Objectives:

- To briefly describe various programming paradigms.
- To provide conceptual understanding of High level language design and implementation.
- To introduce the power of scripting languages.

UNIT I:

UNIT II:
**Data types:** Introduction, primitive, character, user defined, array, associative, record, union, pointer and reference types, design and implementation uses related to these types. Names, Variable, concept of binding, type checking, strong typing, type compatibility, named constants, variable initialization. **Expressions and Statements:** Arithmetic relational and Boolean expressions, Short circuit evaluation mixed mode assignment, Assignment Statements, Control Structures – Statement Level, Compound Statements, Selection, Iteration, Unconditional Statements, guarded commands.

UNIT III:
**Subprograms and Blocks:** Fundamentals of sub-programs, Scope of life time of varaibles, static and dynamic scope, design issues of subprograms and operations, local referencing environments, parameter passing methods, overloaded sub-programs, generic sub-programs, parameters that are sub-program names, design issues for functions user defined overloaded operators, co routines.

UNIT IV:
**Abstract Data types:** Abstractions and encapsulation, introductions to data abstraction, design issues, language examples, C++ parameterized ADT, object oriented programming in small talk, C++, Java, C#, Ada 95
Concurrency: Subprogram level concurrency, semaphores, monitors, massage passing, Java threads, C# threads.

Exception handling: Exceptions, exception Propagation, Exception handler in Ada, C++ and Java.

Logic Programming Language: Introduction and overview of logic programming, basic elements of prolog, application of logic programming.

UNIT V:

Functional Programming Languages: Introduction, fundamentals of FPL, LISP, ML, Haskell, application of Functional Programming Languages and comparison of functional and imperative Languages.


TEXT BOOKS:


REFERENCE BOOKS:

3. LISP Patric Henry Winston and Paul Horn Pearson Education.

Outcomes:

- Ability to express syntax and semantics in formal notation.
- Ability to apply suitable programming paradigm for the application.
- Gain knowledge and comparison of the features programming languages.
Objectives: This introductory course input is intended.

a. To help the students appreciate the essential complementarity between 'VALUES' and 'SKILLS' to ensure sustained happiness and prosperity which are the core aspirations of all human beings.
b. To facilitate the development of a Holistic perspective among students towards life, profession and happiness, based on a correct understanding of the Human reality and the rest of Existance. Such a holistic perspective forms the basis of Value based living in a natural way.
c. To highlight plausible implications of such a Holistic understanding in terms of ethical human conduct, trustful and mutually satisfying human behaviour and mutually enriching interaction with Nature.

UNIT - I:

Course Introduction - Need, basic Guidelines, Content and Process for Value Education: Understanding the need, basic guidelines, content and process for Value Education. Self Exploration - what is it? - its content and process; 'Natural Acceptance' and Experiential Validation - as the mechanism for self exploration. Continuous Happiness and Prosperity - A look at basic Human Aspirations. Right understanding, Relationship and Physical Facilities - the basic requirements for fulfillment of aspirations of every human being with their correct priority. Understanding Happiness and Prosperity correctly - A critical appraisal of the current scenario. Method to fulfill the above human aspirations: understanding and living in harmony at various levels.

UNIT - II:

Understanding Harmony in the Human Being - Harmony in Myself! : Understanding human being as a co-existence of the sentient 'I' and the material 'Body'. Understanding the needs of Self (T) and 'Body' - Sukh and Suvidha. Understanding the Body as an instrument of 'T' ( I being the doer, seer and enjoyer). Understanding the harmony of I with the Body: Sanyam and Swasthya; correct appraisal of Physical needs, meaning of Prosperity in detail. Programs to ensure Sanyam and Swasthya.

UNIT - III:

Understanding Harmony in the Family and Society - Harmony in Human - Human Relationship: Understanding harmony in the Family the basic unit of human interaction. Understanding values in human relationship; meaning of Nyaya and program for its fulfillment to ensure Ubhay-tripti; Trust (Vishwas) and Respect (Samman) as the foundational values of relationship. Understanding the meaning of Vishwas; Difference between intention and competence. Understanding the meaning of Samman, Difference between
respect and differentiation; the other salient values in relationship. Understanding the harmony in the society (society being an extension of family): Samadhan, Samridhi, Abhay, Sah-astiva as comprehensive Human Goals. Visualizing a universal harmonious order in society - Undivided Society (Akhand Samaj), Universal Order (Sarvabhaum Vyawastha) - from family to world family!

UNIT - IV:

Understanding Harmony in the nature and Existence - Whole existence as Co-existence: Understanding the harmony in the Nature. Interconnectedness and mutual fulfillment among the four orders of nature - recyclability and self-regulation in nature. Understanding Existence as Co-existence (Sah-astiva) of mutually interacting units in all-pervasive space. Holistic perception of harmony at all levels of existence.

UNIT - V:

Implications of the above Holistic Understanding of Harmony on Professional Ethics: Natural acceptance of human values, Definitiveness of Ethical Human Conduct, Basic for Humanistic Education, Humanistic Constitution and Humanistic Universal Order. Competence in professional ethics:

a. Ability to utilize the professional competence for augmenting universal human order,

b. Ability to identify the scope and characteristics of people-friendly and eco-friendly production systems,

c. Ability to identify and develop appropriate technologies and management patterns for above production systems.

Case studies of typical holistic technologies, management models and production systems. Strategy for transition from the present state to Universal Human Order.

a. At the level of individual: as socially and ecologically responsible engineers, technologists and managers

b. At the level of society: as mutually enriching institutions and organizations.

TEXT BOOKS:


REFERENCE BOOKS:

UNIT - I:
Introduction to Intellectual property: Introduction, types of intellectual property, international organizations, agencies and treaties, importance of intellectual property rights.

UNIT - II:
Trade Marks: Purpose and function of trade marks, acquisition of trade mark rights, protectable matter, selecting and evaluating trade mark, trade mark registration processes.

UNIT - III:
Law of copy rights: Fundamental of copy right law, originality of material, rights of reproduction, rights to perform the work publicly, copy right ownership issues, copy right registration, notice of copy right, international copy right law.

Law of patents: Foundation of patent law, patent searching process, ownership rights and transfer

UNIT - IV:
Trade Secrets: Trade secrete law, determination of trade secrete status, liability for misappropriations of trade secrets, protection for submission, trade secrete litigation.

Unfair competition: Misappropriation right of publicity, False advertising.

UNIT - V:
New development of intellectual property: new developments in trade mark law; copy right law, patent law, intellectual property audits.

International overview on intellectual property, international - trade mark law, copy right law, international patent law, international development in trade secrets law.

TEXT BOOKS & REFERENCES:

1. Intellectual property right, Deborah, E. Bouchoux, cengage learning.
(Open Elective)

UNIT - I:


UNIT - II:

**Types of Environmental hazards & Disasters:** Natural hazards and Disasters - Man induced hazards & Disasters - Natural Hazards - Planetary Hazards / Disasters - Extra Planetary Hazards / disasters - Planetary Hazards - Endogenous Hazards - Exogenous Hazards

UNIT - III:


UNIT - IV:

Exogenous hazards / disasters - Infrequent events - Cumulative atmospheric hazards / disasters

**Infrequent events:** Cyclones - Lightning - Hailstorms

**Cyclones:** Tropical cyclones & Local stroms - Destruction by tropical cyclones & local stroms (causes, distribution human adjustment, perception & mitigation) Cumulative atmospheric hazards/ disasters :- Floods - Droughts - Cold waves - Heal waves Floods :- Causes of floods - Flood hazards India - Flood control measures (Human adjustment, perception & mitigation) Droughts :- Impacts of drougths - Drought hazards in India - Drought control measures - Extra Planetary Hazards / Disasters - man induced Hazards / Disasters - Physical hazards / Disasters - Soil erosion

**Soil Erosion:** Mechanics & forms of Soil Erosion - Factors 7 causes of Soil Erosion - Conservation measures of Soil Erosion.

**Chemical hazards / disasters:** Release of toxic chemicals, nuclear explosion - Sedimentation processes Sedimentation processes :- Global Sedimentation problems - Regional Sedimentation problems - Sedimentation & Environmental problems - Corrective measures of Erosion & Sedimentation

**Biological hazards / disasters:** Population Explosion.
UNIT - V:

Emerging approaches in Disaster Management - Three stages

1. Pre-disaster Stage (preparedness)
2. Emergency Stage
3. Post Disaster stage - Rehabilitation

TEXT BOOKS:

1. Disaster Mitigation: Experiences And Reflections by Pradeep Sahni
2. Natural Hazards & Disasters by Donald Hyndman & David Hyndman - Cengage Learning

REFERENCES:

1. R. B. Singh (Ed) Environmental Geography, Heritage Publishers New Delhi, 1990
2. Savinder Singh Environmental Geography, Prayag Pustak Bhawan 1997
4. R. B. Singh (Ed) Disaster Management, Rawat Publication, New Delhi, 2000
6. R. B. Singh, Space Technology for Disaster Mitigation in India (INCED), University of Tokyo, 1994
7. Dr. Satender, Disaster Management in Hills, Concept Publishing Co., New Delhi, 2003
9. R. K. Bhandani An overview on Natural & Man made Disaster & their Reduction, CSIR, New Delhi
Objectives:

- To understanding of software process models such as waterfall and evolutionary models.
- To understanding of software requirements and SRS document.
- To understanding of different software architectural styles.
- To understanding of software testing approaches such as unit testing and integration testing.
- To understanding on quality control and how to ensure good quality software.

UNIT - I:

**Introduction to Software Engineering:** The evolving role of software, Changing Nature of Software, Software myths.

**A Generic view of process:** Software engineering- A layered technology, a process framework, The Capability Maturity Model Integration (CMMI), Process patterns, process assessment, personal and team process models.

**Process models:** The waterfall model, Incremental process models, Evolutionary process models, The Unified process.

UNIT - II:

**Software Requirements:** Functional and non-functional requirements, User requirements, System requirements, Interface specification, the software requirements document.

**Requirements engineering process:** Feasibility studies, Requirements elicitation and analysis, Requirements validation, Requirements management.

**System models:** Context Models, Behavioral models, Data models, Object models, structured methods.

UNIT - III:

**Design Engineering:** Design process and Design quality, Design concepts, the design model.

**Creating an architectural design:** Software architecture, Data design, Architectural styles and patterns, Architectural Design.

**Object-Oriented Design:** Objects and object classes, An Object-Oriented design process, Design evolution

**Performing User interface design:** Golden rules, User interface analysis and design, interface analysis, interface design steps, Design evaluation.
UNIT - IV:

**Testing Strategies**: A strategic approach to software testing, test strategies for conventional software, Black-Box and White-Box testing, Validation testing, System testing, the art of Debugging.

**Product metrics**: Software Quality, Metrics for Analysis Model, Metrics for Design Model, Metrics for source code, Metrics for testing, Metrics for maintenance.


UNIT - V:


**TEXT BOOKS**:


**REFERENCE BOOKS**:

3. Fundamentals of Software Engineering, Rajib Mall, PHI, 2005
10. Introduction to Software Engineering, R. J. Leach, CRC Press.

**Outcomes**:

- Ability to identify the minimum requirements for the development of application.
- Ability to develop, maintain, efficient, reliable and cost effective software solutions.
- Ability to critically thinking and evaluate assumptions and arguments.
Objectives:

- To describe the steps and algorithms used by language translators.
- To discuss the effectiveness of optimization.
- To explain the machine dependent aspects of Compilation

UNIT – I

Overview of Compilation: Phases of Compilation – Lexical Analysis, Regular Grammar and regular expression for common programming language features, pass and Phases of translation, interpretation, bootstrapping, data structures in compilation – LEX lexical analyzer generator.

Top down Parsing: Context free grammars, Top down parsing – Backtracking, LL (1), recursive descent parsing, Predictive parsing, Preprocessing steps required for predictive parsing.

UNIT – II

Bottom up parsing: Shift Reduce parsing, LR and LALR parsing, Error recovery in parsing, handling ambiguous grammar, YACC – automatic parser generator.

UNIT – III

Semantic analysis: Intermediate forms of source Programs – abstract syntax tree, polish notation and three address codes. Attributed grammars, Syntax directed translation, Conversion of popular Programming languages language Constructs into Intermediate code forms, Type checker.

Symbol Tables: Symbol table format, organization for block structures languages, hashing, tree structures representation of scope information. Block structures and non block structure storage allocation: static, Runtime stack and heap storage allocation, storage allocation for arrays, strings and records.

UNIT – IV

Code optimization: Consideration for Optimization, Scope of Optimization, local optimization, loop optimization, frequency reduction, folding, DAG representation.

Data flow analysis: Flow graph, data flow equation, global optimization, redundant sub expression elimination, Induction variable elements, Live variable analysis, Copy propagation.

UNIT – V

Object code generation: Object code forms, machine dependent code optimization, register allocation and assignment generic code generation algorithms, DAG for register allocation.
TEXT BOOKS:


REFERENCES:

1. lex & yacc – John R. Levine, Tony Mason, Doug Brown, O’reilly

Outcomes:

- Ability to understand the design of a compiler given features of the languages.
- Ability to implement practical aspects of automata theory.
- Gain Knowledge of powerful compiler generation tools.
Objectives:

- To understand main components of OS and their working
- To study the operations performed by OS as a resource manager
- To understand the different scheduling policies of OS
- To understand the different memory management techniques
- To understand process concurrency and synchronization
- To understand the concepts of input/output, storage and file management
- To study different OS and compare their features.

UNIT - I:


UNIT - II:


UNIT - III:

UNIT - IV:


UNIT - V:

Deadlocks - System Model, Deadlock Characterization, Methods for Handling Deadlocks, Deadlock Prevention, Deadlock Avoidance, Deadlock Detection and Recovery from Deadlock.


TEXT BOOKS:


REFERENCES BOOKS:

5. An Introduction to Operating Systems, P.C.P. Bhatt, PHI.

Outcome:

- Apply optimization techniques for the improvement of system performance.
- Ability to understand the synchronous and asynchronous communication mechanisms in their respective OS.
- Learn about minimization of turnaround time, waiting time and response time and also maximization of throughput with keeping CPU as busy as possible.
- Ability to compare the different OS
Objectives:

- To introduce the fundamental various types of computer networks.
- To demonstrate the TCP/IP and OSI models with merits and demerits.
- To introduce UDP and TCP Models.

UNIT - I:


Physical Layer: Guided transmission media, wireless transmission media.

Data Link Layer - design issues, CRC codes, Elementary Data Link Layer Protocols, sliding window protocol

UNIT - II:

Multi Access Protocols - ALOHA, CSMA, Collision free protocols, Ethernet- Physical Layer, Ethernet Mac Sub layer, data link layer switching & use of bridges, learning bridges, spanning tree bridges, repeaters, hubs, bridges, switches, routers and gateways.

UNIT - III:

Network Layer: Network Layer Design issues, store and forward packet switching connection less and connection oriented networks-routing algorithms-optimality principle, shortest path, flooding, Distance Vector Routing, Control to Infinity Problem, Hierarchical Routing, Congestion control algorithms, admission control.

UNIT - IV:

Internetworking: Tunneling, Internetwork Routing, Packet fragmentation, IPv4, IPv6 Protocol, IP addresses, CIDR, IMCP, ARP, RARP, DHCP.

Transport Layer: Services provided to the upper layers elements of transport protocol-addressing connection establishment, connection release, Connection Release, Crash Recovery.
UNIT - V:


Application Layer- Introduction, providing services, Applications layer paradigms, Client server model, Standard client-server application- HTTP, FTP, electronic mail, TELNET, DNS, SSH

TEXT BOOKS:


REFERENCES BOOKS:


Outcomes:

- Students should be understand and explore the basics of Computer Networks and Various Protocols. He/She will be in a position to understand the World Wide Web concepts.
- Students will be in a position to administrate a network and flow of information further he/she can understand easily the concepts of network security, Mobile, and ad hoc networks.
Objectives:

- To use linux operating system for study of operating system concepts.
- To write the code to implement and modify various concepts in operating systems using Linux.

List of Programs:

1. Simulate the following CPU scheduling algorithms
   a. Round Robin
   b. SJF
   c. FCFS
   d. Priority
2. Simulate all file allocation strategies
   a. Sequential
   b. Indexed
   c. Linked
3. Simulate MVT and MFT
4. Simulate all File Organization Techniques
   a. Single level directory
   b. Two level
   c. Hierarchical
   d. DAG
5. Simulate Bankers Algorithm for Dead Lock Avoidance
6. Simulate Bankers Algorithm for Dead Lock Prevention
7. Simulate all page replacement algorithms
   a. FIF
   b. LRU
   c. LFU etc.
8. Simulate Paging technique of memory management.

Outcomes:

- The course objectives ensure the development of students applied skills in operating systems related areas.
- Students will gain knowledge in writing software routines modules or implementing various concepts of operating system.
Objectives:

- To provide an understanding of the language translation peculiarities by designing a complete translator for a mini language.

Recommended System / Software Requirements:

- Intel based desktop PC with minimum of 166 MHZ or faster processor with at least 64 MB RAM and 100 MB free disk space
- C++ compiler and JDK kit

Consider the following mini Language, a simple procedural high-level language, only operating on integer data, with a syntax looking vaguely like a simple C crossed with Pascal. The syntax of the language is defined by the following BNF grammar:

<program> ::= <block>

<block> ::= { <variabledefinition><slist> } | { <slist> }

<variabledefinition> ::= int<vardeflist>;

<vardeflist> ::= <vardec> | <vardec>, <vardeflist>

<vardec> ::= <identifier> | <identifier> [ <constant> ]

<slist> ::= <statement> | <statement>; <slist>

<statement> ::= <assignment> | <ifstatement> | <whilestatement> | <block> | <printstatement> | <empty>

<assignment> ::= <identifier> = <expression> | <identifier> [ <expression> ] = <expression>

<ifstatement> ::= <bexpression> then <slist> else <slist> endif | if <bexpression> then <slist> endif

<whilestatement> ::= while <bexpression> do <slist> enddo

<printstatement> ::= print ( <expression> )

<expression> ::= <expression><additionop><term> | <term> | additionop<term>
<bexpression> ::= <expression><relop><expression>

<relop> ::= < | <= | == | >= | > | !=

<addingop> ::= + | -

<term> ::= <term><mulitop><factor> | <factor>

<multop> ::= * | /

<factor> ::= <constant> | <identifier> | <identifier> [ <expression> ] | ( <expression> )

<constant> ::= <digit> | <digit><constant>

<identifier> ::= <identifier><letterordigit> | <letter>

<letterordigit> ::= <letter> | <digit>

<letter> ::= a|b|c|d|e|f|g|h|i|j|k|l|m|n|o|p|q|r|s|t|u|v|w|x|y|z

<digit> ::= 0|1|2|3|4|5|6|7|8|9

<empty> has the obvious meaning

Comments (zero or more characters enclosed between the standard C / Java style comment brackets /*...*/) can be inserted. The language has rudimentary support for 1-dimensional arrays. The declaration

int a[3] declares an array of three elements, referenced as a[0], a[1] and a[2] Note also that you should worry about the scoping of names.

A simple program written in this language is:

{ int a[3], t1, t2; t1 = 2; a[0] = 1; a[1] = 2; a[t1] = 3; t2 = -(a[2] + t1 * 6)/ a[2] - t1); if t2 > 5 then print(t2); else { int t3; t3 = 99; t2 = -25; print(-t1 + t2 * t3); /* this is a comment on 2 lines */}
1. Design a Lexical analyzer for the above language. The lexical analyzer should ignore redundant spaces, tabs and newlines. It should also ignore comments. Although the syntax specification states that identifiers can be arbitrarily long, you may restrict the length to some reasonable value.

2. Implement the lexical analyzer using JLex, flex or lex or other lexical analyzer generating tools.

3. Design Predictive parser for the given language.

4. Design LALR bottom up parser for the above language.

5. Convert the BNF rules into Yacc from and write code to generate abstract syntax tree.

6. Write program to generate machine code from the abstract syntax tree generated by the parser. The following instruction set may be considered as target code.

The following is a simple register-based machine, supporting a total of 17 instructions. It has three distinct internal storage areas. The first is the set of 8 registers, used by the individual instructions as detailed below, the second is an area used for the storage of variables and the third is an area used for the storage of program. The instructions can be preceded by a label. This consists of an integer in the range 1 to 9999 and the label is followed by a colon to separate it from the rest of the instruction. The numerical label can be used as the argument to a jump instruction, as detailed below.

In the description of the individual instructions below, instruction argument types are specified as follows:

R specifies a register in the form R0, R1, R2, R3, R4, R5, R6 or R7 (or r0, r1, etc).

L specifies a numerical label (in the range 1 to 9999).

V specifies a "variable location" (a variable number, or a variable location pointed to by a register - see below).

A specifies a constant value, a variable location, a register or a variable location pointed to by a register (an indirect address). Constant values are specified as an integer value, optionally preceded by a minus sign, preceded by a # symbol. An indirect address is specified by an @ followed by a register.

So, for example an A-type argument could have the form 4 (variable number 4), #4 (the constant value 4), r4 (register 4) or @r4 (the contents of register 4 identifies the variable location to be accessed).

The instruction set is defined as follows:

LOAD A, R
loads the integer value specified by A into register R.
STORE R, V
stores the value in register R to variable V.
OUT R
outputs the value in register R.
NEG R
negates the value in register R.
ADD A, R
adds the value specified by A to register R, leaving the result in register R.
SUB A, R
subtracts the value specified by A from register R, leaving the result in register R.
MUL A, R
multiplies the value specified by A by register R, leaving the result in register R.
DIV A, R
divides register R by the value specified by A, leaving the result in register R.
JMP L
causes an unconditional jump to the instruction with the label L.
JEQ R, L
jumps to the instruction with the label L if the value in register R is zero.
JNE R, L
jumps to the instruction with the label L if the value in register R is not zero.
JGE R, L
jumps to the instruction with the label L if the value in register R is greater than or equal to zero.
JGT R, L
jumps to the instruction with the label L if the value in register R is greater than zero.
JLE R, L
jumps to the instruction with the label L if the value in register R is less than or equal to zero.
JLT R, L
jumps to the instruction with the label L if the value in register R is less than zero.
NOP
is an instruction with no effect. It can be tagged by a label.
STOP
stops execution of the machine. All programs should terminate by executing a STOP instruction.

Outcomes:

- By this laboratory, students will understand the practical approach of how a compiler works.
- This will enable him to work in the development phase of new computer languages in industry.