Proposed Syllabus for M.Sc. (Computer Science) in affiliated colleges to University of Pune

(To be implemented from Academic year 2011-2012)

Course Structure – The entire course is a Two year and Four semester full time course. For three semesters there will be four theory courses and one Lab course. The last semester will be Industrial training/Institutional project. The Lab course of semester I and semester III and one theory course each from semester II and Semester III are departmental courses.

Assessment Pattern -

Every theory paper is evaluated for 80 marks externally through an end semester examination and for 20 marks internally, the pattern of internal evaluation is through a mid-semester examination.

Every departmental theory course is internally evaluated for 100 marks; the pattern of evaluation should include one end semester examination (40 marks), one mid semester examination (30 marks) and assignments (30 marks). For few of the elective courses the evaluation pattern will be as indicated along with the outline.

Every lab course is divided into project work and assignments related to theory subjects and the break up are given along with each of the lab course.

The Industrial Project will be graded. The grades are O, A+, A, B+, B, C+ , C and D. D grade indicates failure

		Semester 1
CS-101(New)	:	Principles of Programming Languages
CS-102(New)	:	Advanced Networking
CS-103(New)	:	Distributed Database Concepts
CS-104(New)	:	Design and Analysis of Algorithms
CS-105(New)	:	Laboratory Course (Departmental)
		Semester 2
CS-201(New)	:	Digital Image Processing
CS-202(New)	:	Advanced Operating Systems
CS-203(New)	:	Data Mining and Data Warehousing
CS-204(New)	:	Elective Course(Departmental)
CS-205(New)	:	Laboratory Course (University)
		Semester 3
CS-301(New)	:	Software Metrics & Project Management
CS-302(New)	:	Mobile Computing
CS-303(New)	:	Soft Computing
CS-304(New)	:	Elective Course(Departmental)
CS-305(New)	:	Laboratory course (Departmental)
		Semester 4
CS-401(New)	:	Full time Industrial Training

Important information regarding the CS-204(New)/CS-304(New): Elective Course

Please note that, one elective from the following list to be opted for each of the semesters 2nd and 3rd (CS-204 in semester two and CS-304 in semester three respectively) according to prerequisite conditions (if any).

List of Elective Courses

1. Advanced Algorithms
2. Functional Programming
3. Linux Kernel Programming and Introduction to Device Drivers
4. Natural Language Processing
5. Program Analysis
6. DOT NET
7. Information Systems Security
8. Software Architecture and Design Patterns
9. Software Testing Tools & Methodologies
10. Modeling and Simulations
11. Embedded System Programming
12. Language Processors
13. Artificial Intelligence

M.Sc. (Computer Science)
First Year Semester 1

CS-101(New): Principles of Programming Languages

[Total Lectures: 48 Hours]

Course Prerequisites:

It is assumed that student learning this course have the following background:

- Experience with an OOP language (such as Java or C++)
- Experience with a procedural language (such as C)
- Working knowledge of C, C++, and Java programming.
- Basic algorithms and data structure concepts.

Why to study this course?

- To allow Informed Design Decisions
- Gives insight when debugging
- Permits effective use of compilers/linkers interpreters and language oriented tools.
- Helps to understand how language features work.
- Learn features, emulate missing features.
- Develop a greater understanding of the issues involved in programming language design and implementation
- Develop an in-depth understanding of functional, logic, and object-oriented programming paradigms
- Implement several programs in languages other than the one emphasized in the core curriculum (Java/C++)
- Understand design/implementation issues involved with variable allocation and binding, control flow, types, subroutines, parameter passing
- Develop thorough understanding of the compilation process
- To introduce several different paradigms of programming
- To gain experience with these paradigms by using example programming languages
- To understand concepts of syntax, translation, abstraction, and implementation

Course Objectives:

- This course will prepare you to think about programming languages analytically:
    - Separate syntax from semantics
    - Compare programming language designs
    - Learn new languages more quickly
    - Use standard vocabulary when discussing languages
    - Understand basic language implementation techniques

- This course focuses on both:
    - Theory is covered by the textbook readings, lectures, and on the tests
    - Implementation is covered by the homework assignments

1. Introduction [T1 chap. 1] [2]

- The Art of Language Design [T1 1.1]
- The Programming Language Spectrum [T1 1.2]
- Why Study Programming Languages? [T1 1.3]
- Compilation and Interpretation [T1 1.4]
- Programming Environments [T1 1.5]

2. Non-Imperative Programming Models: Functional, Logic Languages

[Text books 3, 4] **[10]**

Common LISP

- Basic LISP Primitives (FIRST, REST, SETF, CONS, APPEND, LIST, NTHCDR, BUTLAST,LAST, LENGTH, REVERSE, ASSOC)
- Procedure definition and binding, DEFUN, LET
- Predicates and Conditional,
  EQUAL, EQ, EQL, =, MEMBER, LISTP, ATOM, NUMBERP, SYMBOLP, NIL, NULL, IF, WHEN, UNLESS, COND, CASE
- Procedure Abstraction and Recursion
  [**T 4**]

Turbo Prolog

Introduction, facts, Objects and Predicates, Variables, Using Rules, Controlling execution fail and cut predicates

[**T3** chapter 1 through 9 except chapter 2]

3. Names, Scopes, and Bindings [T1 chap.3] [5]

The Notion of Binding Time [T1 chap.3.1]

Object Lifetime and Storage Management : [T1 chap. 3.2]

Static Allocation, Stack-Based Allocation, Heap-Based Allocation, Garbage Collection

Scope Rules [T1 chap. 3.3]

Static Scoping, Nested Subroutines, Declaration Order, Dynamic Scoping

The meaning of Names in a Scope [T1 chap. 3.5]

Aliases, Overloading, Polymorphism and Related Concepts

The Binding of Referencing Environments [T1 chap. 3.6]

Subroutine Closures, First-Class Values and Unlimited Extent, Object Closures

Macro Expansion [T1 chap. 3.7]

4. Control Flow [T1 chap.6] [5]

Expression Evaluation [T1 6.1]

Precedence and Associativity, Assignments, Initialization, Ordering Within Expressions, Short-Circuit Evaluation

Structured and Unstructured Flow [T1 6.2]

Structured Alternatives to goto

Sequencing [T1 6.3]

Selection [T1 6.4]

Short-Circuited Conditions, Case/Switch Statements

Iteration [T1 6.5]

Enumeration-Controlled Loops, Combination Loops, Iterators, Logically Controlled Loops

Recursion [T1 6.6]

Iteration and Recursion, Applicative- and Normal-Order Evaluation

5. Data Types [T2 chap.6] [8]

Introduction [T2 6.1]

Primitive Data Types [T2 6.2]

Numeric Types [T2 6.2.1]

Integer [T2 6.2.1.1]

- Floating point [T2 6.2.1.2]
- Complex [T2 6.2.1.3]
- Decimal [T2 6.2.1.4]
- Boolean Types[T2 6.2.2]
- Character Types [T2 6.2.3]
- Character String Types [T2 6.3]
  - Design Issues [T2 6.3.1]
  - Strings and Their Operations [T2 6.3.2]
  - String Length Operations [T2 6.3.3]
  - Evaluation [T2 6.3.4]
  - Implementation of Character String Types [T2 6.3.5]
- User defined Ordinal types [T2 6.4]
  - Enumeration types [T2 6.4.1]
    - Designs
    - Evaluation
  - Subrange types [T2 6.4.2]
    - Ada's design
    - Evaluation
  - Implementation fo used defined ordinal types [T2 6.4.3]

Array types [T2 6.5]
- Design issues [T2 6.5.1]
- Arrays and indices [T2 6.5.2]
- Subscript bindings and array categories [T2 6.5.3]
- Heterogeneous arrays [T2 6.5.4]
- Array initialization [T2 6.5.5]
- Array operations [T2 6.5.6]
- Rectangular and Jagged arrays [T2 6.5.7]
- Slices [T2 6.5.8]
- Evaluation [T2 6.5.9]
- Implementation of Array Types [T2 6.5.10]

Associative Arrays [T2 6.6]
- Structure and operations [T2 6.6.1]
- Implementing associative arrays [T2 6.6.2]

Record types [T2 6.7]
- Definitions of records [T2 6.7.1]
- References to record fields [T2 6.7.2]
- Operations on records [T2 6.7.3]
- Evaluation [T2 6.7.4]
- Implementation of Record types [T2 6.7.5]

Union Types [T2 6.8]
- Design issues [T2 6.8.1]
- Discriminated versus Free unions [T2 6.8.2]
- Evaluation [T2 6.8.4]
- Implementation of Union types [T2 6.8.5]

Pointer and Reference Types [T2 6.9]
- Design issues [T2 6.9.1]
- Pointer operations [T2 6.9.2]
- Pointer problems [T2 6.9.3]
  - Dangling pointers
  - Lost heap dynamic variables
- Pointers in C and C++ [T2 6.9.5]
- Reference types [T2 6.9.6]
- Evaluation [T2 6.9.7]

Implementation of pointer and reference types [T2 6.9.8]
- Representation of pointers and references
- Solution to dangling pointer problem
- Heap management

6. Subroutines and Control Abstraction [T2 chap.9,10] [5]

Fundamentals of Subprograms [T2 9.2 (excluding 9.2.4)]
Design Issues for subprograms [T2 9.3]
Local Referencing Environments [T2 9.4]
Parameter-Passing Methods [T2 9.5]
Parameters That are Subprograms [T2 9.6]
Overloaded Subprograms [T2 9.7]
Generic Subroutines [T2 9.8]
- Generic Functions in C++ [T2 9.8.2]
- Generic Methods in Java [T2 9.8.3]

Design Issues for Functions [T2 9.9]
User-Defined Overloaded Operators [T2 9.10]
Coroutines [T2 9.10]
The General Semantics of Calls and Returns [T2 10.1]
Implementing "Simple" Subprograms [T2 10.2]
Implementing Subprograms with Stack-Dynamic Local Variables [T2 10.3]
Nested Subprograms [T2 10.4]
Blocks [T2 10.5]
Implementing Dynamic Scoping [T2 10.6]

7. Data Abstraction and Object Orientation [T1 chap.9] [8]

Object-Oriented Programming [T1 9.1]
Encapsulation and Inheritance [T1 9.2]
- Modules, Classes, Nesting (Inner Classes), Type Extensions, Extending without Inheritance

Initialization and Finalization [T1 9.3]
- Choosing a Constructor, References and Values, Execution Order, Garbage Collection

Dynamic Method Binding [T1 9.4]
- Virtual- and Non-Virtual Methods, Abstract Classes, Member Lookup, Polymorphism, Object Closures

Multiple Inheritance [T1 9.5]
- Semantic Ambiguities, Replicated Inheritance, Shared Inheritance, Mix-In Inheritance

8. Concurrency [T2 chap. 13] [5]

Introduction
- Multiprocessor Architecture [T2 13.1.1]
- Categories of concurrency [T2 13.1.2]
- Motivations for studying concurrency [T2 13.1.3]

Introduction to Subprogram-level concurrency
- Fundamental concepts [T2 13.2.1]
- Language Design for concurrency. [T2 13.2.2]
- Design Issues [T2 13.2.3]

Semaphores
- Introduction [T2 13.3.1]
- Cooperation synchronization [T2 13.3.2]

- Competition Synchronization [T2 13.3.3]
- Evaluation [T2 13.3.4]

Monitors
- Introduction [T2 13.4.1]
- Cooperation synchronization [T2 13.4.2]
- Competition Synchronization [T2 13.4.3]
- Evaluation [T2 13.4.4]

Message Passing
- Introduction [T2 13.5.1]
- The concept of Synchronous Message Passing [T2 13.5.2]

Java Threads
- The **Thread** class [T2 13.7.1]
- Priorities [T2 13.7.2]
- Competition Synchronization [T2 13.7.3]
- Cooperation Synchronization [T2 13.7.4]
- Evaluation [T2 13.4.5]

Text Books:

T1. Scott Programming Language Pragmatics, 3e(With CD) ISBN 9788131222560 Kaufmann Publishers, An Imprint of Elsevier, USA
T2. Concepts of Programming Languages, Eighth Edition by Robert W. Sebesta, Pearson Education.
T3. Introduction to Turbo Prolog by Carl Townsend
T4. LISP 3rd edition by Patrick Henry Winston & Berthold Klaus Paul Horn (BPB)

Additional Reading:
Programming Languages: Principles and Paradigms, M. Gabbrielli, S. Martini, Springer, ISBN: 9781848829138

CS102 (New) - Advanced Networking

1. Review of Basic Concepts [3]

TCP/IP Protocol Suite	[T1	2.3]
Underlying Technologies : LAN (802.3)	T 1	3.1
Wireless Lans (802.11)	T 1	3.2
Point-to-point WANS	T 1	3.3
Switched WANS	T 1	3.4

2. The Internet Layer Protocols [4]

Review of IPv4 Protocol	T 1	7.1,7.2,7.3,7.4,7.5
IPv6	T 1	27.1,27.2
Transition from IPv4 to IPv6	T 1	27.3
ICMPv4	T 1	9.1,9.2,9.3,9.4
ICMPv6	T 1	28.1,28.2,28.3,28.4

3. Routing Protocols [6]

Forwarding	T 1	6.2
Structure of a Router	T 1	6.3
Routing Tables	T 1	11.1
Intra – And Inter-Domain Routing	T 1	11.2
Distance Vector Routing	T 1	11.3
RIP	T 1	11.4
OSPF	T 1	11.6
BGP	T 1	11.8
Multicast Routing	T 1	.4

4. The Transport Layer [6]

The Transport Service	T 2	6.1
Elements of Transport Protocols	T 2	6.2
UDP	T 2	6.4.1
TCP	T 2	6.5.1 to 6.5.9

5. Multimedia [3]

Digitizing Audio and Video	T 1	25.2
Streaming stored Audio / Video	T 1	25.4
Streaming Live Audio / Video	T 1	25.5
Real-Time Interactive Audio / Video	T 1	25.6
RTP	T 1	25.7
RTCP	T 1	25.8
Voice Over IP	T 1	25.9

6. Introduction To Security [2]

The need for Security	T 3	1.2
Security Approaches	T 3	1.3
Principles of Security	T 3	1.4
Types of Attacks	T 3	1.5

7. Cryptography: Concepts and Techniques [3]

Introduction	T 3	2.1
Plain Text and Cipher Text	T 3	2.2
Substitution Techniques	T 3	2.3.1,2.3.2,2.3.3,2.3.7
Transposition Techniques	T 3	2.4.1,2.4.2,2.4.3
Symmetric and Asymmetric key cryptography	T 3	2.6.1,2.6.2

8. Symmetric Key Algorithms [3]

Algorithms types and modes	T 3	3.2.1,3.2.2

DES	T 3	3.4
9. Asymmetric key Algorithms [2]		
RSA	T 3	4.4
Symmetric and Asymmetric key Cryptography	T 3	4.5
Digital Signatures	T 3	4.6.1,4.6.2
10. Digital Certificates [2]		
Introduction	T 3	5.1
Digital Certificates	T 3	5.2
11. Internet Security Protocols [10]		
Secure Socket Layer	T 3	6.3
TLS	T 3	6.4
SHTTP	T 3	6.5
TSP	T 3	6.6
SET	T 3	6.7
SSL Verses SET	T 3	6.8
3-D Secure Protocol	T 3	6.9
Electronic Money	T 3	6.10
Email Security	T 3	6.11
Firewalls	T 3	9.3
IP Security	T 3	9.4
VPN	T 3	9.5
12. User Authentication [4]		
Passwords	T 3	7.3
Certificate-based Authentication	T 3	7.5
Kerberos	T 3	7.7
Security Handshake Pitfalls	T 3	7.9

Text Books:

T1 : TCP / IP Protocol Suite Fourth Edition – Behrouz A. Forouzan
T2 : Computer Networks Fourth Edition – Andrew Tanenbaum
T3 : Cryptography and Network Security Second Edition – Atul Kahate

Supplementary but very useful references/texts: (Few of the references below contain latest research and trends related to Networks and Security and are useful for seminar/presentations by the students.)

1. Computer Network Security, Kizza, Springer, 9780387204734
2. Guide to Computer Network Security, Kizza, Springer, 978-1-84800-916-5
3. Network Security, Harrington, Elsevier, ISBN 9788131202166
4. Douglas E. Comer, Internetworking with TCP/IP, Vol. 1, Principles, Protocols and Architecture Fifth Edition, Prentice Hall, 2000, ISBN 0-13-018380-6.
5. William Stallings, Data and Computer Communications , Seventh Edition, Pearson Education
6. Douglas E. Comer, Internetworking with TCP/IP, Vol. 2, Design, Implementation and Internals, Prentice Hall Publisher.
7. Internetworking with TCP/IP, Vol. 3, Client-server Programming and Applications by Douglas E. Comer, Prentice Hall Publisher. (Excellent reference for distributed programming over TCP/IP networks)
8. Richard Stevens, TCP/IP Illustrated, Vol. 1, by, Addison Wesley (A very practical book with lots of useful network diagnostic tools and programs.)
9. Craig Hunt, TCP/IP Network Administration O’Reilly & Associates, Inc. (A must for network and system administrators dealing with internetworking.)

10. L. Peterson and B. Davie. Morgan , Computer Networks: A Systems Approach by Kaufmann Publishers Inc., ISBN 9788131210451
11. J. Kurose, K. Ross ``Computer Networking: A Top-Down Approach Featuring the Internet" Addison-Wesley, '00
12. William Stallings," Cryptography And Network Security” Prentice Hall /Pearson Education

Guidelines to paper setters:

Frame formats of protocols are not expected
Problems should be asked on Routing Protocols , TCP, Cryptography, RSA

CS-103(New): Distributed Database Concepts

Pre-requisites: Students should be well-versed with the basic and advanced concepts of RDBMS
Objectives:
Main objective is to understand the principles and foundations of distributed databases. This course addresses architecture, design issues, integrity control, query processing and optimization, transactions, and concurrency control & distributed transaction reliability.

Distributed databases: An overview			**[2]**
1.1	Features of distributed Vs centralized databases	Chapter 1 from Book 2	
1.2	Why DDB? DDBMS		
1.3	Promises / problem areas in implementing a DDB	Section 1.3,1.5 from Book 1	
DDBMS Architecture			**[4]**
2.1	DBMS Standardization	Chapter 4 from Book 1	
2.2	Architectural models for DDBMS		
2.3	DDBMS architecture		
2.4	Distributed catalog management	Section 21.8 from Book 3	
Distributed database design			**[10]**
3.1	Alternative design strategies	Chapter 5 from book 1	
3.2	Distributed design issues		
3.3	Concepts of join graphs	Section 4.2.1.2 from book 2	
3.4	Fragmentation and allocation	Chapter 5 from Book1	
Overview of Query processing			**[4]**
4.1	Query processing problems		
4.2	Objectives of query processing	Chapter 7 from	
4.3	Complexity of relational algebra operators	book 1	
4.4	Characterization of query processors		
4.5	Layers of query processing		
Query decomposition & data localization			**[2]**
5.1	Query decomposition		
5.2	Localization of distributed data	Chapter 8 from book 1	
Optimization of distributed queries			**[10]**
6.1	Query optimization		
.	Centralized query optimization Join ordering in fragment queries. Distributed query optimization algorithms	Chapter 9 from book1	
6.2	Centralized query optimization		
6.3	Join ordering in fragment queries		
6.4	Distributed query optimization algorithms		
Management of distributed transactions			**[2]**
7.1	Framework for transaction management	Chapter 7 from book 2	
7.2	Supporting atomicity of distributed transactions		
7.3	Concurrency control of distributed transactions		
7.4	Architectural aspects of distributed transactions		

Concurrency control **[6]**

8.1	Foundations of distributed concurrency control	Chapter 8 from book 2
8.2	Distributed deadlocks	
8.3	Concurrency control based on timestamps	
8.4	Optimistic methods for distributed concurrency control	

Distributed DBMS reliability **[8]**

9.1	. Reliability concepts & measures	
9.2	Failures & fault tolerance in distributed systems	Chapter from book 1
9.3	Failures in DDBMS	
9.4	Local reliability protocols	
9.5	Distributed reliability protocols	
9.6	Dealing with site failures	
9.7	Network partitioning	

Reference Books:

1. Principles of Distributed Database Systems; 2nd Edition By M. Tamer Ozsu and Patrick Valduriez Publishers: Pearson Education Asia ISBN: 81-7808-375-2
2. Distributed Database; Principles & Systems By Stefano Ceri and Giuseppo Pelagatti Publications: McGraw-Hill International Editions ISBN: 0-07-010829-3
3. Database systems (2nd edition) By Raghuramakrishnan and Johannes

CS-104(New): Design and Analysis of Algorithms

Prerequisites

- Basic algorithms and data structure concepts.
- Basic programming concepts

Objectives

This course will prepare students in

- Basic Algorithm Analysis techniques and understand the use o asymptotic notation
- Understand different design strategies
- Understand the use of data structures in improving algorithm performance
- Understand classical problem and solutions
- Learn a variety of useful algorithms
- Understand classification o problems

1. Analysis

Algorithm definition, space complexity, time complexity, worst case –best case –average case complexity, asymptotic notation, sorting algorithms (insertion sort, heap sort) , sorting in linear time, searching algorithms, recursive algorithms (Tower of Hanoi , Permutations).
[T1 1.1 , 1.2, 1.3] **[6]**

2. Design strategies

Divide and conquer-control abstraction, binary search, merge sort, Quick sort, Strassen's matrix multiplication [T1 3.1, 3.2, 3.4,3.5,3.7] **[6]**

Greedy method- knapsack problem, job sequencing with deadlines,
Minimum-cost spanning trees, Kruskal and Prim's algorithm, optimal storage on tapes, optimal merge patterns, Huffman coding [T1 4.1, 4.2, 4.4, 4.5, 4.6,4.7, 4.8] **[8]**

Dynamic programming- matrix chain multiplication, **.** single source shortest paths, Dijkstra's algorithm, Bellman- ford algorithm , all pairs shortest path, longest common subsequence, string editing, 0/1 knapsack problem, Traveling salesperson problem.
[T1 5.1, 5.3, 5.6, 5.7, 5.9] **[8]**

Decrease and conquer: - DFS and BFS, Topological sorting, connected components
[T6.1, 6.2, 6.3, 6.4] **[6]**

Backtracking: General method, 8 Queen's problem, Sum of subsets problem, graph coloring problem, Hamiltonian cycle
[T1 7.1 , 7.2, 7.3, 7.4, 7.5] **[4]**

Branch and Bound Technique : FIFO, LIFO, LCBB, TSP problem, 0/1 knapsack problem
[T1 8.1.1, 8.2, 8.3 **[4]**

Transform and conquer:- Horner's Rule and Binary Exponentiation – Problem Reduction –
[T1 9.1, 9.2 ,9.3] **[4]**

Problem classification

Nondeterministic algorithm, The class of P, NP, NP-hard and NP- Complete problems, significance of Cook's theorem
[T1 11.1] **[2]**

Text Books

T1. Ellis Horowitz, Sartaj Sahni & Sanguthevar Rajasekaran, Computer Algorithms, Galgotia.
T2 T. Cormen, C. Leiserson, & R. Rivest, Algorithms, MIT Press, 1990 1

References Texts

1) A. Aho, J. Hopcroft, & J. Ullman, The Design and Analysis of Computer Algorithms, Addison Wesley, 1974
2) Donald Knuth, The Art of Computer Programming (3 vols., various editions, 1973-81), Addison Wesley
3) The Algorithm Manual, Steven Skiena, Springer ISBN:9788184898651
4) Graphs, Networks and Algorithms, Jungnickel, Springer, ISBN: 3540219056

CS-105: Lab Course (Departmental)

CS -101 – [10 MARKS]

PPL Assignments

LISP

Sample Set of Assignments:

1. Define a LISP function to compute sum of squares.
2. Define a LISP function to compute difference of squares. (if x > y return x^2-y^2, otherwise y^2-x^2)
3. Define a Recursive LISP function to solve Ackermann's Function.
4. Define a Recursive LISP function to compute factorial of a given number.
5. Define a Recursive LISP function which takes one argument as a list and returns last element of the list. (do not use last predicate)
6. Define a Recursive LISP function which takes one argument as a list and returns a list except last element of the list. (do not use butlast predicate)
7. Define a Recursive LISP function which takes one argument as a list and returns reverse of the list. (do not use reverse predicate)
8. Define a Recursive LISP function which takes two arguments first, an atom, second, a list, returns a list after removing first occurrence of that atom within the list.
9. Define a Recursive LISP function which appends two lists together.
10. Define a recursive LISP function which takes 2 lists as arguments and returns a list containing alternate elements from each list.
    e.g. if L1=(1 5 7) and L2=(2 4 9 3) output should be
    (1 2 5 4 7 9 3)

Prolog:

Sample Set of Assignments:

1. Prolog programs doing formal reasoning and resolutions proofs.
   e.g. Consider the following statements: "John likes all kinds of food. Apples are food. Chicken is food. Anything anyone eats, and is still alive, means whatever he ate was a food. Sue eats everything Bill eats. Bill eats Peanuts and is still alive." Write a Prolog program to prove that John likes Peanuts, and to answer the question "What food does Sue eat?"
2. Simple Prolog programs using facts, rules, built-in I/O predicates, fail, recursion with or without repeat predicate, and cut predicate.

CS -102 – [10 MARKS]

Suggested List of Assignments:

1.Write a C prog that contains a string (char pointer) with a value "Hello World" .The program should XOR each character in this string with 0 & display the result. Repeat the exercise by an XOR operation with 1.

2. Study phishing in more detail. Find out which popular bank sites have been phished and how .

3. Think about offering phishing prevention techniques. Which ones of them would be most effective and why ?

4. Encrypt the following message by using ___________ technique. Key is _____.

5. Write C/Java program to implement DES algorithm logic.

6. Write C/Java program that generate the message digest of a given number.

7. Write a Java prog , which accepts the details of a base-64 encoded digital certificate, parses it and displays its main contents such as issuer name, serial number, subject name , valid from and valid to.

8. Write a C program to perform Base-64 encoding on a 24 bit input.

9. SSL talks about security at the transport layer. What if we want to enforce security at the lower layers

10. Investigate how to use SSL in Java & .Net. Write an SSL client and server in these technologies.

11. Implement Kerberos.

. Download &Study at least one free real life firewall product.

13.Study how VPN is implemented.

CS -103 – [10 MARKS]

Suggested Topic List of Assignments For CS-103 (DDB):

1. Problems on Fragmentation : Small case studies, where each case specifies the database schema, the workload for the database is given. Based on this info, the fragmentation has to be done (horizontal or vertical)

2. Problems on deriving & classifying the join graphs, given the relation fragments, of a database

3. Problems on query optimization (Exrecise problems given in reference book , “Database systems” By Raghuramakrishnan (the 3rd reference book given in the syllabus)

4. Problems on concurrency control algorithms (similar to the problems given in exercises of reference book 1 & 2)

CS -104 – Lab/ Home Assignments [10 MARKS]

1. Problem Assignments

A set of problems from the exercises of T1 and T2 can be given as problem solving assignments

2. Programming Assignment

Implementing a set of algorithms And comparing their performance on an input set.

Seminar [10 Marks]

Suggested List of Seminar Topics:

Wireless Security
Wi-Fi and WIMAX
Digital Certificates and Digital Signatures
Mobile Security
Public Key Infrastructure

Project [50 Marks]

M.Sc. (Computer Science)
First Year Semester 2

CS-201 (New): Digital Image Processing

Introduction **4**

Definition of Digital Image Processing, The Origins of Digital Image Processing, Examples of Fields that Use Digital Image Processing - X-ray Imaging, Ultraviolet Band, Visible and Infrared Bands, Microwave Band, and Radio Band Imaging; Fundamental Steps in Digital Image Processing, Components of an Image Processing System,

Digital Image Fundamentals **4**

Elements of Visual Perception, Light and the Electromagnetic Spectrum, Image Sensing and Acquisition - Single Sensor, Sensor Strips, Sensor Arrays, A Simple Image Formation Model; Image Sampling and Quantization - Spatial and Gray-Level Resolution, Aliasing and Moiré Patterns, Zooming and Shrinking Digital Images; Some Basic Relationships Between Pixels - Neighbors, Adjacency, Connectivity, Regions, and Boundaries, Distance Measures, Image Operations on a Pixel Basis; Linear and Nonlinear Operations

Image Enhancement in the Spatial Domain **8**

Some Basic Gray Level Transformations - Negatives, Log, Power-Law, Piecewise-Linear Transformations; Histogram Processing - Histogram Equalization, Histogram Matching (Specification), Local Enhancement; Enhancement Using Arithmetic/Logic Operations - Image Subtraction, Image Averaging; Basics of Spatial Filtering, Smoothing Spatial Filters - Smoothing Linear and Order-Statistics Filters; Sharpening Spatial Filters - Use of Second Derivatives for Enhancement : The Laplacian, Use of First Derivatives for Enhancement: The Gradient; Combining Spatial Enhancement Methods

Image Enhancement in the Frequency Domain **7**

Introduction to the Fourier Transform and the Frequency Domain - One-Dimensional Fourier Transform and its Inverse, Two-Dimensional DFT and Its Inverse, Filtering in the Frequency Domain, Correspondence between Filtering in the Spatial and Frequency Domains; Smoothing and Frequency-Domain Filters - Ideal , Butterworth, and Gaussian Lowpass Filters; Sharpening Frequency Domain Filters - Ideal , Butterworth, and Gaussian Highass Filters, Laplacian in the Frequency Domain, Unsharp Masking, High-Boost Filtering, and High-Frequency Emphasis Filtering; Homomorphic Filtering Implementation - Some Additional Properties of the 2-D Fourier Transform, Computing the Inverse Fourier Transform Using a Forward Transform Algorithm, More on Periodicity: the Need for Padding, The Convolution and Correlation Theorems, Summary of Properties of the 2-D Fourier Transform, The Fast Fourier Transform;

Image Restoration **5**

A Model of the Image Degradation/Restoration Process, Noise Models, Restoration in the Presence of Noise Only – Spatial Filtering - Mean, Order-Statistics, and Adaptive Filters Filters; Periodic Noise Reduction by Frequency Domain Filtering - Bandreject, Bandpass, and Notch Filters Filters; Estimating the Degradation Function - Estimation by Image Observation, Experimentation and Modeling; Inverse Filtering, Minimum Mean Square Error (Wiener) Filtering, Geometric Mean Filter - Geometric Transformations, Spatial Transformations, Gray-Level Interpolation

Color Image Processing 5

Color Fundamentals, Color Models - RGB, CMY, HSI; Pseudocolor Image Processing - Intensity Slicing, Gray Level to Color Transformations; Basics of Full-Color Image Processing, Color Transformations - Formulation, Color Complements, Color Slicing, Tone and Color Corrections, Histogram Processing; Smoothing and Sharpening, Color Segmentation, Color Edge Detection, Noise in Color Images

Morphological Image Processing 4

Some Basic Concepts from Set Theory, Logic Operations Involving Binary Images, Dilation and Erosion, Opening and Closing, The Hit-or-Miss Transformation, Some Basic Morphological Algorithms - Boundary Extraction, Region Filling, Extraction of Connected Components, Convex Hull, Thinning, Thickening; Extensions to Gray-Scale Images

Image Segmentation 6

Detection of Discontinuities - Point Detection, Line Detection, Edge Detection, Edge Linking and Boundary Detection - Local Processing, Global Processing via the Hough Transform, Thresholding - The Role of Illumination, Basic Global Thresholding, Basic Adaptive Thresholding, Optimal Global and Adaptive Thresholding, Use of Boundary Characteristics for Histogram Improvement and Local Thresholding, Thresholds Based on Several Variables, Region-Based Segmentation - Region Growing, Region Splitting and Merging,

Representation and Description 5

Chain Codes, Polygonal Approximations, Signatures, Boundary Segments, Skeletons, Simple Boundary Descriptors, Shape Numbers, Fourier Descriptors, Statistical Moments, Simple Regional Descriptors, Topological Descriptors, Texture, Moments of Two-Dimensional Functions Use of Principal Components for Description, Relational Descriptors

Text Book:

1. Gonzalez, R. C. and Woods, R. E. [2002/2008], Digital Image Processing, 2nd/3rd ed., Prentice Hall

Reference Books:

1. Sonka, M., Hlavac, V., Boyle, R. [1999]. Image Processing, Analysis and Machine Vision (2nd edition), PWS Publishing, or (3rd edition) Thompson Engineering, 2007
2. Gonzalez, R. C., Woods, R. E., and Eddins, S. L. [2009]. Digital Image Processing Using MATLAB, 2nd ed., Gatesmark Publishing, Knoxville, TN.
3. Anil K. Jain [2001], Fundamentals of digital image processing (2nd Edition), Prentice-Hall, NJ
4. Willian K. Pratt [2001], Digital Image Processing (3rd Edition), , John Wiley & Sons, NY
5. Burger, Willhelm and Burge, Mark J. [2008]. Digital Image Processing: An Algorithmic Introduction Using Java, Springer
6. Digital Image Analysis (With CD-ROM), Kropatsch, Springer, ISBN 978038795066
7. Digital Image Processing, 6e (With CD), Jähne, Springer, ISBN:978-3-540-24035-8 2

CS-202(New): Advanced Operating System

Prerequisites:

- Working knowledge of C programming.
- Working knowledge of NASM/GAS assembler for 80x86 (32 and 64 bit) instruction Set
- Basic Computer Architecture concepts.
- Basic algorithms and data structure concepts.

Course Objectives:
This course teaches Advanced Operating Systems Concepts using Unix/Linux and Windows as representative examples. This course strikes a delicate balance between theory (covered in TextBook-2,3) and practical applications (covered in TextBook-1,4,5,6). In fact, most chapters start with the theory and then switches focus on how the concepts in a C program. This course describes the programming interface to the Unix/Linux system - the system call interface. It is intended for anyone writing programs that run under Unix/Linux. Finally, it concludes with an overview of Windows Internals. This course provides an understanding of the functions of Operating Systems. It also provides provide an insight into functional modules of Operating Systems. It discusses the concepts underlying in the design and implementation of Operating Systems. This course also gives implementation details at lower level using Assembly language Programming by creating some interest in system call design.

Syllabus:

1. **Introduction to UNIX/Linux Kernel**
   [03]
   - System Structure, User Perspective, Assumptions about Hardware, Architecture of UNIX Operating System (TextBook-3: Chapter Topics: 1.2, 1.3, 1.5, 2.1)
   - Concepts of Linux Programming, Getting Started with System Programming (TextBook-1: Chapter 1-Relevant Topics)
   - Introduction to the tools on Linux (Chapter 3 Text Book 5), NASM (Chapter 3,4,7,8 Book 6)

2. **File and Directory I/O**
   [10]
   - inodes, structure of regular file, open, read, write, lseek, close, pipes, dup (TextBook-3: Chapter Topics: 4.1, 4.2, 5.1-5.6)
   - open, creat, close, lseek, read, write, file sharing, atomic operations, dup and dup2, fcntl, ioctl, /dev/fd, stat, fstat, lstat, file types, Set-User-ID and Set-Group-ID, file access permissions, ownership of new files and directories, access function, umask function, chmod and fchmod, sticky bit, chown, fchown, and lchown, file size, file truncation, file systems, link, unlink, remove, and rename functions, symbolic links, symlink and readlink functions, file times, utime, mkdir and rmdir, reading directories, chdir, fchdir, and getcwd, device special files (TextBook-4: Chapter Topics: 3.3-3.16, 4.2-4.23)
   - Scatter/Gather I/O, The Event Poll Interface, Mapping Files into Memory, Advice for Normal File I/O, Synchronized, Synchronous, and Asynchronous Operations, I/O Schedulers and I/O Performance, Files and their Metadata, Directories, Links,

Copying and Moving files, Device Nodes, Out-of-Band Communication, Monitoring File Events (TextBook-1: Chapters: 4 and 7)

3. **Process Environment, Process Control and Process Relationships**
   [12]
   - Process states and transitions, the context of a process, saving the context of a process, sleep, process creation, signals, process termination, awaiting process termination, invoking other programs, the user id of a process (TextBook-3: Chapter Topics: 6.1, 6.3, 6.4, 6.6, 7.1-7.6)
   - Process termination, environment list, memory layout of a C program, shared libraries, memory allocation, environment variables, setjmp and longjmp, getrlimit and setrlimit, process identifiers, fork, vfork, exit, wait and waitpid, waitid, wait3 and wait4, race conditions, exec, changing user IDs and group IDs, interpreter files, system function, process accounting, user identification, process times, Terminal logins, network logins, process groups, sessions, controlling terminal, tcgetpgrp, tcsetpgrp, and tcgetsid functions, job control, shell execution of programs, orphaned process groups (TextBook-4: Chapter Topics: 7.3-7.11, 8.2-8.16, 9.2-9.10)
   - The Process ID, Running a New Process, Terminating a Process, Waiting for Terminated Child Processes, Users and Groups, Sessions and Process Groups, Daemons, Process Scheduling, Yielding the Processor, Process Priorities, Processor Affinity (TextBook-1: Chapter 5 and 6 [Relevant Topics])

4. **Memory Management**
   [04]
   - The Process Address Space, Allocating Dynamic Memory, Managing Data Segment, Anonymous Memory Mappings, Advanced Memory Allocation, Debugging Memory Allocations, Stack-Based Allocations, Choosing a Memory Allocation Mechanism, Manipulating Memory, Locking Memory, Opportunistic Allocation (TextBook-1: Chapter 8)

5. **Signal Handling**
   [08]
   - Signal concepts, signal function, unreliable signals, interrupted system calls, reentrant functions, SIGCLD semantics, reliable-signal technology, kill and raise, alarm and pause, signal sets, sigprocmask, sigpending, sigaction, sigsetjmp and siglongjmp, sigsuspend, abort, system function revisited, sleep, job-control signals (TextBook-4: Topics: 10.2-10.20)
   - Signal Concepts, Basic Signal Management, Sending a Signal, Reentrancy, Signal Sets, Blocking Signals, Advanced Signal Management, Sending a Signal with a Payload (TextBook-1: Chapter 9)

6. **Windows Internals** (TextBook-2: Chapter 1, 2, and 5 [relevant topics])
   [10]
   - Foundation Concepts, utilities and Terms
     - Services, Functions, and Terms, Processes, Threads, and Jobs, Virtual Memory, Kernel Mode vs. User Mode

- Introduction to the Sysinternals Troubleshooting Utilities (File and Disk, Process , Security, System Information and Miscellaneous Utilities) (http://download.sysinternals.com/Files/SysinternalsSuite.zip)
- System Architecture
    - Requirements and Design Goals, OS Model, Architecture Overview (Portability, Symmetric Multiprocessing)
- Process Internals
    - Data Structures, Kernel variables, Performance Counters, Relevant Functions
- Protected Processes
- Flow of *CreateProcess*
    - Stage 1 through Stage 7, New Process
- Thread Internals
    - Data Structures, Kernel Variables, Performance Counters, Relevant Functions, Birth of a Thread
- Examining Thread Activity
    - Limitations on Protected Process Threads
- Worker Factories (Thread Pools)
- Thread Scheduling
    - Overview of Windows Scheduling, Priority Levels, Windows Scheduling APIs, Relevant Tools, Real-Time Priorities, Thread States, Dispatcher Database, Quantum, Scheduling Scenarios, Context Switching, Idle Thread, Priority Boosts
- Job Objects

Recommended Text:

1. Linux System Programming, O'Reilly, by Robert Love. (Chapter 1 and 4-9 [Relevant Topics])

2. Windows Internals, Microsoft Press, by Mark E. Russinovich and David A. Soloman. Chapter References: (Chapter 1, 2, and 5 [Relevant Topics])

3. The Design of the UNIX Operating System, PHI, by Maurice J. Bach.
   Chapter References: (1.2, 1.3, 1.5, 2.1, 4.1, 4.2, 5.1-5.6, 6.1, 6.3, 6.4, 6.6, 7.1-7.6)

4. Advanced Programming in the UNIX Environment, Addison-Wesley, by Richard Stevens.
   Chapter References: (3.3-3.16, 4.2-4.23, 7.3-7.11, 8.2-8.16, 9.2-9.10, 10.2-10.20)
5. Guide to Assembly Language Programming in Linux, Sivarama P. Dandamudi, Springer
6. Professional Assembly Language, Richard Blum, Wrox, Wiley India

CS-203(New): Data Mining and Data Warehousing

1. **Introduction to Data Mining** 4
   - Basic Data Mining Tasks
   - DM versus Knowledge Discovery in Databases
   - Data Mining Issues
   - Data Mining Metrics
   - Social Implications of Data Mining
   - Overview of Applications of Data Mining
2. **Introduction to Data Warehousing** 4
   - Architecture of DW
   - OLAP and Data Cubes
   - Dimensional Data Modeling-star, snowflake schemas
   - Data Preprocessing – Need, Data Cleaning, Data Integration & Transformation, Data Reduction
   - Machine Learning
   - Pattern Matching
3. **Data Mining Techniques** 4
   - Frequent item-sets and Association rule mining: Apriori algorithm, Use of sampling for frequent item-set, FP tree algorithm
   - Graph Mining: Frequent sub-graph mining, Tree mining, Sequence Mining
4. **Classification & Prediction** 16
   - Decision tree learning: [3 hrs]
     Construction, performance, attribute selection
     Issues: Over-fitting, tree pruning methods, missing values, continuous classes
     Classification and Regression Trees (CART)
   - Bayesian Classification: [6 hrs]
     Bayes Theorem, Naïve Bayes classifier,
     Bayesian Networks
     Inference
     Parameter and structure learning
   - Linear classifiers [4 hrs]
     Least squares, logistic, perceptron and SVM classifiers
   - Prediction [3 hrs]
     Linear regression
     Non-linear regression

5 **Accuracy Measures** 4
Precision, recall, F-measure, confusion matrix, cross-validation, bootstrap

6. **Software for data mining and applications of data mining** 4
R, Weka, Sample applications of data mining

7. **Clustering** 4
   - k-means
   - Expectation Maximization (EM) algorithm
   - Hierarchical clustering, Correlation clustering

8. **Brief overview of advanced techniques** 4

- Active learning
- Reinforcement learning
- Text mining
- Graphical models
- Web Mining

Reference Books:

1. Data Mining: Concepts and Techniques, Han, Elsevier ISBN:9789380931913/ 9788131205358
2. Margaret H. Dunham, S. Sridhar, Data Mining – Introductory and Advanced Topics, Pearson Education
3. Tom Mitchell, ―Machine Learning‖, McGraw-Hill, 1997
4. R.O. Duda, P.E. Hart, D.G. Stork. Pattern Classification. Second edition. John Wiley and Sons, 2000.
5. Christopher M. Bishop, ―Pattern Recognition and Machine Learning‖, Springer 2006
6. Raghu Ramkrishnan, Johannes Gehrke, Database Management Sysstems, Second Edition, McGraw Hill International
7. Ian H.Witten, Eibe Frank Data Mining: Practical Machine Learning Tools and Techniques, Elsevier/(Morgan Kauffman), ISBN:9789380501864
8. [Research-Papers]: Some of the relevant research papers that contain recent results and developments in data mining field

CS-205: Laboratory Assignments

Image Processing Assignments (15 Marks)

Note: Images required for implementing the assignments below are made available on the link below:

http://cs.unipune.ac.in/obx/~hod_cs/dip.zip

Fundamentals and Coding Practice

1. Design 128, 64, 32, 16, 8 and 4-level uniform quantizers and quantize the gray-level image, *lena.pgm*. Compare the results by these six different quantizers. Explain the artifacts (e.g., the visibility of undesirable contours).
2. Divide the image *lena.pgm* into blocks and each block has the size 4 x 4 pixels. Replace each block by the intensity of the (2,2) pixel within the block. The new image will be 1/4th the size in both dimensions. Display the down-sampled image

Spatial Transforms and Filtering

3. Apply power law transformation to the *city.pgm* image taking different values for gamma (□ = 3, □ = 4, and □ = 5).
4. Compute and plot (show the image and its histogram) the histogram of *lena.pgm* and *city.pgm*. Comment on what information can be discerned about the images from an examination of the histogram.
5. Apply histogram equalization to the input images *lcgrain.pgm* and *darkgrain.pgm*; submit your code and the output images.
6. Reduce the salt-and-pepper noise in the *circuit.pgm* image; submit your code and the output image.

Filtering in Frequency Domain and Image Restoration

7. Remove the noise from the input images *circuit1.pgm*, *circuit2.pgm*, *circuit3.pgm*, and *moon.pgm* . Submit your code and the output images.
8. Restore the original images from the inputs *degrade1.pgm*, *degrade2.pgm* and *degrade3.pgm*.

Morphological Image Processing

9. Remove the noise from the input image *mimage.pgm*. Submit your code and the output image.
10. Extract the gradient parts from the input image *brain.pgm*. Perform edge detection.

Segmentation and Object recognition

11. Extract the rice objects from the input image *rice.pgm*.
12. Separate the two types of blobs in the input image *twoblobs.pgm*.
13. Develop an imaging application to detect and count human faces in an image.
14. Develop an imaging application to detect and count text lines and number of words in a scanned document.

NOTE:

a) The submitted answer to the assignments should be a maximum of 2 pages plus the relevant figures.

b) All programming should be done using C/C++/Java/OpenCV/MATLAB.

c) A copy of your complete program must be attached to your submitted answer as an appendix.

Advanced Operating Systems Assignments (10 Marks)

Advanced Operating Systems Assignments

1. Write your own dup2 function that performs the same service as the dup2 system call without calling the fcntl function. Be sure to handle errors correctly.
2. Write a utility like cp(1) that copies a file containing holes, without writing the bytes of 0 to the output file.
3. Write a C program that creates a *zombie*, and then call *system* to execute the ps(1) command to verify that the process is a *zombie*.
4. Implement your own sig2str function.
5. Write a C program that creates a file and writes the integer 0 to the file. The process then creates a child, and the Parent and Child alternate incrementing the counter in the file. Each time the counter is incremented, print which process (Parent or Child) is doing the increment.
6. Write a C program that calls fork and has the child create a new session. Verify that the child becomes a process group leader and that the child no longer has a controlling terminal.
7. Write a C function which handles all possible signals. The function should consist of a single loop that iterates once for every signal in the current signal mask (not once for every possible signal).
8. Write a C program that calls sleep (60) in an infinite loop. Every five times through the loop (every 5 minutes), fetch the current time of day and print the tm_sec field.
9. Write a C program that calls fwrite with a large buffer (a few hundred megabytes). Before calling fwrite, call alarm to schedule a signal in 1 second. In your signal handler, print that the signal was caught and return.
10. Any one assignment on windows internals (**Compulsory**) ***Please note that you are not expected to write sdk program related to windows programming)*****.**

11. Write as many programs in Assembly as you can for getting familiar with Assembly Programming, binutils(tool chain for system programmers), OS services on Linux, These programs should make use of system calls related to memory management, process management, file management etc) (**Minimum 5 programs Compulsory**)
12. Understanding how the utilities of Sysinternals suit (http://technet.microsoft.com/en-us/sysinternals/bb842062) are useful for system programmer point of view.

Project (Optional)
This is a small project. Write a simple scheduler in C for a microcontroller (Optional). The scheduler should be cooperative and event driven. When the scheduler is executes, it should hand over processor control by calling the primary function of the process which:

1. is waiting for attention
2. has the highest priority of the waiting programs
3. has been waiting the longest in case of a tie above

This behavior may result in lower priority processes never receiving attention from the processor. It is up to the programmer of the individual processes to ensure that the process returns to the scheduler in a timely matter.

The scheduler should maintain a list of all processes, in order of priority. For each process, the following information should be kept:

- the priority number,
- the process id number (an identifier unique to that process,
- a pointer to the function itself, and
- the status (waiting for attention or waiting for an event).

The order of the list should reflect the order in which processes should be called.

The scheduler should support the following functions to facilitate scheduling. The processes the scheduler managed can be known and created at compile time. Extra credit will be provided for implementing the functions that allow for dynamic process creation [hint: this requires the use of function pointers]. While the processes are known at compile time, the creation and use of semaphores and the setting of priorities should be possible at runtime.

Most of the functions are related to setting up the relationship between processes (priorities, semaphores, etc). None of these methods should be called from inside an interrupt because the interaction of interrupts with the scheduler data structure cannot be guaranteed. More advanced discussions of operating systems will explain atomic instructions and how they handle this impasse.

create_new_process

unsigned int *create_new_process*(void (*f)(void), unsigned int priority) [fact check how to pass function names as pointers]

parameters:

- pointer to the process function parameter
- priority of the process

returns:

- process id

This function is only necessary if dynamic process creation is implemented. If dynamic allocation is used, the create_new_process method will generate the unique identifier for each process. If static process creation is used, then the identifier may be added into the code. #define may be useful for making these process_ids human readable. One process_id should be reserved as an error code if something goes wrong. For example 0x0000 might be returned if the maximum number of processes has already been reached.

create_binary_semaphore

unsigned int*create_binary_semaphore*(unsigned int process, unsigned int state)

parameters:

- process id
- initial state of the semaphore

returns:

- semapore id

Semaphore creation will add a new semaphore to the process specified. The integer returned is the identifier for the semaphore that is unique to each process (rather unique to all processes). Each process should be allowed at least 8 semaphores. A process will not be allowed to run unless all of its created semaphores are ready.

set_semaphore

void *set_semaphore*(unsigned int process, unsigned int semaphore, unsigned int state)

parameters:

- process id
- priority of the process
- new state of the semaphore

Sets the state of a certain created semaphore of a certain process. The states are ready and waiting.

set_priority

void *set_priority*(unsigned int process, unsigned int new_priority) *parameters:*

- process id
- new priority of the process

schedule

void *schedule*(void)

The main() function calls schedule once the system has been set up. Schedule then handles all further negotiation of the processes. More advanced schedulers will actually adjust the stack pointer and the context data of each function call to allow seemless transition between processes. This process requires writing part of the scheduler in assembly to avoid the automatic adjustments C will make on its own. In this case, you need only call each function normally from inside schedule().

Laboratory Data Mining and Data Warehousing ASSIGNMENTS: (20 Marks)

Assignment 1: Using the WEKA Workbench

A. Become familiar with the use of the WEKA workbench to invoke several different machine learning schemes. Use latest stable version. Use both the graphical interface (Explorer) and command line interface (CLI). See Weka home page for Weka documentation.

B. Use the following learning schemes, with the default settings to analyze the weather data (in weather.arff). For test options, first choose "Use training set", then choose "Percentage Split" using default 66% percentage split. Report model percent error rate.

   - ZeroR (majority class)
   - OneR
   - Naive Bayes Simple
   - J4.8

C. Which of these classifiers are you more likely to trust when determining whether to play? Why?

D. What can you say about accuracy when using training set data and when using a separate percentage to train?

Assignment 2: Basic classification and usage of weka (Acknowledgement: Assignments 1 and 2 are from http://www.kdnuggets.com/data_mining_course/assignments/index.html)

- Become familiar with the use of the WEKA workbench to invoke several different machine learning schemes.

Use latest stable version. Use both the graphical interface (Explorer) and command line interface (CLI). See Weka home page for Weka documentation.

- Use the following learning schemes, with the default settings to analyze the weather data (in weather.arff). For test options, first choose "Use training set", then choose "Percentage Split" using default 66% percentage split. Report model percent error rate.
    - ZeroR (majority class)
    - OneR
    - Naive Bayes Simple
    - J4.8
- Which of these classifiers are you more likely to trust when determining whether to play? Why?
- What can you say about accuracy when using training set data and when using a separate percentage to train?

Assignment 3: Classification – I

Build different classification models using different classification algorithms (such as decision tree, naive bayes, bayesian networks) for the IRIS dataset that comes with weka / R. (UCI ML link: http://archive.ics.uci.edu/ml/datasets/Iris). Compare the accuracy (precision, recall, F1 measure) of these different classification algorithms.

Assignment 4: Classification – II

For this assignment, you will use the "Iris" data-set from UCI Machine Learning Repository. You can acess this data-set from http://archive.ics.uci.edu/ml/datasets/Iris. This data-set considers three classes of flowers : Iris Setosa, Iris Versicolour, Iris Virginica and has 50 samples from each flower (so a total of 150 samples). For each sample, it records 4 features: sepal length in cm; sepal width in cm; petal length in cm; petal width in cm

Task:

1. Divide the data into 2 parts : part I containing a random set of 30 samples, and the rest into part II.

2. Use part I to get prior probabilities of each class.

3. First you will use only the first feature to classify the flowers into these 3 classes. Assume that the class conditional distribution of "sepal length" is Gaussian for each of the 3 classes. Estimate the parameters of the three distributions using maximum likelihood estimation. Use data in part II.

4. Using the parameters obtained above, classify the data in part II. Report on the number of errors.

5. Now use all four features in Step 3 above : assume that the class conditional distribution is a 4-dimensional Gaussian. Repeat Step 4. (Acknowledgement: Assignment from IITDelhi course(http://www.cse.iitd.ac.in/~pkalra/siv895/assignment.html)

Assignment 5: Regression

Build a linear regression model for the housing dataset from UCI Machine learning repository (Dataset:http://archive.ics.uci.edu/ml/datasets/Housing). Use feature selection algorithm (say PCA) to reduce the dimensionality of data. Which are the features selected in this reduced dataset? Build a linear regression with the reduced dataset? Compare the two models built.

Elective Course [CS-204(N)/CS-304(N)]: Advanced Algorithm

- Advanced data structures (Fibonacci heaps, splay trees, dynamic trees, B-Trees) in-memory representations and persistence of DS, Revision of Graph algorithms: Network flows (max flow and min-cost flow/circulation) (10 Hrs)
- String algorithms: (10 Hrs)
    - String searching - (Knuth–Morris–Pratt algorithm, Boyer–Moore string search algorithm, Rabin–Karp string search algorithm)
    - Suffix trees - mathematical properties of suffix trees
    - Applications of Suffix trees:
        - regular expression searches using suffix trees;
        - Finding all maximal pairs and maximal repeats, Patricia trees

- Intractable problems: approximation algorithms (14 Hrs)

    1. Steiner tree and TSP
    2. Steiner forest
    3. Group Steiner trees
    4. Set cover via primal-dual
    5. k-median on a cycle

- Integer programming and optimization algorithms (14 Hrs.)

    1. Formulations, complexity and relaxations
    2. discrete optimization,
    3. cutting plane methods,
    4. enumerative and heuristic methods
    5. Convex programming algorithms: ellipsoid method, interior-point methods, proximal-point methods.

Preliminary reading:

- Introduction to Algorithms: by Cormen, T.H., C.E. Leiserson, R.L. Rivest, and C. Stein; MIT Press; ISBN: 9780262032933
- The Algorithm Manual, Steven Skiena, Springer ISBN:9788184898651

Reference Books:

- Theory of Linear and Integer Programming: by Schrijver; John Wiley & Sons. ISBN: 9780471982326
- Convex Optimization: by Boyd and Vandenberghe; Cambridge University Press; ISBN: 9780521833783
- Approximation Algorithms: by Vazirani; Springer-Verlag: ISBN: 9783540653677
- Advances in Steiner Trees (Combinatorial Optimization) by Ding-Zhu Du (Editor), J.M. Smith (Editor), J. Hyam Rubinstein (Editor); Springer; ISBN: 978-0792361107

- Algorithms On Strings,Trees, And Sequences; by D. Gusfield; Cambridge University Press, (ISBN 052158519)

Additional reading:

- Algorithmic Number Theory: by Bach and Shallit; MIT Press; ISBN: 9780262024051

Elective Course [CS-204(N)/CS-304(N)]: Functional Programming

- Introduction to Functional programming (2)
  - Expressions and values.
  - Functions.
  - Recursion.
  - Types.
- Introduction to Haskell (10)
  - tuples
  - polymorphism
  - higher order functions
  - strings & characters
- Data types (10)
  - Data-Type declarations
  - Data and type constructors
  - Defining functions over datatypes using pattern
  - Abstract data types
  - Polymorphism
  - Polymorphic Functions,
  - Polymorphic datatypes,
  - Type Constructors to define polymorphic Constructor functions
  - Recursive datatypes
  - Higher Order functions
- The Haskell Class System (6)
  - Classes as predicates on types
  - Instance declarations
  - Inheritance and dependent classes
  - Derived instances
  - The Show class
  - The Eq class
- Programs and Proofs(12)
  - Equational reasoning
  - Proofs on program equivalence
- Monads (8)
  - IO monad
  - List monad
  - Maybe monad
  - State monad

Reference Books

- Haskell - The Craft of Functional Programming: by Simon Thompson, Addison-Wesley, ISBN 0-201-34275-8
- Types and Programming Languages, by Benjamin C. Pierce; The MIT Press; ISBN 0262162091

Additional reading:

- Haskell 98 language and libraries - the Revised Report: by Simon Peyton Jones; Cambridge University Press; ISBN 0521826144

Elective Course [CS-204(N)/CS-304(N)]: Linux Kernel Programming and Introduction to Device Drivers

Objectives:

- Important concepts in OS kernel development getting hands-on Knowledge with Linux kernel
- Getting to know the device driver programming.

Pre-requisite:

- Should have done course on Advanced Operating System
- UNIX/LINUX Internals: as Operating System.
- Usage and Implementation of UNIX/LINUX System Calls

Course Contents

- Introduction to the kernel design - monolithic Vs microkernel
- Empowering the kernel - drivers and module
- Communication with the userspace
- Memory Management issues in the kernel
- Accessing Hardware
  - address space concepts
  - memory mapped and I/O
  - Virtual devices, char devices
- Handling hardware events
  - interrupt handling
  - timers and polling
- Kernel source code organization
  - Linux kernel sources and organisation
  - Browsing and understanding the kernel sources
  - Common Techniques used during Linux kernel programming
- Linux module programming
  - Hello World
  - Basic Virtual device, char device, mknod,
  - Debugging typical kernel Oops
  - strace
- Communication with the userspace
  - ioctls, proc,
- Accessing Hardware
  - address space concepts
  - memory mapped and I/O
- kobjects and sysfs
- Handling hardware events
  - interrupt handling, bottom halves, timers
  - softirq, tasklets and workqueues

- Sleeping and Locking
  - sleeping: interruptible, timed
  - locking: mutex, semaphores, spinlocks
- Exploring and developing a driver for any peripheral of one embedded device. (The college/Institute has complete freedom to include or exclude this chapter, as per the availability of proper devices. Not to be asked for any examination)

Reference books

- Linux kernel development, Robert Love, Pearson, ISBN: 9788131758182
- Professional Linux Kernel Architecture, Wolfgang Mauerer, Wrox (Wiley India), ISBN 9788126519293
- Understanding the Linux Kernel, Bovet, Cesati, Shroff/O'Reilly ISBN: 9788184040838
- Linux Device Drivers, 3rd Edition, Corbet, Rubini, Greg-Kroah Hartman , Shroff/O'Reilly , ISBN: 8173668493
- ARM System Developer's Guide: Designing and Optimizing System Software - Sloss, Symes, Wright

Elective Course [CS-204(N)/CS-304(N)]: Natural Language Processing

PREREQUISITES

1. A previous course on Artificial Intelligence will help.
2. Courses of Data Structures and Algorithms should have been done.
3. Exposure to Linguistics is useful, though not mandatory.

COURSE OUTLINE

Words and Word Forms : Morphology fundamentals; Morphological Diversity of Indian Languages; Morphology Paradigms; Finite State Machine Based Morphology; Automatic Morphology Learning; Shallow Parsing; Named Entities; Maximum Entropy Models; Random Fields.

Structures : Theories of Parsing, Parsing Algorithms; Robust and Scalable Parsing on Noisy Text as in Web documents; Hybrid of Rule Based and Probabilistic Parsing; Scope Ambiguity and Attachment Ambiguity resolution.

Meaning : Lexical Knowledge Networks, Wordnet Theory; Indian Language Wordnets and Multilingual Dictionaries; Semantic Roles; Word Sense Disambiguation; WSD and Multilinguality; Metaphors; Coreferences.

Applications of NLP: Sentiment Analysis; Text Entailment; Robust and Scalable Machine Translation; Question Answering in Multilingual Setting; Cross Lingual Information Retrieval (CLIR).

REFERENCES

1. Allen, James, Natural Language Understanding, Second Edition, Benjamin/Cumming, 1995.
2. Charniack, Eugene, Statistical Language Learning, MIT Press, 1993.
3. Jurafsky, Dan and Martin, James, Speech and Language Processing, Second Edition, Prentice Hall, 2008.
4. Manning, Christopher and Heinrich, Schutze, Foundations of Statistical Natural Language Processing, MIT Press, 1999.
5. Natural Language Processing and Text Mining, Kao, Springer, ISBN-9781846281754

Elective Course [CS-204(N)/CS-304(N)]: Program Analysis

Static analysis:

- Abstract interpretation (dataflow analysis)
- Type systems
  - simple type systems
  - type reconstruction
  - universal and existential polymorphism
  - subtyping
  - bounded quantification
  - recursive types
  - type operators
- Model checking
  - decision procedures
  - SAT solvers
  - BDDs
  - Partial Order Reduction
  - Satisfiability Modulo Theory (SMT) solvers
- Theorem-proving

Dynamic Analysis

- Automated testing and debugging using model inference
- Software visualization

Reference Books:

- Logic in Computer Science - Modelling and Reasoning About Systems: by Michael Huth and Mark Ryan; Cambridge University Press (ISBN 9780521670890)
- Principles of Program Analysis: by Nielson and Chris Hankin, Springer ISBN 978-3-540-65410-0
- Model Checking: by Clarke, Grumberg, and Peled; MIT Press
- Decision Procedures - An Algorithmic Point of View: by Kroening and Strichman; Springer
- Software Visualization: by John T. Stasko, John B. Domingue, Marc H. Brown and Blaine A. Price; MIT Press (ISBN: 978-0-262-19395-5)
- Types and Programming Languages, by Benjamin C. Pierce; The MIT Press; ISBN 0262162091

Elective Course [CS-204(N)/CS-304(N)]: DOT NET

Objectives:

- To understand the DOTNET framework, C# language features and Web development using ASP.NET
- Evaluation will be as below
  1. Theory paper : 50 marks
  2. Project work (either in C# or ASP.NET): 50 marks.
     Students are supposed to give the project demo and presentation of their project.
     Project Evaluation : (to be done internally by subject teacher)
     a. Coding : 20 Marks
     b. Documentation & Demo : 15 marks
     c. Presentation + presentation style : 15 marks

Prerequisite:

- Knowledge of object-oriented programming concepts such as data abstraction, encapsulation, inheritance, and polymorphism.
- Familiarity with programming language such as C++ and/or Java.
- Knowledge of web development

Topics to be covered:

Part I : C#

1. **DOTNET Framework (2)**
   a. Introduction to DOTNET
   b. DOT NET class framework
   c. Common Language Runtime
      i. Overview
      ii. Elements of .NET application
      iii. Memory Management
      iv. Garbage Collector : Faster Memory allocation, Optimizations
   d. Common Language Integration
      i. Common type system
      ii. Reflection API
   e. User and Program Interface
2. **Introduction to C# (8)**
   a. Language features
      i. Variables and Expressions, type conversion
      ii. Flow Control
      iii. Functions, Delegates

iv. Debugging and error handling, exception handling (System Defined and User Defined)

b. Object Oriented Concepts
   i. Defining classes, class members, Interfaces, properties
   ii. Access modifiers, Implementation of class, interface and properties
   iii. Concept of hiding base class methods, Overriding
   iv. Event Handling

c. Collections, Comparisons and Conversions
   i. Defining and using collections, Indexers, iterators
   ii. Type comparison, Value Comparison
   iii. Overloading Conversion operators, as operator

d. Generics
   i. Using generics
   ii. Defining Generics, generic Interfaces, Generic methods, Generic Delegate

3. Window Programming (6)

a. Window Controls
   i. Common Controls
   ii. Container Controls
   iii. Menus and Toolbars
   iv. Printing
   v. Dialogs

b. Deploying Window Application
   i. Deployment Overview
   ii. Visual studio setup and Deployment project types
   iii. Microsoft windows installer architecture
   iv. Building the project : Installation

4. Data Access (6)

a. File System Data
b. XML
c. Databases and ADO.NET
d. Data Binding

5. Web Programming (6)

a. Basic Web programming
b. Advanced Web programming
c. Web Services
d. Deployment Web applications

6. .NET Assemblies (3)

a. Components
b. .NET Assembly features

    c. Structure of Assemblies
    d. Calling assemblies, private and shared assemblies

7. **Networking** **(2)**
    a. Networking overview
    b. Networking programming options
        i. Webclient
        ii. WebRequest and WebResponse
        iii. TcpListener &TcpClient
8. **Introduction to GDI+** **(2)**
    a. Overview of Graphical Drawing
    b. Pen Class, Brush Class, Font Class
    c. Using Images
    d. Clipping, Drawing2D, Imaging

Part II : ASP.NET

1. **Introduction to ASP.NET** **(1)**
2. **Server Controls and Variables, control Structures & Functions (4)**
    a. Forms, webpages, HTML forms, Webforms
    b. Request & Response in Non-ASP.NET pages
    c. Using ASP.NET Server Controls
    d. Datatypes : Numeric, text, arrays, datacollections
    e. Overview of Control structures
    f. Functions : web controls as parameters
3. **Even Driven Programming and PostBack** **(3)**
    a. HTML events
    b. ASP.NET page events
    c. ASP.NET Web control events
    d. Event driven programming and postback
4. **Reading from Databases** **(3)**
    a. Data pages
    b. ADO.NET
5. **ASP.NET Server Controls** **(4)**
    a. ASP.NET Web Controls
    b. HTML Server Controls
    c. Web Controls
6. **DOTNET assemblies and Custom Controls** **(2)**
    a. Introduction to Coolies, Sessions
    b. Session events
    c. State management Recommendations
7. **Web Services** **(2)**
    a. HTTP, XML & Web services
    b. SOAP

c. Building ASP.NET web service
d. Consuming a web service

Recommended Text and Reference books:

- Beginning Visual C#, Wrox Publication
- Professional Visual C#, Wrox Publication
- Inside C#, by Tom Archer ISBN: 0735612889 Microsoft Press Â© 2001, 403 pages
- Beginning ASP.NET 3.5, Wrox Publication
- Programming ASP.NET 3.5 by Jesse Liberty, Dan Maharry, Dan Hurwitz, O'Reilly
- Illustrated C# 2008, Solis, Publication APRESS, ISBN 978-81-8128-958-2
- Professional C# 4.0 and .NET 4by Christian Nagel, Bill Evjen, Jay Glynn, Karli Watson, Morgan Skinner, WROX
- Beginning C# Object-Oriented Programming By Dan Clark , Apress
- ADO.NET Examples and Best Practices for C# Programmers, By Peter D. Blackburn Apress
- Database Programming with C#, By Carsten Thomsen, Apress

Elective Course [CS-204(N)/CS-304(N)]: Information Systems Security

Objectives of the Course:

1. To enable students to get sound understanding of Info-Sys-Security, Net Security Cryptography
2. To equip with knowledge and skills necessary to support for their career in Information Security
3. To develop attitude and interest along with necessary knowledge and skills among the students to encourage them to do further academic studies / research in this area, after the completion of their M.Sc. Course

Conceptual foundation of Information Systems Security (1)

- Concepts and Terminology
  - Threats, Attacks, Types of attacks, Programs that attack
  - Vulnerabilities, Risks, Risk Assessment and Mitigation
- Security & Elements / principles of Information Security
  - Confidentiality, Integrity, Availability, Identification, Authentication, Authorization, Accountability, Privacy

Data Encryption techniques (4)

- Introduction, Plain text, Cipher text
- Substitution techniques: Caeser cipher, Mono-alphabetic cipher, Homophonic, polygra, polyalphabetic, playfair, Hill cipher
- Transposition techniques: Reil Fense technique, simple columnar, Vernam, Book cipher
- Encryption & Decryption
- Symmetric and Asymmetric key cryptography: Diffie-Hellman key exchange
- Steganography

Symmetric / Secret Key Encryption (8)

- Algorithm Types and Modes
- Overview of symmetric key cryptography
- DES (Data Encryption Standard)
- Double DES, Triple DES
- AES (Advanced Encryption Standard)
- IDEA (International Data Encryption Algorithm)
- Blowfish
- RC4 & RC5

Asymmetric key / Public Key Encryption (4)

- History & overview of asymmetric key cryptography
- RSA algorithm
- key management
- Deffie-Hellman key exchange

- Elliptic curve cryptography

Message Integrity techniques **(4)**

- Message Digest
- MD5
- SHA
- Message Authentication Code (MAC) & HMAC
- Digital Signature techniques
    - Digital Signatures using DSA (Digital Signature Algorithm)
    - DSS (Digital Signature Standard) and RSA

Digital Certificates and PKI (Public Key Infrastructure) **(4)**

- Digital Certificates
- Private key management
- PKIX Model
- Public key cryptography standards (PKCS)
- XML, PKI and Security

Authentication techniques **(6)**

- Passwords
- Authentication Tokens
- Certificate Based Authentication
- Biometric Authentication
- Kerberos
- Key Distribution Center(KDC)
- Security Handshake Pitfalls

Internet Security protocols **(12)**

- Secure Socket Layer (SSL)
- Transport Layer Security (TLS) , Secure HTTP (SHTTP)
- Time Stamping Protocol (TSP)
- Secure Electronic Transaction (SET)
- 3-D Secure Protocol
- Electronic Money
- Email Security : SMTP, PEM, PGP, S/MIME
- Wireless Application Protocol (WAP) Security
- Security in GSM
- Security in 3G

Server Security & Firewalls **(5)**

- Intrusion Detection, IDS, Intrusion Prevention Systems (IPS)
- Introduction of Firewall, Packet Filters,
- Application Level Gateways
- Circuit Level Gateways
- Firewall architecture
- This chapter should also include detailed study of at least one free Firewall, IDS, IPS products with demonstrations (For Internal evaluation only)

Malicious Software **(2)**

- Malicious Code
- Viruses : types, working of anti-virus software
- Worms
- Trojan horse, Spyware
- Attacks: Hoax, Back-door, Brute Force, Dictionary, Spoofing, Denial-of-service, Man-in-the-middle, spam, E-mail Bombing & Spamming, Sniffer. Timing attack.

Recommended Books (Text and Reference):

1. Atul Kahate, "Cryptography And Network Security" TMH
2. William Stallings," Cryptography and Network Security" Prentice Hall /Pearson Education
3. Cryptography and Information Security By V.K. Pachghare (PHI Learning Private Limited)
4. Introduction to Computer Security By Matt Bishop and Sathyanarayana (PEARSON EDUCATION)
5. Applied Cryptography Protocols, Algorithms, and Source Code in C By Bruice Schneier (Wiley India)
6. A Classical Introduction to Cryptography, Vaudenay, Springer, 978-0-387-25464-7
7. A Classical Introduction to Cryptography Exercise Book, Baigneres, Springer, 978-0-387-27934-3

Important Links:

1. http://crsc.nist.gov/publications/nistpubs/index.html

2. Virus Bulletin: http://virusbtn.com

3. http://www.cryptool.org

Elective Course [CS-204(N)/CS-304(N)]: Software Architecture and Design Patterns

1. The Big Picture – How it all fits in? [2]

- UML → The Notation
- Process → Unified Process / Rational Unified Process inception, elaboration, construction, transition.
- How various components fit in the life cycle
- The artifacts at end of each process / discipline

2. Software Architecture: [2]

- What Software Architecture is and what it isn't.
- Why is architecture important?
- Architectural structures and views

3. Architectural Styles: [3]

- Architectural Styles
- Pipes and Filters
- Data Abstraction and Object – Oriented Organization
- Event-Based, Implicit Invocation
- Layered Systems
- Repositories
- Interpreters
- Other familiar Architectures
- Heterogeneous Architectures.

4. Patterns: [4]

- What is a Pattern & Design Pattern?
- What makes a Pattern? (GOF)
- Describing Design Patterns.
- Pattern Categories & Relationships between Patterns.
- Organizing the Catalog.
- Patterns and Software Architecture.

5. Study of Design Patterns: [18]

- Creational Patterns-singleton, factory method, abstract factory
- Structural Patterns-adapter, decorator, facade
- Behavioral Patterns-iterator, observer, strategy, command and state

(study of intent, applicability, participants, structure, collaboration and consequences)

- GRASP(General Responsibility Assignment Software Patterns : Patterns for Assigning Responsibilities
    - Expert, Creator, High Cohesion, Low Coupling, Controller, Polymorphism, Pure Fabrication, Indirection, Don't Talk to Strangers.

6. Study of Frameworks: **[6]**

- Frameworks as reusable chunks of architecture,
- The framework lifecycle, development using frameworks,
- Struts for Identify the MVC (Separation of layers)
- Configuration
- Declarative error handling
- Validation Framework
- Interaction with web application
- Case Study
- Use of Front controller & Service to worker patterns.
- Web Architectures
- Available
    - Baracudda, Webworks, Velocity, Struts etc.
- Selection of proper framework
- Comparing Frameworks
- Advantages of Struts

7. Components: **(5)**

- Development using components, composition, components as units of deployment, different approaches to components (e.g. OMG, Microsoft, Sun), developing components.

8. Case Study (struts) **(5)**

- Take a Framework and find Patterns in the Frame work.
- Benefits of Patterns in the choosen Framework
- How Pattern interact in the selected Framework.

Reference Books:

- Design Patterns – Elements of Reusable Object-oriented Software By E. Gamma, Richard Helm, Ralph Johnson , John Vlissides (GoF)
- Struts By Chuck Canvass.
- Pattern – Oriented Software Architecture (POSA) Volume 1. By: Frank Buschmann, Regine Meunier, Hans Rohnert, Peter Sommerlad, Michael Stal.
- Software Architecture in Practice. By Len Bass, Paul Clements, Rick Kazman.
- Applying UML and Patterns By Craig Larman.
- Software Architecture- Perspectives on an emerging discipline by Mary shaw and David Garlan
- Software Design Methodology: From Principles to Architectural Styles, Zhu, Elsevier ISBN:9788131203569

Elective Course [CS-204(N)/CS-304(N)]: Software Testing Tools & Methodologies

1. Preliminaries **[2]**

- Software Quality Assurance
- Software Quality
- Software Testing
- Quality Control
- Quality Assurance
- Quality Factors
- Difference between quality control and quality assurance

2. Basics of Software Testing **[4]**

- Inspection and Testing
- What is testing?
- Testing objectives
- Terms : fault, failure, error, fault masking, test, test case
- Fundamental Test process : test planning, test specification, Test execution, test records, test completion
- Prioritizing the tests
- Psychology of testing
- Difference between QA and Testing

3. Testing in the Software Lifecycle **[4]**

- The general V-model
- Component Testing
- Integration testing
- System Testing
- Acceptance Testing
- Maintenance testing

4. Software Testing Process **[4]**

- When Testing should occur?
- Requirement Phase
- Design Phase
- Program (Build) Phase
- Test Phase
- Installation Phase
- Maintenance Phase
- Testing activities
- Test Plan
- Test Development
- Test Execution
- Results
- Defect tracking
- Reports

5. Test Plan **[4]**

- Objective of the test
- Scope of the test
- Approach
- Resources
- Roles and Responsibilities
- Entry and Exit Criteria
- Schedules and mile stones
- Risks
- Defect Management
- Deliverables
- Sign off
- Case Study

6. Test Development **[4]**

- Test Case
- Good Test case
- Successful Test case
- Test case design methods
- Business logic base test case design
- Input domain base test case design
- User interface base test case design
- Common Mistakes in writing Test case
- Case Study

7. Test Execution **[2]**

- What is it?
- Why is it important?
- Who does it?
- How?

8. Defect tracking **[6]**

- Defects
- Variance from product specifications
- Variance from customer/user expectation
- Purpose for recording defects
- Severity versus Priority
- What should be done after a bug is found?
- Defect Classification
- Defect Severity
- Defect Priority
- Defect log example

9. Test Metrics **[4]**

- Functional or Test Coverage Metric
- Function Test Metric
- Software Release Metrics
- Software Maturity Metric
- Reliability Metric

10. Test Reports **[4]**

- Interim Reports
- Functional Testing Status
- Functions Working Timeline
- Expected verses Actual Defects Detected
- Defects Detected verses Corrected Gap
- Defect Distribution
- Relative Defect Distribution
- Testing Actions
- Final Test Report

11. Levels of Testing **[4]**

- Levels of Testing
- Entry and exit Criteria for each level of testing
- Integration Testing
- System Testing
- User Acceptance Testing

12. Software Testing Techniques **[6]**

- Static and Dynamic Testing
- White Box Testing
- Basis Path Testing
- Control Structure Testing
- Functional Testing
    - Black Box Testing
        - Equivalence Class Testing
        - Boundary Value Testing
        - Comparison Testing
        - Graph based Testing
    - Incremental Integration Testing
    - Integration Testing
    - Regression Testing
    - Smoke Testing
    - Alpha Testing
    - Beta Testing
    - System Testing
    - Recovery Testing
    - Security Testing
    - Sanity Testing
    - End-to-End Testing
    - User Acceptance Testing
    - Usability Testing

        - Compatibility Testing
        - Install/Uninstall Testing
- Non-Functional Testing
- Performance Testing
    - Load Testing
    - Stress Testing
    - Endurance Testing
    - Robustness Testing
    - Scalability Testing
    - Case examples on each type of testing

13. Test Tools **[2]**

- Types of test tools
- Selection and use of test tools

References:

- Cem Kaner, Jack Falk, and Hung Quoc Nguyen, *Testing Computer Software, Second edition*, Wiley, New York, 1999.
- Practical Software Testing: A Process-Oriented Approach, Burnstein, Springer, ISBN 978-81-8128-089-3
- Edward Kit, *Software Testing in the Real World: Improving the Process,* Addison Wesley, 1995.
- Glenford J. Myers, *The Art of Software Testing,* Wiley, New York, 1979.
- Elfriede Dustin, Jeff Rashka, and John Paul, *Automated Software Testing: Introduction, Management, and Performance,* Addison Wesley, Reading, Mass., 1999.
- Frank P. Ginac, *Customer Oriented Software Quality Assurance,* Prentice-Hall, Upper Saddle River, NJ, 1998.
- Alka Jarvis and Vern Crandall, *Inroads to Software Quality: "How To" Guide and Toolkit,* Prentice-Hall, Upper Saddle River, NJ, 1997.
- Stephen H. Kan, *Metrics and Models in Software Quality Engineering* Addison Wesley, Reading, Mass., 1995.
- Michael R. Lyu, Ed., *Handbook of Software Reliability Engineering* McGraw-Hill, New York, 1996.
- William E. Perry, *How to Test Software Packages,* Wiley, New York, 1986.
- Stephen R. Schach, *Classical and Object-Oriented Software Engineering with UML and Java, Fourth edition*, McGraw-Hill, Boston, 1999. Textbook for COSC 158 Systems Analysis.
- Robert L. Baber, *Error-Free Software: Know-how and Know-why of Program Correctness,* Wiley, New York, 1991. Program proving.
- Richard Fairley, *Software Engineering Concepts,* McGraw-Hill, New York, 1985.
- John Musa, *Software Reliability Engineering,* McGraw-Hill, New York, 1989. Assumes an organization at least at CMM level 3.
- Jerry Peek, Tim O'Reilly, and Mike Loukides, *Unix Power Tools,* O'Reilly, Sebastopol, CA. Especially SCCS, p. 366.
- Thomas C. Royer, Software Testing Management -- Life on the Critical Path, Prentice Hall, Englewood Cliffs, NJ, 1993.
- Edward Yourdon, *Structured Walkthroughs,* Yourdon Press, Englewood Cliffs, NJ, 1989.

Elective Course [CS-204(N)/CS-304(N)]: MODELLING AND SIMULATION

1. SIMULATION CONCEPTS

Systems, modeling, general system theory, concept of simulation, simulation as a decision making tool, types of simulation. [3]

2. Random numbers

Pseudo random numbers, methods of generating random varieties, discrete and continuous distributions, testing of random numbers. [5]

3. Design of simulation experiments

problem formulation , data collection and reduction, time flow mechanism , key variables, logic flow chart, starting condition, run size, experimentaldesign consideration, output analysis and interpretation validation. [8]

4. Simulation language

comparison, and selection of simulation languages, study of any one simulation language. [14]

5. Case Studies

Development simulation models using the simulation language studied for systems like queuing systems, production systems, inventory systems. [15]

BOOKS:

1. Simulation, 4e, Ross, Elsevier, ISBN-9788131214626
2. Theory of Modeling and Simulation, 2e, Zeigler, Elsevier, ISBN-9788131207406
3. Modeling and Simulation: Exploring Dynamic System Behaviour, Birta, Springer, IBSN 978-81-8489-365-6
4. Jerry Banks and John, S. Carson, "Discrete Event System Simulation" PHI
5. Shannon, R.E., "Systems Simulation, The Art and Science" , PHI

Elective Course [CS-204(N)/CS-304(N)]: EMBEDDED SYSTEM PROGRAMMING

Introduction to ES **[2]**

- What is ES?
- Examples of ES
- Inside ES : processor, memory, peripherals, software

Embedded Processors, Memories & Peripherals **[8]**

- Microcontrollers 8051
- Discrete processors : 8-bit architecture, 16/32 bit CISC, RISC, DSP
- Integrated processors : ARM RISC
- Choosing a processor
- Memory systems : types (SRAM, DRAM, FLASH), organization, access time, validating the contents of memory
- Basic peripherals : parallel ports, timers, clocks

Real time system concepts **[10]**

- Foreground/ background systems
- Critical section of code
- Resource, shared resource
- Multitasking, task, task switch
- Kernel, scheduler, non-preemptive kernel, preemptive kernel
- Reentrancy, round-robin scheduling
- Task priority, static priority, dynamic priority, priority inversions, assigning task priorities
- Mutual exclusion, deadlock, synchronization, event flags, intertask communication
- Interrupts : latency, response, recovery, ISR processing time, NMI
- *Note: For 'C' implementation of above concepts, please refer to chapters 4,5,6,7 of the book "An Embedded Software Primer" by David E. Simon published by Pearson Educations*

Writing software for embedded systems **[8]**

- The compilation process : compile, link, load
- Cross compilers
- Run-time-libraries : processor dependent, I/O dependent, system calls, exit routines
- Writing a library, using alternative libraries
- Porting Kernels
- C extensions for embedded systems
- Buffering and other data structures
  Linear buffers, Directional buffers, double buffering, Buffer exchange, Linked lists, FIFO, Circular buffers, Buffer underrun and overrun, Allocating buffer memory, Buffer leakage
- Downloading

Emulation and Debugging techniques **[8]**

- Debugging techniques: HLL simulation, low level simulation, on-board debugger, task level debugging, symbolic debug
- Emulation
- Optimization problems

Basic design using RTOS **[6]**

- Overview
- Principles
- Example

- Encapsulating semaphores and queues
- Hard real time scheduling considerations
- Saving memory space
- Saving power

Real time without RTOS **[8]**

- Choosing the SW environment
- Deriving real time performance from non-real time system
- Scheduling and data sampling
- Controlling from an external switch
- Problems

<u>Reference books:</u>

1. Embedded Systems Design, 2e, Heath, Elsevier, ISBN:9788181479709
2. Embedded Systems Design with FPGAs, Saas, Elsevier, ISBN: 9789380501918
3. Programming Embedded Systems – Michael Barr
4. Embedded Systems Building Blocks _ Jean J. Labrosse
5. An Embedded Software Primer _ David E. Simon published by Pearson Educations

Elective Course [CS-204(N)/CS-304(N)]: Language Processors

Prerequisites –

- System programming concepts
- Detailed knowledge of : DFA, NFA, Regular expressions and regular languages (Scanning), Context Free grammars, Parsing (Top Down and Bottom up Parsing), Syntax Directed Translation(SDT)
- Concepts of Code Generation and Optimization.

Objectives :

The course should make a student to be in a position to design a small compiler and implement it.

Course contents –

1. Classic theory of compilers –
    - Scanning (Lexical analysis) **(4)**
        - It is expected to cover Lex utility in depth and the student should be able to design the lexical analyzer for any given language set. While designing the lexical analyzer, the paper work should be done using the concepts of DFA,NFA, RE etc and then the lex program to be written.
        - Scanning in 2 commercial compilers : case study
    - Parsing (Syntax analysis) **(4)**
        - It is expected to cover YACC utility in depth and the student should be able to design a LALR parser for the given language set and that should call a lex utility for the token separation. While designing the parser, paper works should be done using parse tree creation.
        - Parsing in 2 commercial compilers : case study
    - Semantic analysis **(5)**
        - Attributes and attribute grammars
        - Algorithms for attribute computations
        - Symbol table
        - Data types and type checking
        - Semantic analysis in 2 commercial compilers : case study
    - Runtime Environments **(6)**
        - Memory organization during program execution
        - Fully static runtime environments
        - Stack based runtime environments
        - Dynamic memory
        - Parameter passing mechanisms
    - Code generation **(7)**
        - Intermediate code and data structures for code generation
        - Basic code generation techniques

        - Code generation -
            - Of data structure references
            - Of control statements and logical expressions
            - Of procedure and function calls
        - Code generation in 2 commercial compilers : case study
    - Optimization and data flow analysis **(10)**
        - Principal source of optimization
        - Optimization of basic blocks
        - Loops in flow graphs
        - Global data flow analysis
        - Interactive solution of data flow equations
        - Code improving transformations
        - Dealing with aliases
        - Data flow analysis of structures flow graphs
        - Efficient data flow analysis
        - A tool for data flow analysis
        - Estimation of types
        - Symbolic debugging of optimized code.

2. Implementation of TINY sample language based on each topic –

    - Scanning (Lexical analysis) : TINY sample language and Compiler and implementation of TINY Compiler **(3)**
    - Parsing (Syntax analysis) : Syntax of TINY Language, RDP for TINY Language and generation of TINY parser using YACC **(3)**
    - Semantic analysis : A semantic analyzer for TINY language **(4)**
    - Code generation : Simple Optimizations for TINY code generator **(5)**

    (The language details and the guidance regarding each of the above points should be given the lectures.)

Reference books –

1. Engineering-A Compiler, 2e, Cooper, Elsevier, ISBN:9789380931876
2. Advanced Compiler Design and Implementation, Muchnick,Elsevier,ISBN-9788131214039
3. Compiler Construction by Loudan,
4. Lex and Yacc by O'Really publications
5. Compiler Design using FLEX and YACC by DAS, PHI
6. Compilers: Principles, Techniques, and Tools By Aho, Sethi, Ullman

Elective Course [CS-204(N)/CS-304(N)]: Artificial Intelligence

Prerequisites –

- Concepts of Data structures and Design and Analysis of algorithms

Objectives-

- To understand and gain the knowledge of the subject

Course contents –

1. Introduction to Artificial Intelligence
   - What is AI?
   - Early work in AI
   - AI and related fields
   - AI problems and Techniques
2. Problems, Problem Spaces and Search
   - Defining AI problems as a State Space Search: example
   - Production Systems
   - Search and Control Strategies
   - Problem Characteristics
   - Issues in Design of Search Programs
   - Additional Problems
3. Heuristic Search Techniques
   - Generate-and-test
   - Hill Climbing
   - Best First Search
   - Problem Reduction
   - Constraint Satisfaction
   - Mean-Ends Analysis
4. Knowledge Representation
   - Representations and Mappings
   - Approaches to Knowledge Representation
   - Knowledge representation method
   - Propositional Logic
   - Predicate logic
   - Representing Simple facts in Logic
   - Representing Instances and Isa relationships
   - Computable Functions and Predicates
   - Resolution
   - Forward and backward chaining
5. Slot – and – Filler Structures
   - Weak Structures
   - Semantic Networks
   - Frames
   - Strong Structures
   - Conceptual Dependencies
   - Scripts
6. Game Playing
   - Minimax Search Procedures
   - Adding alpha-beta cutoffs

- Uncertianty Reasoning: Basic Probabilty Axioms, Baye's
  Rule, Baysian Classification, Certainty Factor Theory, Dempster Shafar
  Theory.

7. Learning
   - What is learning?
   - Rote Learning
   - Learning by taking advice
   - Learning in problem solving
   - Learning from examples
   - Explanation based learning

Internal evaluation

- To implement the AI concepts using programming language PROLOG.

Reference books –

1. Computational Intelligence, Eberhart, Elsevier, ISBN 9788131217832
2. Artificial Intelligence: A New Synthesis, Nilsson, Elsevier, ISBN 9788181471901
3. Artificial Intelligence, Tata McGraw Hill, 2nd Edition, by Elaine Rich and Kevin Knight
4. Introduction to Artificial Intelligence and Expert System, Prentice Hall of India Pvt. Ltd., New Delhi, 1997, 2nd Printing, by Dan Patterson.