:2	**CIE Marks:**	50
Exam Hours:	3	**SEE Marks:**	100
Total Hours:	50		

COURSE OBJECTIVES:	
1.	**Analyze** the non-recursive, recursive algorithms and Brute Force algorithms and to represent efficiency of these algorithms in terms of the standard Asymptotic notations.
2.	**Devise** the Divide and Conquer and Decrease and Conquer techniques to design the algorithms and **Apply** these methods in designing algorithms to solve a given problem.
3.	**Explain** the Time versus Space Trade-offs and **Implement** dynamic programming methods and apply these methods in designing algorithms to solve a given problem
4.	**Get** the idea of Greedy method and **Describe** the limitations of Algorithmic power
5.	**Illustrate** the idea of Backtracking and Branch and Bound algorithm design techniques to solve a given problem

COURSE OUTCOMES: AT THE END OF THE COURSE, STUDENT WILL BE ABLE TO:	
CO1	**Explain** the algorithm design techniques and standard Asymptotic notations. **Analyze** non-recursive and recursive algorithms to obtain worst-case running times of algorithms using asymptotic analysis. Describe the Brute-force technique and analyze the performance of its algorithms
CO2	**Describe** the Divide-and-Conquer and Decrease-and-Conquer paradigms. Explain when an algorithmic design situation calls for it. Recite algorithms that employ the paradigms. **Solve** recurrences describing the performance of Divide-and-Conquer algorithms.
CO3	**Explain** Time and Space Trade-offs in designing algorithms. Describe the Dynamic-Programming paradigm and explain when an algorithmic design situation calls for it. Recite algorithms that employ this paradigm. **Apply** and **solve** Dynamic-Programming algorithms.

CO4	**Describe** the Greedy technique and explain when an algorithmic design situation calls for it. Recite algorithms that employ these paradigms. **Apply** and **solve** Greedy algorithms. **Compare** and contrast the limitations of various algorithms and finding approximate solution to them
CO5	**Describe** the Backtracking and Branch and Bound algorithm design paradigms and analyze them
CO6	**Design , Develop and Implement** the given problems using appropriate algorithm techniques and **analyze** their performance

MAPPING OF COURSE OUTCOMES TO PROGRAM OUTCOMES:

	PO1	PO2	PO3	PO4	PO5	PO6	PO7	PO8	PO9	PO10	PO11	PO12	PSO1	PSO2	PSO3
CO1	3	3			2				2	2		2	3	2	1
CO2	3	3			2				2	2		2	3	2	1
CO3	3	3			2				2	2		2	3	2	1
CO4	3	3			2				2	2		2	3	2	1
CO5	2	2			2				2	2		1	3	2	1
CO6	3	3	3		2				2	2		1	3	2	1

Module	Module Contents	Hours	CO's
1.	**INTRODUCTION:** What is an algorithm, Fundamentals of Algorithmic Problem Solving, Fundamental Data Structures, Fundamentals of Analysis of Algorithm Efficiency: Analysis Framework, Asymptotic Notations, Mathematical Analysis of Recursive and Non-recursive Algorithms **BRUTE FORCE**: Selection sort and Bubble Sort, Sequential Search and String Matching	**08**	**1**
2.	**DIVIDE AND CONQUER**: Recurrences – Substitution Method, Recursion Tree Method, Master Method, Merge Sort , Quick Sort, Binary Search, Strassen's Matrix Multiplication **DECREASE AND CONQUER:** Introduction, Insertion Sort, Depth-First Search and Breadth-First Search, Topological Sorting	**08**	**2**

3.	**TRANSFORM AND CONQUER**: Heaps and Heapsort **SPACE AND TIME TRADEOFFS**: Boyer-Moore Algorithm, Hashing **DYNAMIC PROGRAMMING:** Introduction, Warshall's and Floyd's Algorithms, The Knapsack Problem and Memory Functions	**08**	**3**
4.	**GREEDY TECHNIQUE:** Introduction, Prim's Algorithm, Kruskal's Algorithm, Dijkstra's Algorithm, Huffman Trees and Codes. **LIMITATIONS OF ALGORITHM POWER:** Lower-Bound Arguments, Decision Trees, P, NP, and NP Complete 5Problems	**08**	**4**
5.	**COPING WITH THE LIMITATIONS OF ALGORITHM POWER:** Backtracking & Branch-and-Bound: nqueens problem, sum of subset, assignment, problem, Knapsack, Traveling Salesman Problem.	**08**	**5**

Expt. No	Contents of the experiment	Hours	CO's
1.	Design, develop, and execute a Binary Search program using Recursion	**2**	**1, 6**
2.	Design, develop, and execute a program called MERGE_SORT to sort a given set of elements using the merge sort method and determine the time required to sort the elements. Repeat the experiment for different values of n, the number of elements in the list to be sorted and plot a graph of the time taken versus n. The elements should be read from a file and also be generated using the random number generator.	**2**	**2, 6**
3.	Design, develop, and execute a program called QUICK_SORT to sort a given set of elements using the Quick sort method and determine the time required to sort the elements. Repeat the experiment for different values of n, the number of elements in the list to be sorted and plot a graph of the time taken versus n. The elements should be read from a file and also be generated using the random number generator.	**2**	**2, 6**
4.	A) Design, develop, and execute a program called BFS. Print all the nodes reachable from a given starting node in a digraph using BFS method. B) Design, develop, and execute a program called DFS. Print all the nodes reachable from a given starting node in a digraph using DFS method.	**2**	**2, 6**

5.	A) Design, develop, and execute a program to create a class called FLOYDS that represents the cost adjacency matrix and member functions to Implement All-Pairs Shortest Paths Problem using Floyd's algorithm. B) Design, develop, and execute a program called WARSHALL to compute the transitive closure of a given directed graph using Warshall's algorithm. Print all the matrices of transitive closure.	**2**	**3, 6**
6.	Design, develop and execute a program to sort an array of elements using Heap Sort Technique.	**2**	**3, 6**
7.	Design, develop, and execute a program called KNAPSACK to Implement 0/1 Knapsack problem using Dynamic Programming.	**2**	**4, 6**
8.	Design, develop, and execute a program called KRUSKAL. Find Minimum Cost Spanning Tree of a given undirected graph using Kruskal's algorithm.	**2**	**4, 6**
9.	Design, develop, and execute a program called DIJKSTRA'S. From a given vertex in a weighted connected graph, find shortest paths to other vertices using Dijkstra's algorithm.	**2**	**4, 6**
10.	Design, develop, and execute a program called N QUEENS to Implement N Queen's problem using Back Tracking.	**2**	**5, 6**

TEXT BOOKS:		
TB No.	**Author / Edition/ Publication / Year**	**Chapters**
1.	AnanyLevitin: Introduction to the Design & Analysis of Algorithms, 3rd Edition, Pearson Education, 2012.	1, 2, 3, 4, 5, 6, 7, 8, 9, 10, 11
2.	Thomas H. Cormen, Charles E. Leiserson, Ronal L. Rivest, Clifford Stein: Introduction to Algorithms, 3rd Edition, PHI, 2010.	4, 11, 15, 16, 23, 24, 25

REFERENCE BOOKS:		
RB No.	**Author / Edition/ Publication / Year**	**Chapters**
1.	Ellis Horowitz, Sartaj Sahni, Sanguthevar Rajasekaran: Fundamentals of Computer Algorithms, 2nd Edition, Universities Press.	
2.	Kenneth A. Berman, Jerome L. Paul: Algorithms, Cengage Learning, 2002.	

MICROPROCESSORS & MICROCONTROLLERS			
Course code:	**IPCC21IS43**	**Credits:**	4
L: T:P:	3:0:2	**CIE Marks:**	50
Exam Hours:	5	**SEE Marks:**	50
Total Hours:			

COURSE OBJECTIVES:	
1.	Understand the architecture of 8086 microprocessor
2.	Able to write software and hardware programs using assembly language programming
3.	Understand the difference between microprocessors and microcontrollers
4.	Understand the architecture of ARM processor

COURSE OUTCOMES: AT THE END OF THE COURSE, STUDENT WILL BE ABLE TO:	
CO1	Analyze the architecture of Microprocessor
CO2	Apply the concept of data accessing using different addressing modes to write simple ALPs
CO3	Apply the defined instruction set to design assembly level language programs
CO4	Compare and Contrast the difference between Microprocessors and microcontrollers
CO5	the architecture of ARM processor
CO6	Understand the architecture of Arduino and simple programming principles

MAPPING OF COURSE OUTCOMES TO PROGRAM OUTCOMES:

	PO1	**PO2**	**PO3**	**PO4**	**PO5**	**PO6**	**PO7**	**PO8**	**PO9**	**PO10**	**PO11**	**PO12**	**PSO1**	**PSO2**	**PSO3**
CO1		3	3		3	1	1		1	1		3			2
CO2		3	3		3	1	1		1	1		3			2
CO3		3	3		3	1	1		1	1		3			2
CO4		3	3		3	1	1		1	1		3			2
CO5		3	3		3	1	1		1	1		3			2
CO6		3	3		3	1	1		1	1	1	3			2

Module	Module Contents	Hours	CO's
1.	**Introduction, Microprocessor Architecture – 1:** The Microprocessor-Based Personal Computer Systems. The Microprocessor and its Architecture: Internal Microprocessor Architecture, Real Mode Memory Addressing.	**8**	**1, 2**
2.	**Microprocessor Architecture – 2, Addressing Modes:** Introduction to Protected Mode Memory Addressing, Memory Paging, Flat Mode Memory **Addressing Modes:** Data Addressing Modes, Program Memory Addressing Modes, Stack Memory Addressing Modes	**8**	**2, 3**
3.	**Programming – 1:** Data Movement Instructions. PUSH/POP, Load-Effective Address, String Data Transfers, Miscellaneous Data Transfer Instructions, Segment Override Prefix, Assembler Details. Arithmetic and Logic Instructions: Addition, Subtraction and Comparison, Multiplication and Division **Programming – 2:** Arithmetic and Logic Instructions (continued): BCD and ASCII Arithmetic, Basic Logic Instructions, Shift and Rotate, String Comparisons. Program Control Instructions: The Jump Group	**8**	**3, 4**
4.	**Microprocessors versus Microcontrollers, ARM Embedded Systems:** The RISC design philosophy, The ARM Design Philosophy, Embedded System Hardware, Embedded System Software, ARM Processor Fundamentals: Registers, Current Program Status Register	**8**	**4, 5**
5.	**Introduction to the ARM Instruction Set:** Data Processing Instructions, Branch Instructions, Software Interrupt Instructions, Program Status Register Instructions **Programming with Arduino:** Understanding the ecosystem of Arduino, Pinout configuration, Digital input and output, Analog input and output, working with sensors and actuators	**8**	**5, 6**

Expt. No	Contents of the experiment	Hours	CO's
1.	Design and develop an assembly language program to search a key element "X" in a list of 'n'16-bit numbers. Adopt Binary search algorithm in your program for searching	**2**	**1, 2**
2.	Design and develop an assembly program to sort a given set of 'n' 16-bit numbers in ascending order. Adopt Bubble Sort algorithm to sort given elements	**2**	**1, 2**
3.	Develop an assembly language program to reverse a given string and verify whether it is a palindrome or not. Display the appropriate message.	**2**	**2, 3**
4.	Develop an assembly language program to compute nCr using recursive procedure. Assume that 'n' and 'r' are non-negative integers.	**2**	**3, 4**
5.	Design and develop an assembly language program to read the current time and Date from the system and display it in the standard format on the screen.	**2**	**4, 5**
6.	Design and develop an ARM assembly language program to find the factorial of a number.	**2**	**4, 5**
7.	To write and simulate ARM assembly language programs for data transfer, arithmetic and logical operations (Demonstrate with the help of a suitable program).	**2**	**4, 5**
8.	Design and develop an ARM assembly language program to find the square of a number (1 to 10) using look up table.	**2**	**4, 5**
9.	Write and simulate an ARM assembly language program to multiply two 16-bit binary numbers.	**2**	**4, 5**
10.	Design and develop an ARM assembly language program to find the sum of first 10 integer numbers.	**2**	**4, 5**

TEXT BOOKS:		
TB No.	**Author / Edition / Publication / Year**	**Chapters**
1.	Barry B Brey: The Intel Microprocessors, 8th Edition, Pearson Education, 2009.	
2.	ARM system developers guide, Andrew N Sloss, Dominic Symes and Chris Wright, Elsevier, Morgan Kaufman publishers, 2008.	
3.	ARDUINO PROGRAMMING: 3 books in 1 - The Ultimate Beginners, Intermediate and Expert Guide to Master Arduino Programming	1, 2, 3

REFERENCE BOOKS:		
RB No.	**Author / Edition / Publication / Year**	**Chapters**
1.	Douglas V. Hall: Microprocessors and Interfacing, Revised 2nd Edition, TMH, 2006	
2.	K. Udaya Kumar & B.S. Umashankar: Advanced Microprocessors & IBM-PC Assembly Language Programming, TMH 2003.	
3.	Ayala: The 8086 Microprocessor: programming and interfacing - 1st edition, Cengage Learning	
4.	The Definitive Guide to the ARM Cortex-M3, by Joseph Yiu, 2nd Edition, Newnes, 2009	

OPERATING SYSTEMS			
Course code:	**PCC21IS44**	**Credits:**	03
L: T:P:	2:2:0	**CIE Marks:**	50
Exam Hours:	03	**SEE Marks:**	50
Total Hours:	40		

COURSE OBJECTIVES:	
1.	Understand the structure and services of the operating system that provides to users and system
2.	Know the various CPU scheduling algorithms and multithreading concepts
3.	Identify methods for handling deadlocks and recognize the classic synchronization problems
4.	To gain knowledge on the various memory management techniques and file systems with their storage structure
5.	Realize the different concepts of Operating Systems through case study

COURSE OUTCOMES: AT THE END OF THE COURSE, STUDENT WILL BE ABLE TO:	
CO1	Summarize types, features and design considerations of operating systems
CO2	Analyze and Apply the various process scheduling algorithms.
CO3	Solve appropriate deadlock handling and synchronization problems.
CO4	Analyze memory management techniques and choose the suitable algorithm.
CO5	Recognize the design considerations of file system and compare various disk scheduling algorithms.
CO6	Explore about case study of Linux operating system

MAPPING OF COURSE OUTCOMES TO PROGRAM OUTCOMES:														
	PO1	**PO2**	**PO3**	**PO4**	**PO5**	**PO6**	**PO7**	**PO8**	**PO9**	**PO10**	**PO11**	**PO12**	**PSO1**	**PSO2**	**PSO3**
CO1	2	2	2	-	-	-	-	-	-	-	-	2	-	2	1
CO2	3	2	2	-	-	-	-	-	-	-	-	2	-	2	1
CO3	3	2	2	-	-	-	-	-	-	-	-	2	-	2	1
CO4	3	2	2	-	-	-	-	-	-	-	-	2	-	2	1
CO5	3	2	2	-	-	-	-	-	-	-	-	2	-	2	1
CO6	2	2	2	-	-	-	-	-	-	-	-	2	-	2	1

Module	Module Contents	Hours	CO's
1.	**OVERVIEW-Introduction:** What operating systems do; Computer system architecture, Operating System operations. **System Structures:** Operating System Services, System calls, Operating system structure.	**08**	**CO1**
2.	**PROCESS MANAGEMENT**: **Process concept**: Process scheduling, interprocess communication, **Threads:** Overview; Multithreading models; **CPU Scheduling:** Basic concepts, scheduling criteria, Scheduling algorithms: FCFS (5.3.1), SJF(5.3.2), Priority Scheduling (5..3.4) and Round Robin Scheduling(5.3.3).	**08**	**CO2**
3.	**PROCESS SYNCHRONIZATION - Synchronization:** The Critical section problem, Peterson's solution, Semaphores; Classical problems of synchronization: Bounded-Buffer Problem, Reader's-Writer's Problem. **Deadlocks:** System model; Deadlock characterization; Methods for handling deadlocks; Deadlock prevention; Deadlock avoidance; Deadlock detection and recovery from deadlock.	**08**	**CO3**
4.	**MEMORY MANAGEMENT - Main Memory:** Background, Contiguous memory allocation, Paging, Swapping. **Virtual Memory:** Background, Demand Paging, Basic Page Replacement Algorithms: FIFO(10.4.2), Optimal(10.4.3), LRU(10.4.4), **File System:** File concept, Access methods, Protection.	**08**	**CO4, CO5**

5.	**STORAGE MANAGEMENT- Mass Storage Structure**: Overview of Mass storage structure; HDD scheduling algorithms. **PROTECTION AND SECURITY-System Protection:** Goals of protection, Principles of protection, Domain of protection, Access matrix. **Case Study: The Linux System:** Design principles; Kernel modules; Process management; Scheduling	**08**	**CO5, CO6**

TEXT BOOKS:		
TB No.	**Author / Edition/ Publication / Year**	**Chapters**
1.	Abraham Silberschatz, Peter Baer Galvin, Greg Gagne: Operating System Concepts, 10th edition, Wiley India, 2018.	1, 2, 3, 4, 5, 6, 7, 8, 9, 10, 11, 13, 17, 20

REFERENCE BOOKS:		
RB No.	**Author / Edition/ Publication / Year**	**Chapters**
1.	D.M Dhamdhere: Operating systems - A concept based Approach, 3rd edition, McGraw Hill, 2013.	
2.	P.C.P. Bhatt: Introduction to Operating Systems: Concepts and Practice, 4th edition, PHI(EEE), 2014.	
3.	Harvey M Deital: Operating systems	
4.	William Stallings Operating Systems: Internals and Design Principles, 8th Edition, Pearson, 2014.	

APPLICATION DEVELOPMENT USING JAVA			
Course code:	**PCC21ISL46**	**Credits:**	01
L: T:P:	0:0:2	**CIE Marks:**	50
Exam Hours:	03	**SEE Marks:**	50
Total Hours:	12		

COURSE OBJECTIVES:	
1.	Get an idea of fundamental concepts of creating classes and objects in Java programming
2.	Demonstrate the basic principles of object oriented programming and usage of Constructors
3.	Understand the use and creation of packages and interfaces.
4.	Develop the applications using the concept of multi-threading, swings and string functionalities
5.	Analyze and use exception handling in java.

COURSE OUTCOMES: AT THE END OF THE COURSE, STUDENT WILL BE ABLE TO:	
CO1	Write java programs by creating objects , classes and methods
CO2	Apply the basic principles of object oriented programming
CO3	Design and use packages and interfaces
CO4	Analyze and implement the multi –threading and string functions
CO5	Develop an application to handle various exceptions in java
CO6	Construct the GUI based application using swings

MAPPING OF COURSE OUTCOMES TO PROGRAM OUTCOMES:														
	PO1	**PO2**	**PO3**	**PO4**	**PO5**	**PO6**	**PO7**	**PO8**	**PO9**	**PO10**	**PO11**	**PO12**	**PSO1**	**PSO2**	**PSO3**
CO1	3	2			2			1		1		2		1	
CO2	3	2	2		2			1		1		2		1	
CO3	3	2			2			1		1		2		1	
CO4	3	2			2			1		1		2		1	
CO5	2	2			2			1		1		2		2	
CO6	2	2	2		2							2		2	2

Expt. No.	**Contents of the experiment**	**Hours**	**CO's**
1.	Create a class Vehicle. The class should have two fields - no_of_seats, no_of_wheels and a method showVehicle. Create two objects-Motorcycle and Car for this class. Display the output to show the descriptions for Car and Motorcycle and demonstrate Constructor Overloading.	**2**	**CO1, CO2**
2.	Develop a Java program to read array elements (no sorted order) and compute the average value of an array of integers except the largest and smallest values.	**2**	**CO1, CO2**
3.	Incorporate the concept of package creation and demonstrate importing classes from user defined packages. Also record and process student details using multiple inheritance.	**2**	**CO1, CO3**
4.	Create a class MentorLog with Name, dialledNumber, duration, dialledDate. Read the No. of entries required, then those many values from User and sort the details using the sort method in Collection class (Implement Comparable Interface & Override compareTo method), based on duration.	**2**	**CO1, CO3**
5.	Demonstrate various forms of writing lambda expressions in java.	**2**	**CO1, CO2**

6.	Implement a multithreaded application that has two threads marked "even" and "odd". The threads cooperate to print the numbers from 1 to 20 in sequence with the "even" thread printing only even numbers and the "odd" thread printing only odd numbers.	**2**	**CO1, CO4**
7.	Create a custom Exception class. Consider two integer inputs which must be supplied by the user and display the sum of the integers if and only if the sum is less than 100. If it is not less than 100, throw the custom exception.	**2**	**CO1, CO5**
8.	Develop a Java Program to demonstrate different Mouse event types by displaying appropriate messages based on the event type.	**2**	**CO1, CO6**
9.	Write a Swing application which uses a) JTabbed Pane b) Each tab should use Jpanel which include any one component given below in each JPanel c) ComboBox/List/Tree/RadioButton	**2**	**CO1, CO6**
10.	Create a Java program to obtain a String and write it into a file (Ex: USN Value). Read the file contents and print the number of digits present in the file.	**2**	**CO1, CO6**

TEXT BOOKS:	
TB No.	**Author / Edition/ Publication / Year**
1.	The Complete Reference - Java: Herbert Schildt, 12th Edition, McGraw Hill, 2021.

REFERENCE BOOKS:	
RB No.	**Author / Edition/ Publication / Year**
1.	Y. Daniel Liang: Introduction to JAVA Programming, 10th Edition, Pearson Education, 2015.

PYTHON PROGRAMMING			
Course code:	**AEC21ISL481**	**Credits:**	01
L: P: T: S:	0: 0: 1: 0	**CIE Marks:**	50
Exam Hours:	1.5	**SEE Marks:**	50
Total Hours:	15		

COURSE OBJECTIVES:	
1.	Develop a basic Understanding of the Python Programming language
2.	Introduction the python language it's most important Data types, libraries, and its recommended programming styles and idioms.
3.	Methods to solve logical programming problems in python style. It also describes the some of the python programming environments that are available.
4.	Describing python programming environments that are available.

COURSE OUTCOMES: AT THE END OF THE COURSE, STUDENT WILL BE ABLE TO:	
CO1	Acquire knowledge of - Various types of data types, identifiers and keywords, Floating Point numbers, String operator and Methods.
CO2	Apply various concepts of Looping and control structures.
CO3	Ability to analyze collections of data types to illustrate the process of structuring the data using lists, tuples and dictionaries.
CO4	Analyze and implement the object oriented programming concepts in python.
CO5	Ability to demonstrate the use of built-in functions like numpy.
CO6	Design and apply appropriate Python Programming for solving computing problems.

MAPPING OF COURSE OUTCOMES TO PROGRAM OUTCOMES:

	PO1	PO2	PO3	PO4	PO5	PO6	PO7	PO8	PO9	PO10	PO11	PO12	PSO1	PSO2	PSO3
CO1	3	2	1	-	-	-	-	-	-	-	-	-	-	-	-
CO2	3	2	1	-	-	-	-	-	-	-	-	-	-	-	-
CO3	3	2	-	-	-	-	-	-	-	-	-	-	-	-	-
CO4	3	3	-	-	-	-	-	-	-	-	-	-	1	1	-
CO5	3	2	1	-	-	-	-	-	-	-	-	3	3	3	2
CO6	3	3	1	-	-	-	-	-	-	-	-	3	3	3	2

Module	Module Contents	Hours	CO's
1.	Write a program to demonstrate different number data types in Python.	**3**	**CO1, CO2**
2.	Write a program to create a menu with the following options: i) TO PERFORM ADDITITON ii) TO PERFORM SUBTRACTION iii) TO PERFORM MULTIPICATION iv) TO PERFORM DIVISION Accepts users input and perform the operation accordingly. Use functions with arguments.	**3**	**CO1, CO2**
3.	Write a program to create, concatenate and print a string and accessing sub-string from a given string.	**3**	**CO1, CO5**
4.	Using a numpy module create an array and check the following: i) Type of array ii) Axes of array iii) Shape of array iv) Type of elements in array	**3**	**CO1, CO5**

5.	Create a list and perform the following methods: i) insert() ii) remove() iii) append() iv) len() v) pop() vi) clear()	**3**	**CO3, CO6**
6.	Write a Python function that takes two lists and returns True if they are equal otherwise false	**3**	**CO3, CO6**
7.	Create a tuple and perform the following methods i) Add items ii) len() iii) Check for item in tuple iv) Access items	**3**	**CO3, CO6**
8.	Create a dictionary and apply the following methods: i) Print the dictionary items ii) Access items iii) Use get() iv) Change values v) Use len()	**3**	**CO3, CO6**
9.	Write a python Program to call data member and function using classes and objects.	**3**	**CO4, CO6**
10.	Write a program to find sum of two numbers using class and methods.	**3**	**CO4, CO6**

TEXT BOOKS:		
TB No.	**Author / Edition/ Publication / Year**	**Chapters**
1.	Mark Summerfield, Programming in Python 3, A complete introduction to the Python Language, Second Edition	2, 12
2.	Al Sweigart, "Automate the Boring Stuff with Python", 1st Edition, No Starch Press, 2015.	2, 3, 4, 5, 7, 8, 9, 10
3.	Allen B. Downey, "Think Python: How to Think Like a Computer Scientist", 2nd Edition, Green Tea Press, 2015	15, 16, 17, 18

REFERENCE BOOKS:		
RB No.	**Author / Edition/ Publication / Year**	**Chapters**
1.	Paul Gries, Jennifer Campbell, Jason Montojo, Practical Programming: An Introduction to Computer Science Using Python 3, Pragmatic Bookshelf, 2/E 2014	
2.	James Payne, Beginning Python: Using Python 2.6 and PYHTON 3, Wiley India 2010	

UNIX SHELL PROGRAMMING			
Course code:	**AEC21ISL482**	**Credits:**	1
L: T:P:	0:0:2	**CIE Marks:**	50
Exam Hours:	1.5	**SEE Marks:**	50
Total Hours:	12		

COURSE OBJECTIVES:	
1.	To gain ample understanding of UNIX shell scripts and its usage.
2.	To gain comprehensive knowledge about UNIX files and its operation.
3.	To understand the design principles and working of UNIX processes.
4.	To know the basic concepts of UNIX process attributes

COURSE OUTCOMES: AT THE END OF THE COURSE, STUDENT WILL BE ABLE TO:	
CO1	Apply the basic concepts of UNIX shell programming
CO2	Write, compile and execute UNIX shell programs
CO3	Develop scripts using UNIX shell
CO4	Implement basic programs using UNIX shell
CO5	Use shell scripts to write programs for UNIX file operations
CO6	Develop scripts to understand UNIX process attributes

MAPPING OF COURSE OUTCOMES TO PROGRAM OUTCOMES:

	PO1	**PO2**	**PO3**	**PO4**	**PO5**	**PO6**	**PO7**	**PO8**	**PO9**	**PO10**	**PO11**	**PO12**	**PSO1**	**PSO2**	**PSO3**
CO1	3	3	3	3	-	3	-	-	-	2	-	-	-	-	-
CO2	3	3	3	-	2	3	-	-	2	-	-	-	-	-	-
CO3	3	3	3	-	-	3	-	-	2	-	-	-	-	-	-
CO4	3	3	3	-	-	-	3	-	3	-	-	-	1	1	-
CO5	3	3	3	-	-	-	3	-	-	2	-	-	3	3	2
CO6	3	3	3	-	-	-	3	-	-	-	-	-	3	2	2

Expt. No	Contents of the experiment	Hours	CO's
1.	Design, develop, and execute a non-recursive shell script that accepts any number of arguments and prints them in the Reverse order, (For example, if the script is named rargs, then executing rargs A B C should produce C B A on the standard output).	**2**	**1, 2, 3, 4**
2.	Design, develop, and execute a shell script that accepts two file names as arguments, checks if the permissions for these files are identical and if the permissions are identical, outputs the common permissions, otherwise outputs each file name followed by its permissions.	**2**	**1, 2, 3, 4, 5**
3.	Design, develop, and execute a shell script that accepts file names specified as arguments and creates a shell script that contains this file as well as the code to recreate these files. Thus, if the script generated by your script is executed, it would recreate the original files.	**2**	**1, 2, 3, 4, 5**
4.	Design, develop, and execute a shell script that takes a valid directory name as an argument and recursively descend all the sub-directories, finds the maximum length of any file in that hierarchy and writes this maximum value to the standard output.	**2**	**1, 2, 3, 4, 5**
5.	Design, develop, and execute a shell script that accepts a path name and creates all the components in that path name as directories. For example, if the script is named mpc, then the command mpc a/b/c/d should create directories a, a/b, a/b/c, a/b/c/d.	**2**	**1, 2, 3, 4**
6.	Design, develop, and execute a shell script: a. To check whether the given number is even or odd. b. To search whether element is present is in the list or not.	**2**	**1, 2, 3, 4**
7.	Design, develop, and execute a shell script: a. To check whether given file is a directory or not. b. To count number of files in a Directory. c. To copy contents of one file to another.	**2**	**1, 2, 3, 4, 5**
8.	Design, develop, and execute a shell script: a. To create a directory, write contents on that and copy to a suitable location in your home directory. b. To use a pipeline and command substitution to set the length of a line in file to a variable.	**2**	**1, 2, 3, 4, 6**
9.	Design, develop, and execute a shell script: a. To display the process attributes.	**2**	**1, 2, 3, 4, 6**

	b. To change the priority of processes. c. To change the ownership of processes.		
10.	Design, develop, and execute a shell script that accepts one or more filenames as argument and converts all of them to uppercase, provided they exist in current directory.	**2**	**1, 2, 3, 4, 5**

TEXT BOOKS:		
TB No.	**Author / Edition/ Publication / Year**	**Chapters**
1.	Sumitabha Das: UNIX – Concepts and Applications, 4th edition, Tata McGraw Hill, 2006	
2.	Terrence Chan: UNIX System Programming Using C++, Prentice Hall India, 1999	1, 5, 6, 7, 8, 9, 10
3.	W. Richard Stevens: Advanced Programming in the UNIX Environment, 3rd Edition, Pearson Education, 2015	7, 8, 9, 13, 14, 15

REFERENCE BOOKS:		
RB No.	**Author / Edition/ Publication / Year**	**Chapters**
1.	Marc J. Rochkind: Advanced UNIX Programming, 2nd Edition, Pearson Education, 2005.	
2.	Maurice J Bach: The Design of the UNIX Operating System, Pearson Education, 1987.	
3.	Uresh Vahalia: UNIX Internals: The New Frontiers, Pearson Education, 2001.	

UNIVERSAL HUMAN VALUES

Course Code: 21UH49 **Credits: 01**
L: P: T: S: 0: 0: 1: 0 **CIE Marks: 50**
Exam Hours: 03 **SEE Marks: 50**
Total Hours: 15

COURSE OBJECTIVES:

1. Recognize the need for harmony in society.
2. Understand the underlying the propositions of harmony in society and nature.
3. Verify the propositions and realise the implications of holistic understanding and professional ethics

COURSE OUTCOMES: AFTER COMPLETION OF THE COURSE, THE GRADUATES WILL BE ABLE TO	
CO1	Recognize the importance of harmony in society.
CO2	Understand the concepts of harmony in nature
CO3	Develop skills for living in harmony and apply in professional life
CO4	Outline the principles of human values and examine its role in education
CO5	Adapt to natural acceptance of human values build holistic understanding
CO6	Test and verify the propositions on human values by self exploration

Mapping of Course outcomes to Program outcomes:

	PO1	**PO2**	**PO3**	**PO4**	**PO5**	**PO6**	**PO7**	**PO8**	**PO9**	**PO10**	**PO11**	**PO12**
CO1	-	-	-	-	-	3	3	2	2	2	-	2
CO2	-	-	-	-	-	3	3	2	2	2	-	2
CO3	-	-	-	-	-	3	3	2	2	2	-	2
CO4	-	-	-	-	-	3	3	2	2	2		2
CO5	-	-	-	-	-	3	3	2	2	2		2
CO6	-	-	-	-	-	3	3	2	2	2		2

Unit	Course Content	Hours	COs
1	**Harmony in the society** Understanding the harmony in the society (society being an extension of family): Samadhana, Samriddhi, Abhaya, Sah-astitva as comprehensive Human Goals, Visualizing a universal harmonious order in society-Undivided Society (AkhandSamaj), Universal Order (SarvabhaumaVyavastha)-**, Case studies & Practice sessions.**	**03**	**CO1 CO2**
2	**Harmony in the Nature (Existence)-I:** Understanding the harmony in the Nature, Interconnectedness and mutual fulfillment among the four orders of nature. **Case studies & Practice sessions.**	**03**	**CO1 CO2**
3	**Harmony in the Nature (Existence)-II** Understanding existence as co-existence (Sah-astitva) of mutually interacting units in all-pervasive space, **Case studies & Practice sessions.**	**03**	**CO3 CO4**
4	**Implications of the Holistic Understanding: Natural** acceptance of human values, Definitiveness of Ethical Human Conduct, Basis for Humanistic Education, **Case studies & Practice sessions.**	**03**	**CO3 CO4**
5	**A Look at Professional Ethics:** Competence in Professional Ethics: Ability to utilize the professional competence for augmenting universal human order, Ability to identify the scope and characteristics of people-friendly and eco=friendly production systems, technologies and management models, At the level of individual: as socially and ecologically responsible engineers, technologists and managers and at the level of society. **Case studies & Practice sessions.**	**03**	**CO5 CO6**

SELF STUDY COMPONENT:

Unit - 1: From family to world family.
Unit - 2: Recyclability and self-regulation in nature.
Unit - 3: Holistic perception of harmony at all levels of existence.
Unit - 4: Humanistic Constitution and Universal Human Order.
Unit - 5: mutually enriching institutions and organizations.

TEXT BOOKS:

1. A foundation course in HUMAN VALUES and Professional ethics; presenting a universal approach to value education- through self-exploration by, R R Gaur, R Sangal& GP Bagaria, Excel books Pvt. Ltd.

2. Professional Ethics & Human Values: Prof. D.R. Kiran, TATA McGraw Hill Education

REFERENCE BOOKS:

1. Human Values: A. N. Tripathy (2003, New Age International Publishers)
2. Ethics in Engineering Mike W. Martin, Department of Philosophy, Chapman University and Roland Schinzinger, School of Engineering, University of California, Irvine.
3. Fundamentals of Ethics, Edmond G. Seebauer & Robert L. Barry, Oxford University Press.

Assessment Pattern:

CIE –Continuous Internal Evaluation Theory (50 Marks)

Bloom's Category	Tests	Quiz	Practice session/Assignment
Marks (Out of 50)	**30**	**10**	**10**
Remember	--	01	01
Understand	10	02	02
Apply	10	02	02
Analyze	05	02	02
Evaluate	05	02	02
Create		01	01

SEE –Semester End Examination Theory (50 Marks)

Bloom's Category	Marks Theory(50)
Remember	10
Understand	10
Apply	10
Analyze	10
Evaluate	10
Create	

DISCRETE MATHEMATICAL STRUCTURES

Course Code: AEC21IS483 **Credits: 01**
L: P: T: S: 1: 0: 0: 0 **CIE Marks: 50**
Exam Hours: 03 **SEE Marks: 50**
Total Hours: 15

COURSE OBJECTIVES:

1. To understand the fundamental mathematical concepts namely sets, relations, functions, logic and integers.
2. To understand the laws of logic and simplify logical statements with many connectives.
3. To analyze the basic concepts of principle of inclusion and exclusion

COURSE OUTCOMES: AFTER COMPLETION OF THE COURSE, THE GRADUATES WILL BE ABLE TO	
CO1	Get familiarity with set logic and mathematical logic
CO2	Calculate the number of possible outcomes of elementary combinatorial process such as permutations and combinations
CO3	Evaluate the validity of a given argument by applying laws of logic
CO4	Apply and analyze the basic knowledge gained by functions and relations
CO5	Acquire the clarity of identifying the types of functions and apply pigeon-hole principle in various problems solving situation
CO6	Determine solutions to mathematical problems using principle of inclusion and exclusion

Mapping of Course outcomes to Program outcomes:

	PO1	PO2	PO3	PO4	PO5	PO6	PO7	PO8	PO9	PO10	PO11	PO12	PSO1	PSO2	PSO3
CO1	3	3	2	1	-	-	-	-	-	-	-	2	3	2	-
CO2	3	3	2	1	-	-	-	-	-	-	-	2	3	2	-
CO3	3	3	2	1	-	-	-	-	-	-	-	2	3	2	-
CO4	3	3	2	1	-	-	-	-	-	-	-	2	3	2	-
CO5	3	3	2	1	-	-	-	-	-	-	-	2	3	2	-
CO6	3	3	2	1	-	-	-	-	-	-	-	2	3	2	-

Unit	Course Content	Hours	COs
1	**Set Theory :** Set Operations and the Laws of Set Theory, Addition Principle, Probability	**03**	**CO1 CO2**
2	**Fundamentals Of Logic**: Basic Connectives and Truth Tables, Logical Equivalence: The laws of Logic	**03**	**CO3**
3	**Relations And Functions**: Cartesian Products and Relations, Functions, Types of functions, The Pigeonhole Principle	**03**	**CO4 CO5**
4	**Relations**: Zero-one matrices and directed graphs, Properties of Relations, Equivalence Relations, Partial Orders – Hasse Diagrams	**03**	**CO4 CO5**
5	**Principles Of Counting**: Principle of Inclusion and Exclusion, Derangements, Rook Polynomials.	**03**	**CO6**

SELF STUDY COMPONENT:

Unit-1**:** The Well Ordering Principle - Mathematical Induction
Unit-2: Use of Quantifiers
Unit-3: Stirling Numbers of the Second Kind
Unit-4: Group Theory
Unit-5: Binomial and Multinomial Theorem, Catalan Numbers

TEXT BOOKS:

1. Ralph P. Grimaldi: Discrete and Combinatorial Mathematics, 5th Edition, Pearson Education, 2004.

2. Dr. D.S.C, Discrete Mathematical Structures, 3rd Edition, PRISM

REFERENCE BOOKS:

1. Kenneth H. Rosen: Discrete Mathematics and its Applications, 7th Edition, McGraw Hill, 2010.
2. Jayant Ganguly: A Treatise on Discrete Mathematical Structures, Sanguine-Pearson, 2010

INTRODUCTION TO JAVA

Course Code: AEC21IS484 **Credits: 01**
L: P: T: S: 0: 0: 1: 0 **CIE Marks: 50**
Exam Hours: 03 **SEE Marks: 50**
Total Hours: 15

COURSE OBJECTIVES:

1. To provide an introduction to java and object-oriented concepts of java programming.
2. Understand the use and creation of packages and interfaces.
3. Analyze and use exception handling in java.
4. A better understanding of string libraries.

COURSE OUTCOMES: AFTER COMPLETION OF THE COURSE, THE GRADUATES WILL BE ABLE TO	
CO1	Articulate classes, its members and the relationships among them needed for a specific problem.
CO2	Apply the basic concepts of object oriented programming in writing java programs.
CO3	Create and use packages/ interfaces in Java programs.
CO4	Analyze and implement exception handling in Java.
CO5	Use various Input/output packages effectively.
CO6	Design programs using String libraries.

Mapping of Course outcomes to Program outcomes:

	PO1	PO2	PO3	PO4	PO5	PO6	PO7	PO8	PO9	PO10	PO11	PO12
CO1	2	2	1	-	-	-	-	-	-	-	-	1
CO2	3	2	2	-	-	-	-	-	-	-	-	1
CO3	3	2	2	-	-	-	-	-	-	-	-	1
CO4	2	2	1	-	-	-	-	-	-	-	-	1
CO5	2	2	2	-	-	-	-	-	-	-	-	1
CO6	3	2	2	-	-	-	-	-	-	-	-	1

Module	Course Content	Hours	COs
1	**Introduction**: Creation of Java, Byte code, Java Buzzwords, Object Oriented Programming, A simple program, Two Control statements, Lexical Issues, Type conversion and casting, Arrays. **Operators:** Arithmetic operators, Bitwise operators, Relational operators, the assignment operator, The ? Operator, operator precedence. Control Statements: Selection statements, iteration statements, Jump statements.	**03**	**CO1 CO2**
2	**Classes:** Class fundamentals, declaring Objects, assigning object reference variables, introducing methods, constructors, this keyword, garbage collection, the finalize() method. **A Closer Look at Methods and Classes:** Overloading methods, using objects as parameters, returning objects, introducing access control, understanding static, introducing final.	**03**	**CO1 CO2**
3	**Inheritance:** inheritance basics, using super, creating multilevel hierarchy, method overriding, using abstract classes, using final with inheritance. **Interfaces:** Defining an Interface, Implementing Interface, Applying interfaces, Variables in interfaces. **Exception handling:** Fundamentals, Exception types, using try and catch, nested try statements, throw, throws, finally.	**03**	**CO3 CO4**

4	**Input / Output:** I/O Basics, Reading Console Input, Writing Console Output, Reading and Writing files. **Packages:** Defining a package, Access protection; importing packages.	**03**	**CO3** **CO4**
5	**String handling:** String Constructors, String Length, Special string operators, Character extraction, String comparison, Searching Strings, Modifying a string.	**03**	**CO5** **CO6**

SELF STUDY COMPONENT:

Module 1: Data types and other tokens: Boolean Variables, int, long, char, white spaces, literals.
Module 2: nested and inner classes, using command line arguments
Module 3: Interfaces can be extended, multiple catch clauses, Java's built-in exceptions
Module 4: the Print Writer Class,
Module 5: StringBuffer, StringBuilder

TEXT BOOKS:

1. Herbert Schildt: Java The Complete Reference, 7th Edition, Tata McGraw Hill, 2007.

REFERENCE BOOKS:

1. Y. Daniel Liang: Introduction to JAVA Programming, 6th Ed, Pearson Education, 2007.