MARTHANDAM COLLEGE OF ENGINEERING AND TECHNOLOGY

DEPARTMENT OF COMPUTER SCIENCE AND ENGINEERING

COMPUTER LAB III

Equipment's Available in the Lab

Sl.No	Hardware	Specification	Quantity
1	Desktops	HCL Desktop Intel Dual Core Processor Intel 31 Chipset Motherboard 1GB DDR2 RAM 160GB SATA HDD 15" TFT LCD Monitor HCL Multimedia Keyboard and Optical Mouse	55 Nos
Software			
1	Operating Systems: Linux		
2	Front End Tools: Code blocks, FLUX,BISON		
3	Geany, Pycharm, Code Blocks Network simulator like NS2		
4	ArgoUML that supports UML 1.4 and higher		
5	PHP and MySQL Selenium/TestNG		

MARTHANDAM COLLEGE OF ENGINEERING AND TECHNOLOGY

DEPARTMENT OF COMPUTER SCIENCE AND ENGINEERING

COURSES OFFERED

Sl.No	Odd Sem (Course code & Name)	Class	Even Sem (Course code & Name)	Class
1	CD3281 Data structures and Algorithms Laboratory	II IT	CS3271 Programming in C laboratory	I CSE I IT
2	CS3362 C Programming &DatastructuresLaboratory	II EEE	EC3401 Networks and Security	II-ECE
3	EC8563 Communication Networks Laboratory	III ECE	CCS354 Network Security	III-CSE
4	GE3171 Problem Solving And Python Programming Laboratory	I Year	CS3461 Operating Systems Laboratory	II-CSE
5	CCS366 Software Testing and Automation	III CSE III IT	CS3401Algorithms	II CSE
6	CS3501 Compiler Design	III CSE		

CD3281 DATA STRUCTURES AND ALGORITHMS LABORATORY

OBJECTIVES:

- To implement ADTs in Python
- To design and implement linear data structures – lists, stacks, and queues
- To implement sorting, searching and hashing algorithms
- To solve problems using tree and graph structures

OUTCOMES:

- Implement ADTs as Python classes
- Design, implement, and analyse linear data structures, such as lists, queues, and stacks, according to the needs of different applications
- Design, implement, and analyse efficient tree structures to meet requirements such as searching, indexing, and sorting
- Model problems as graph problems and implement efficient graph algorithms to solve them

LIST OF EXPERIMENTS

1. Implement simple ADTs as Python classes
2. Implement recursive algorithms in Python
3. Implement List ADT using Python arrays
4. Linked list implementations of List
5. Implementation of Stack and Queue ADTs
6. Applications of List, Stack and Queue ADTs
7. Implementation of sorting and searching algorithms
8. Implementation of Hash tables
9. Tree representation and traversal algorithms
10. Implementation of Binary Search Trees
11. Implementation of Heaps
12. Graph representation and Traversal algorithms
13. Implementation of single source shortest path algorithm
14. Implementation of minimum spanning tree algorithms

CS3362 C PROGRAMMING & DATASTRUCTURES LABORATORY

OBJECTIVES:

- To develop applications in C
- To implement linear and non-linear data structures
- To understand the different operations of search trees
- To get familiarized to sorting and searching algorithms

OUTCOMES:

- Use different constructs of C and develop applications
- Write functions to implement linear and non-linear data structure operations
- Suggest and use the appropriate linear / non-linear data structure operations for a given problem
- Apply appropriate hash functions that result in a collision free scenario for data storage and Retrieval
- Implement Sorting and searching algorithms for a given application

LIST OF EXPERIMENTS

1.Practice of C programming using statements, expressions, decision making and iterative statements

2. Practice of C programming using Functions and Arrays
3. Implement C programs using Pointers and Structures
4. Implement C programs using Files
5. Development of real time C applications
6. Array implementation of List ADT
7. Array implementation of Stack and Queue ADTs
8. Linked list implementation of List, Stack and Queue ADTs
9. Applications of List, Stack and Queue ADTs
10. Implementation of Binary Trees and operations of Binary Trees
11. Implementation of Binary Search Trees
12. Implementation of searching techniques
13. Implementation of Sorting algorithms : Insertion Sort, Quick Sort, Merge Sort
14. Implementation of Hashing – any two collision techniques

CS3271 PROGRAMMING IN C LABORATORY

OBJECTIVES:

- To familiarise with C programming constructs.
- To develop programs in C using basic constructs.
- To develop programs in C using arrays.
- To develop applications in C using strings, pointers, functions.
- To develop applications in C using structures.
- To develop applications in C using file processing.

OUTCOMES:

Upon Completion of the course, the students will be able to:

- Demonstrate knowledge on C programming constructs.
- Develop programs in C using basic constructs.
- Develop programs in C using arrays.
- Develop applications in C using strings, pointers, functions.
- Develop applications in C using structures.
- Develop applications in C using file processing.

LIST OF EXPERIMENTS

1. I/O statements, operators, expressions
2. decision-making constructs: if-else, goto, switch-case, break-continue
3. Loops: for, while, do-while
4. Arrays: 1D and 2D, Multi-dimensional arrays, traversal
5. Strings: operations
6. Functions: call, return, passing parameters by (value, reference), passing arrays to function.

7. Recursion
8. Pointers: Pointers to functions, Arrays,Strings, Pointers to Pointers, Array of Pointers
9. Structures: Nested Structures, Pointers to Structures, Arrays of Structures and Unions.
10. Files: reading and writing, File pointers, file operations, random access, processor directives

EC8563 COMMUNICATION NETWORKS LABORATORY

OBJECTIVES:

- Learn to communicate between two desktop computers
- Learn to implement the different protocols
- Be familiar with IP Configuration
- Be familiar with the various routing algorithms
- Be familiar with simulation tools

OUTCOMES:

- Communicate between two desktop computers
- Implement the different protocols
- Program using sockets.
- Implement and compare the various routing algorithms
- Use the simulation tool.

LIST OF EXPERIMENTS

1. Implementation of Error Detection / Error Correction Techniques
2. Implementation of Stop and Wait Protocol and sliding window
3. Implementation and study of Goback-N and selective repeat protocols
4. Implementation of High Level Data Link Control
5. Implementation of IP Commands such as ping, Traceroute, nslookup.
6. Implementation of IP address configuration.
7. To create scenario and study the performance of network with CSMA / CA protocol
and compare with CSMA/CD protocols.
8. Network Topology - Star, Bus, Ring
9. Implementation of distance vector routing algorithm
10. Implementation of Link state routing algorithm
11. Study of Network simulator (NS) and simulation of Congestion Control Algorithms
using NS
12. Implementation of Encryption and Decryption Algorithms using any programming
language

GE3171 PROBLEM SOLVING AND PYTHON PROGRAMMING LABORATORY

OBJECTIVES:

- To understand the problem solving approaches.
- To learn the basic programming constructs in Python.
- To practice various computing strategies for Python-based solutions to real world problems
- To use Python data structures - lists, tuples, dictionaries.
- To do input/output with files in Python

OUTCOMES:

- Develop algorithmic solutions to simple computational problems
- Develop and execute simple Python programs.
- Implement programs in Python using conditionals and loops for solving problems.
- Deploy functions to decompose a Python program.
- Process compound data using Python data structures.
- Utilize Python packages in developing software applications.

LIST OF EXPERIMENTS

1. Solving of simple real life or scientific or technical problems, and developing flow charts .
2. Python programming using simple statements and expressions
3. Scientific problems using Conditionals and Iterative loops.
4. Real-time/technical applications using Lists, Tuples.
5. Real-time/technical applications using Sets, Dictionaries.
6. Programs using Functions
7. Programs using Strings.
8. Programs using written modules and Python Standard Libraries
9. Real-time/technical applications using File handling.
10. Real-time/technical applications using Exception handling.
11. Exploring Pygame tool.
12. Developing a game activity using Pygame like bouncing ball, car race etc.

CCS366 SOFTWARE TESTING AND AUTOMATION

COURSE OBJECTIVES:

- To understand the basics of software testing .
- To learn how to do the testing and planning effectively .
- To build test cases and execute them .
- To focus on wide aspects of testing and understanding multiple facets of testing .
- To get an insight about test automation and the tools used for test automation.

OUTCOMES:

At the end of this course, the students will be able to:

CO1: Understand the basic concepts of software testing and the need for software testing
CO2: Design Test planning and different activities involved in test planning.
CO3: Design effective test cases that can uncover critical defects in the application.
CO4: Carry out advanced types of testing.
CO5: Automate the software testing using Selenium and TestNG.

LIST OF EXPERIMENTS

1. Develop the test plan for testing an e-commerce web/mobile application (www.amazon.in).
2. Design the test cases for testing the e-commerce application.
3. Test the e-commerce application and report the defects in it.
4. Develop the test plan and design the test cases for an inventory control system.
5. Execute the test cases against a client server or desktop application and identify the defects.
6. Test the performance of the e-commerce application.
7. Automate the testing of e-commerce applications using Selenium.
8. Integrate TestNG with the above test automation.
9. Mini Project:
   a) Build a data-driven framework using Selenium and TestNG
   b) Build Page object Model using Selenium and TestNG
   c) Build BDD framework with Selenium, TestNG and Cucumber

CS3401 ALGORITHMS

COURSE OBJECTIVES:

- To understand and apply the algorithm analysis techniques on searching and sorting algorithms
- To critically analyze the efficiency of graph algorithms
- To understand different algorithm design techniques
- To solve programming problems using state space tree
- To understand the concepts behind NP Completeness, Approximation algorithms and randomized algorithms.

COURSE OUTCOMES:

At the end of this course, the students will be able to:
CO1: Analyze the efficiency of algorithms using various frameworks
CO2: Apply graph algorithms to solve problems and analyze their efficiency.
CO3: Make use of algorithm design techniques like divide and conquer, dynamic programming and greedy techniques to solve problems
CO4: Use the state space tree method for solving problems.
CO5: Solve problems using approximation algorithms and randomized algorithms

LIST OF EXPERIMENTS

Searching and Sorting Algorithms

1. Implement Linear Search. Determine the time required to search for an element. Repeat the experiment for different values of n, the number of elements in the list to be searched and plot a graph of the time taken versus n.
2. Implement recursive Binary Search. Determine the time required to search an element. Repeat the experiment for different values of n, the number of elements in the list to be searched and plot a graph of the time taken versus n.
3. Given a text txt [0...n-1] and a pattern pat [0...m-1], write a function search (char pat [ ], char txt [ ]) that prints all occurrences of pat [ ] in txt [ ]. You may assume that n > m.
4. Sort a given set of elements using the Insertion sort and Heap sort methods 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.

Graph Algorithms

1. Develop a program to implement graph traversal using Breadth First Search
2. Develop a program to implement graph traversal using Depth First Search
3. From a given vertex in a weighted connected graph, develop a program to find the shortest paths to other vertices using Dijkstra's algorithm.
4. Find the minimum cost spanning tree of a given undirected graph using Prim's algorithm.
5. Implement Floyd's algorithm for the All-Pairs- Shortest-Paths problem.
6. Compute the transitive closure of a given directed graph using Warshall's algorithm.

Algorithm Design Techniques

1.Develop a program to find out the maximum and minimum numbers in a given list of n
numbers using the divide and conquer technique.

2.Implement Merge sort and Quick sort methods to sort an array of elements and determine the time required to sort. 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.

State Space Search Algorithms

1.Implement N Queens problem using Backtracking.

Approximation Algorithms Randomized Algorithms

1.Implement any scheme to find the optimal solution for the Traveling Salesperson problem and then solve the same problem instance using any approximation algorithm and determine the error in the approximation.

2.Implement randomized algorithms for finding the kth smallest number. The programs can be implemented in C/C++/JAVA/ Python.

MARTHANDAM COLLEGE OF ENGINEERING AND TECHNOLOGY

DEPARTMENT OF COMPUTER SCIENCE AND ENGINEERING

EC3401 NETWORKS AND SECURITY

COURSE OBJECTIVES:

- To learn the Network Models and datalink layer functions.
- To understand routing in the Network Layer.
- To explore methods of communication and congestion control by the Transport Layer.
- To study the Network Security Mechanisms.
- To learn various hardware security attacks and their countermeasures.

OUTCOMES:

CO1: Explain the Network Models, layers and functions.

CO2: Categorize and classify the routing protocols.

CO3: List the functions of the transport and application layer.

CO4: Evaluate and choose the network security mechanisms.

CO5: Discuss the hardware security attacks and countermeasures.

LIST OF EXPERIMENTS

1. Implement the Data Link Layer framing methods,

i) Bit stuffing, (ii) Character stuffing

2. Implementation of Error Detection / Correction Techniques i) LRC, (ii) CRC, (iii) Hamming code

3. Implementation of Stop and Wait, and Sliding Window Protocols

4. Implementation of Go back-N and Selective Repeat Protocols.

5. Implementation of Distance Vector Routing algorithm (Routing Information Protocol) (Bellman-Ford).

6. Implementation of Link State Routing algorithm (Open Shortest Path First) with 5 nodes (Dijkstra's).

7. Data encryption and decryption using Data Encryption Standard algorithm.

8. Data encryption and decryption using RSA (Rivest, Shamir and Adleman) algorithm.

9. Implement Client Server model using FTP protocol.

MARTHANDAM COLLEGE OF ENGINEERING AND TECHNOLOGY

DEPARTMENT OF COMPUTER SCIENCE AND ENGINEERING

CCS354 NETWORK SECURITY

OBJECTIVES:

- To learn the fundamentals of cryptography.
- To learn the key management techniques and authentication approaches.
- To explore the network and transport layer security techniques.
- To understand the application layer security standards.
- To learn the real time security practices.

OUTCOMES:

CO1:Classify the encryption techniques

CO2:Illustrate the key management technique and authentication.

CO3:Evaluate the security techniques applied to network and transport layer

CO4: Discuss the application layer security standards.

CO5:Apply security practices for real time applications.

LIST OF EXPERIMENTS

1. Implement symmetric key algorithms
2. Implement asymmetric key algorithms and key exchange algorithms
3. Implement digital signature schemes
4. Installation of Wire shark, tcpdump and observe data transferred in client-server communication using UDP/TCP and identify the UDP/TCP datagram.
5. Check message integrity and confidentiality using SSL
6. Experiment Eavesdropping, Dictionary attacks, MITM attacks
7. Experiment with Sniff Traffic using ARP Poisoning
8. Demonstrate intrusion detection system using any tool.
9. Explore network monitoring tools
10. Study to configure Firewall, VPN directives.

MARTHANDAM COLLEGE OF ENGINEERING AND TECHNOLOGY

DEPARTMENT OF COMPUTER SCIENCE AND ENGINEERING

CS3461 OPERATING SYSTEMS LABORATORY

COURSE OBJECTIVES:

- To install windows operating systems.
- To understand the basics of Unix command and shell programming.
- To implement various CPU scheduling algorithms.
- To implement Deadlock Avoidance and Deadlock Detection Algorithms
- To implement Page Replacement Algorithms .
- To implement various memory allocation methods.
- To be familiar with File Organization and File Allocation Strategies

OUTCOMES:

At the end of this course, the students will be able to:

CO1 : Define and implement UNIX Commands.

CO2 : Compare the performance of various CPU Scheduling Algorithms.

CO3 : Compare and contrast various Memory Allocation Methods.

CO4 : Define File Organization and File Allocation Strategies.

CO5 : Implement various Disk Scheduling Algorithms.

LIST OF EXPERIMENTS

1. Installation of windows operating system.
2. Illustrate UNIX commands and Shell Programming.
3. Process Management using System Calls : Fork, Exit, Getpid, Wait, Close.
4. Write C programs to implement the various CPU Scheduling Algorithms.
5. Illustrate the inter process communication strategy.
6. Implement mutual exclusion by Semaphore .
7. Write C programs to avoid Deadlock using Banker's Algorithm.
8. Write a C program to Implement Deadlock Detection Algorithm.
9. Write C program to implement Threading.
10. Implement the paging Technique using C program.
11. Write C programs to implement the following Memory Allocation Methods
a. First Fit b. Worst Fit c. Best Fit
12. Write C programs to implement the various Page Replacement Algorithms.
13. Write C programs to Implement the various File Organization Techniques
14. Implement the following File Allocation Strategies using C programs
a. Sequential b. Indexed c. Linked
15. Write C programs for the implementation of various disk scheduling algorithms.
16. Install any guest operating system like Linux using VMware.

MARTHANDAM COLLEGE OF ENGINEERING AND TECHNOLOGY

DEPARTMENT OF COMPUTER SCIENCE AND ENGINEERING

CS3501 COMPILER DESIGN

OBJECTIVES:

- To learn the various phases of compiler.
- To learn the various parsing techniques.
- To understand intermediate code generation and run-time environment
- To learn to implement the front-end of the compiler and learn to implement code generator
- To learn to implement code optimization.

OUTCOMES:

CO1:Understand the techniques in different phases of a compiler.

CO2:Design a lexical analyser for a sample language and learn to use the LEX tool.

CO3:Apply different parsing algorithms to develop a parser and learn to use YACC tool
CO4:Understand semantics rules (SDT), intermediate code generation and run-time environment.

CO5:Implement code generation and apply code optimization techniques.

LIST OF EXPERIMENTS

1. Using the LEX tool, Develop a lexical analyzer to recognize a few patterns in C. (Ex. identifiers, constants, comments, operators etc.). Create a symbol table, while recognizing identifiers.
2. Implement a Lexical Analyzer using LEX Tool
3. Generate YACC specification for a few syntactic categories.
    a. Program to recognize a valid arithmetic expression that uses operator +, -, * and /.
    b. Program to recognize a valid variable which starts with a letter followed by any number of letters or digits.
    c. Program to recognize a valid control structures syntax of C language (For loop, while loop, if-else, if-else-if, switch-case, etc.).
    d. Implementation of calculator using LEX and YACC
4. Generate three address code for a simple program using LEX and YACC.
5. Implement type checking using Lex and Yacc.
6. Implement simple code optimization techniques (Constant folding, Strength reduction and Algebraic transformation)
7. Implement back-end of the compiler for which the three address code is given as input and the 8086 assembly language code is produced as output.