Annexure 5

SYLLABUS STRUCTURE

(Effective from academic session 2023-24)

For the Degree of

Bachelor of Computer Applications (BCA Honours)

(Software Product Engineering)

in Association with Industry Partner

Eight-Semester Full Time Programme

ELIGIBILITY OF THE CANDIDATES:

The candidate must have passed 10+2 or A level or IB examination in any discipline with at least 50% marks in aggregate.

Program specific outcomes for Bachelor of Computer applications (BCA) program: At the end of this program, the students will be able:

[PSO1]: To apply the principles of software engineering and design to develop, test, and maintain high-quality software products.

[PSO2]: To effectively communicate and collaborate with team members, stakeholders, and clients to develop and deliver software products that meet their needs and requirements.

[PSO3]: To apply critical thinking, problem-solving skills, and ethical considerations to address complex issues related to software development, deployment, and maintenance.

Year	First Semester						Second Semester					
	Course Code	Course Name	L	T	P	C	Course Code	Course Name	L	T	P	C
I	XXXXXXX	Fundamentals of Discrete Mathematics	3	1	0	4	CAH1201	Full Stack Web Development	2	4	0	6
	CAH1101	C, C++ and Python - Problem solving using Programming	3	1	0	4	CAH1202	Professional skills for the workplace	3	1	0	4
	CAH1102	Critical Thinking	3	1	0	4	CAH1203	Learning how to learn	3	1	0	4
	CAH1103	Fundamentals of Computers & Digital Systems - Breadth of computer science	3	1	0	4	CAH1230	Full Stack Web Development Lab	0	0	8	4
	CAH1104	Introduction to Web Programming - Front end development	3	1	0	4			8	5	8	18
	CAH1105	Design for Developers	3	0	0	3	Total Contact hours (L + T + P)		21			
	CAH1130	C, C++ and Python - Problem solving using Programming Lab	0	0	2	1						
	CAH1131	Introduction to Web Programming - Front end development Lab	0	0	2	1						
			18	5	4	25						
	Total Contact hours (L + T + P)		27									
II	Third Semester						Fourth Semester					
	Course Code	Course Name	L	T	P	C	Course Code	Course Name	L	T	P	C
	CAH2101	Database Management System	3	1	0	4	CAH2201	Data Structure and Algorithm	3	1	0	4
	LLC****	Technical communication - English LSRW	3	0	0	3	CAH2202	Computer Organization and Architecture	3	1	0	4
	CAH2102	Object Oriented Programming	3	0	0	3	CAH2203	Operating System	3	0	0	3

	CAH2130	Database Management System Lab	0	0	2	1	CAH2204	How Human Languages Work	3	0	0	3
	CAH2131	Object-Oriented Programming Lab	0	0	2	1	CAH2230	Operating System Lab	0	0	2	1
	CAH2170	WIP-I* (Work integration project)	0	0	14	7	CAH2270	WIP-II*(Work integration project)	0	0	14	7
			9	1	18	19			12	2	15	22
	Total Contact hours (L + T + P)		28				Total Contact hours (L + T + P)		30			
III	Fifth Semester						Sixth Semester					
	Course Code	Course Name	L	T	P	C	Course Code	Course Name	L	T	P	C
	CAH3101	Computer Network	3	1	0	4	CAH3201	Design and Analysis of Algorithm	3	1	0	4
	CAH3102	Formal language & automata Theory	3	1	0	4	CAH3202	Tools and Techniques of Creative Thinking	3	1	0	4
	XXXXXX	Introduction to Philosophy	3	0	0	3	CAH3203	Compiler Design	3	1	0	4
	XXXXXX	Environmental Science	2	0	0	2	XXXXX	Fundamentals of Business Management	3	1	0	4
	CAH3170	WIP-III*(Work integration project)	0	0	14	7	CAH3270	WIP-IV*(Work integration project)	0	0	14	7
			11	2	14	20			12	4	14	23
	Total Contact Hours (L + T + P)		27				Total Contact Hour (L + T + P)		30			
IV	Seventh Semester						Eight Semester					
	Course Name	Course Name	L	T	P	C	Course Name	Course Name	L	T	P	C
	CAH41XX	Program Elective-I	3	0	0	3	CAH4270	Capstone project/ Internship	0	0	28	14
	CAH41XX	Program Elective-II	3	0	0	3			0	0	28	14
	CAH41XX	Foundation Elective -I	4	0	0	3	Total Contact hour		28			
	CAH41XX	Skilling Elective	3	0	0	3						
	CAH4170	WIP-V*(Work integration project)	0	0	14	7						
			8	0	14	19						
	Total Contact Hours		26									

Academic Elective 1:

Course Code	Course Name
CAH41**	Cloud computing
CAH41**	Distributed systems
CAH41**	Data Mining and Warehousing

Academic Elective II:

Course Code	Course Name
CAH41**	Cryptography
CAH41**	Internet of Things
CAH41**	System Design

Foundation Elective 1:

Course Code	Course Name
CAH41**	Human Mind and Behaviour
CAH41**	Organization Behaviour
CAH41**	Foreign language
CAH41**	Design Thinking 101

Skilling Elective 1:

Course Code	Course Name
CAH41**	Unix Shell Programming
CAH41**	AWS and AWS Security
CAH41**	Data Modelling and Visualization

Index

Program Outcomes

At the end of the program, students will be able to display the following traits/ skills.

1. **Software development:** Develop high-quality software products using industry-standard development tools, techniques, and processes.
2. **Problem-solving:** Analyze complex problems and design creative and effective software solutions using sound computer science principles and best practices.
3. **Collaboration:** Collaborate effectively in diverse, cross-functional teams to develop and deploy software products that meet user needs and business requirements.
4. **Continuous self-learning and improvement:** Continuously self-learn and adapt to new technologies and practices, staying up-to-date with the latest trends and emerging opportunities in the field.
5. **Effective communication:** Communicate effectively and professionally, both orally and in writing, with technical and non-technical stakeholders, and contribute to positive organizational culture and team dynamics.
6. **Critical thinking:** Apply critical thinking skills to evaluate and assess the ethical, legal, social, and economic impacts of software products on individuals, communities, and society as a whole.
7. **Leadership:** Demonstrate leadership and entrepreneurship skills, such as initiative, resourcefulness, risk-taking, and innovation, in creating and launching new software products or businesses.

Program Specific Outcomes

At the end of this program, the students will be able:

1. To apply the principles of software engineering and design to develop, test, and maintain high-quality software products.
2. To effectively communicate and collaborate with team members, stakeholders, and clients to develop and deliver software products that meet their needs and requirements.
3. To apply critical thinking, problem-solving skills, and ethical considerations to address complex issues related to software development, deployment, and maintenance.

Semester 1

Fundamentals of Discrete Mathematics

Introduction

Discrete Mathematics is a foundational course for computer science students that introduces the mathematical concepts and techniques essential to computer science. The course covers topics such as logic, sets, functions, relations, combinatorics, graph theory, and number theory. It emphasizes problem-solving, critical thinking, and effective communication of mathematical ideas. Discrete Mathematics provides the mathematical foundation for further study in algorithms, data structures, and other areas of computer science.

Course outcomes

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

1. To explain the fundamental concepts and principles of discrete mathematics, including logic, sets, functions, and relations.
2. To apply discrete mathematics concepts to solve problems in computer science.
3. To analyze and evaluate the correctness and efficiency of algorithms, and the validity and soundness of logical arguments.
4. To synthesize discrete mathematics concepts to create mathematical models for real-world problems, such as scheduling and network optimization.
5. To evaluate the strengths and limitations of various discrete mathematics techniques, and make informed decisions about which approach to use in a given context.
6. To communicate mathematical ideas and solutions clearly and effectively, both orally and in writing.

Syllabus

Unit 1 Proofs

Introduction and Proofs, Induction, Strong Induction, Number Theory

Unit 2 Structures

Graph theory and Colouring, Matching Problems, Minimum Spanning Tree, Communication Networks, Directed graphs, Relations and partial orders, State machines

Unit 3 Counting - Part 1

Sums, asymptotics, Divide and Conquer Recurrences, Linear Recurrences

Unit 4 Counting - Part 2

Counting Rules, Generating functions, Infinite sets

Unit 5 Probability Part 1
Introduction to Probability, Conditional Probability, Independence, Random variables

Unit 6 Probability Part 2
Expectations, Deviations, Random Walks

Text book(s)

Mathematics for Computer Science; Eric Lehman, F Thomson Leighton, Albert R Meyer; 12th Media Services (5 June 2017)

Reference book(s)

1. Discrete Mathematics and Its Application; Kenneth H Rosen & Dr Kamala Krithivasan; McGraw Hill; 8th edition
2. A Textbook on Discrete Mathematics; CV Sastry and Rakesh Nayak; Wiley (1 October 2020)

C, C++ and Python - Problem solving using Programming

Introduction

Problem Solving using Programming is an introductory course that teaches fundamental programming concepts and techniques using C/C++ and Python. The course emphasizes problem-solving skills and computational thinking, and equips students with the skills necessary to tackle real-world problems using programming.

Course outcomes

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

1. To explain fundamental programming concepts, including data types, control structures, and functions.
2. To apply programming constructs to solve simple problems and algorithms in C/C++ and Python.
3. To analyze and evaluate the efficiency and correctness of algorithms and programs.
4. To synthesize programming constructs to develop larger programs that solve complex problems.
5. To evaluate the strengths and weaknesses of different programming constructs and choose appropriate solutions for different problems.
6. To communicate programming solutions clearly and effectively, both orally and in writing.

Syllabus

Unit 1 Algorithms

Computational thinking, Decomposition, Abstraction, Pattern recognition, Algorithms, Writing pseudocode and translating it to code, Looping (While and do-while loops), Variables (Scope, lifetime and initialization), Datatypes (Structures, classes, enums)

Unit 2 Variables

Review of variables and their uses, including how to declare and initialize variables; Arithmetic operators, Relational operators, Logical operators, Bitwise operators & Assignment operators; Introduction to strings and advanced string manipulation techniques, such as concatenation, substring extraction, searching, and replacing; Introduction to characters including ASCII and Unicode encoding, character classification functions, and character mapping; Typecasting; Using constants to represent fixed values in code; local and global variables, and their uses and limitations; operator precedence, order of evaluation, and short-circuiting

Unit 3 Control basics

If, If else, for loop & while loop; using boolean expressions to control program flow and evaluate conditions; using switch statements to select one of many possible code paths based on a value or condition; advanced topics such as nested loops, loops with multiple variables, and using loops to iterate

Unit 4 Control advanced

Nested if, Nested if elseif, Else, for loop, while loop in arrays and strings; Exception handling; using recursive functions to solve problems that can be broken down into smaller sub-problems; advanced techniques for using loops, such as using loop counters, loop flags, and sentinel values; using advanced branching techniques such as the ternary operator and conditional expressions

Unit 5 Modularity

Organizing code into modules, classes, and functions to improve code structure and reusability; Function parameters; Function return value; reviewing recursion and its use in function design and implementation; Libraries and APIs

Unit 6 Program development

Game development using a programming language, debugging, basic game design principles, user interface design, performance optimization

Text book(s)

Think Like a Programmer: An Introduction to Creative Problem Solving by V. Anton Spraul, Released August 2012, published by No Starch Press

Reference book(s)

1. The C Programming Language; Brian W. Kernighan & Dennis Ritchie; Pearson Education India; 2nd edition
2. Programming in Python 3: A Complete Introduction to the Python Language; Mark Summerfield; Pearson Education; Second edition
3. C++ Programming Language; Bjarne Stroustrup; Pearson Education; 4th edition

Critical Thinking

Introduction

Critical Thinking is a course designed to introduce students to the concepts of reasoning and decision-making, and to the cognitive biases and heuristics that can impede accurate and rational thinking. Based on the seminal book "Thinking, Fast and Slow" by Daniel Kahneman, the course will equip students with the skills to recognize and avoid common thinking errors, and to think more critically and effectively.

Course outcomes

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

1. To describe the cognitive biases and heuristics that can affect human reasoning, and explain how they can lead to thinking errors.
2. To recognize and identify common thinking errors and fallacies in everyday situations.
3. To apply critical thinking skills to analyze and evaluate arguments and evidence.
4. To synthesize ideas and perspectives from different sources to develop reasoned and well-supported arguments.
5. To evaluate the reliability and validity of different sources of information and evidence.
6. To communicate critical thinking ideas and solutions clearly and effectively, both orally and in writing.

Syllabus

Unit 1 – The two systems of thinking

Why think critically, The two systems of thinking, The mental power unit, The lazy system, The marvels of priming, Cognitive ease, Norms, surprises and ease, How judgements work, Jumping to conclusions

Unit 2 – Heuristics and biases

The law of small numbers, Anchoring effect, Availability bias, Representativeness bias, Conjunction fallacy, Survivorship bias, Sunk cost fallacy, Confirmation bias, Google effect and other common biases

Unit 3 – Critical thinking in action

Assignments on identifying biases in the news, creating fake news, writing an unbiased review, alien travel guide, facts vs opinion, worst case scenarios, hypothetical scenarios

Text book(s)

Thinking, Fast and Slow; Daniel Kahneman; Penguin 2012 edition

Reference book(s)

Critical Thinking; Jonathan Haber; The MIT Press; Illustrated edition (7 April 2020)

Fundamentals of Computers & Digital Systems - Breadth of computer science

Introduction

This is a course designed to introduce students to the fundamental concepts and techniques of computer science, including hardware and software systems, algorithms, data structures, programming languages, and software engineering. The course is based on the "Nand2Tetris" project, which guides students through the construction of a complete, working computer system from the ground up, using only basic logic gates.

Course outcomes

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

1. To explain the basic principles and components of computer systems, including logic gates, Boolean algebra, and machine language.
2. To design and implement simple algorithms and data structures, and understand their efficiency and limitations.
3. To develop programming skills in multiple languages, and understand their relative strengths and weaknesses.
4. To analyze and evaluate software engineering practices and principles, and apply them to the design and implementation of larger projects.
5. To synthesize different concepts and techniques in computer science to build a complete, working computer system from scratch.
6. To evaluate the impact of computer science on society and ethical issues related to computer systems and applications.

Syllabus

Unit 1 – Boolean functions, Gate Logic, Boolean arithmetic and the ALU
Boolean logic and functions synthesis, Logic gates and HDL, Hardware simulation and multi-bit buses, Binary numbers and binary addition, Negative numbers and ALU, Boolean arithmetic and ALU

Unit 2 – Memory and Machine language
Sequential logic and flip flops, Memory units and counters, Building registers/ RAM/ Program counter, Overview and elements, The Hack Computer and Machine language, Input/ Output, Hack Programming, Writing assembly level code

Unit 3 – Computer architecture and Assembly languages
Vonn Neumann Architecture and the Fetch-Execute Cycle, CPU, The Hack computer, Memory implementation, CPU implementation, Introduction to Assembly language and assemblers, Hack assembly language, Handling instructions, Handling symbols, Developing a Hack assembler

Unit 4 Virtual machine – Stack arithmetic & Program control
Recap on program compilation, Stack, Memory segments, VM emulator, VM implementation on Hack, VM translator, Program control, Branching, Abstraction, Function Call and Return: Implementation Preview, Function Call and Return: Run-time simulation, Function Call and Return Implementation, VM Implementation on the Hack Platform, VM Translator : Proposed Implementation, Building the VM translator

Unit 5 High level language
The Jack Language in a nutshell, Object-Based Programming, List Processing, Jack Language Specification: Syntax and Data Types, Jack Language Specification: Classes, Jack Language Specification: Methods, Developing Apps using the Jack language and OS,A Sample Jack App: Square Dance, Graphics Optimization, Syntax Analysis & Lexical Analysis, Grammars

Unit 6 Compiler – Syntax Analysis
Parse Trees & Parse Logic, The Jack Grammar, The Jack Analyzer, The Jack Analyzer: Proposed Implementation, Building a syntax Analyzer, Code Generation, Handling Variables, Handling Expressions

Unit 7 Operating systems
Operating System, Efficiency Matters, Mathematical Operations, Memory Access, Heap Management, Graphics, Line Drawing, Handling Textual Output

Text book(s)

The Elements of Computing Systems: Building a Modern Computer from First Principles; Noam Nisan & Shimon Shocken; The MIT Press; 2nd edition

Introduction to Web Programming - Front end development

Introduction

In this course, students will learn the fundamentals of front-end web development, including HTML, CSS, JavaScript, and React JS. They will learn how to create responsive and dynamic web pages, as well as develop their problem-solving and critical thinking skills. The course will focus on hands-on projects and exercises to give students practical experience in front-end web development.

Course outcomes

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

1. To develop proficiency in HTML, CSS, and JavaScript, and apply them to the development of interactive and responsive web pages.
2. To design and implement web pages that are accessible, user-friendly, and optimized for search engines.
3. To create and use reusable code components to improve productivity and maintainability.
4. To demonstrate an understanding of the principles of web design and user experience, and apply them to front-end development.
5. To use debugging tools and techniques to identify and fix errors in web applications.
6. To work collaboratively and effectively in a team environment on web development projects.

Syllabus

Unit 1 HTML, CSS & JS first steps

Environment set up, Introduction to HTML, HTML Block Elements, HTML Inline Elements, HTML Forms, Introduction to CSS, CSS Font & Text, CSS Selectors, CSS Inheritance, CSS Colors, Box Model, Flex Box, JS DOM, Introduction to JS, JS Variables, JS Data Types, Basics of JS Operators, Basics of JS Strings, Basics of JS Conditional Statements, Basics of JS Control Statements, Basics of JS Arrays, Basics of JS Functions, Basics of JS Objects

Unit 2 – HTML, CSS & JS Deep dive Part 1

CSS Advanced Selectors, CSS Positioning, Advanced Flexbox, CSS Grids, Responsive Design, JS Operators, JS Strings, JS Conditional Statements, JS Control Statements, JS Arrays & Functions, JS Objects

Unit 3 HTML, CSS & JS Deep dive Part 2

JS Advanced Functions, JS Nested Data Structures, JS Higher Order Functions, JS Event Handling, Object Oriented JS, JS Closure, JS Storage, CSS Transition, CSS Animation

Unit 4 JS the hard parts

JS Prototypal Inheritance, JS Async, JS Callbacks, JS Promises, JS APIs, JS Axios, Unit testing in JS, Deployment

Unit 5 React first steps
Environment set up, Introduction to React, Props & State, Components, React App using Babel, Rendering lists of data, JSX, Hooks, Additional Hooks, Event Handling, Component Lifecycle, Class based components, Routing

Unit 6 React deep dive
React Forms, Fetching Data from API, Redux, React Redux, Redux Toolkit, React CSS Library, Material-ui

Text book(s)

1. Web development: This book includes: Web development for Beginners in HTML + Web design with CSS + Javascript basics for Beginners; Andy Vickler; Ladoo Publishing LLC (24 May 2021)
2. The Road to Learn React: Your Journey to Master Plain Yet Pragmatic React.Js; Robin Wieruch; Zaccheus Entertainment (1 January 2018)

Reference book(s)

1. HTML, CSS, and JavaScript All in One; Julie C. Meloni & Jennifer Kyrnin; Pearson Education; Third edition
2. React and React Native: A complete hands-on guide to modern web and mobile development with React.js; Adam Boduch & Roy Derks; Packt Publishing Limited; 3rd edition

Design for Developers

Introduction

Design for Developers is a course designed to introduce students to the fundamental principles of user interface (UI) and user experience (UX) design, with a focus on the needs of developers. The course covers topics such as visual design, user-centered design, usability testing, and accessibility, and provides hands-on experience with design tools and techniques.

Course outcomes

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

1. To explain the basic principles of user-centered design, and how they apply to software development.
2. To evaluate and critique the design of existing software applications, and identify areas for improvement in terms of user experience and usability.

3. To apply visual design principles and techniques, such as typography, color theory, and layout, to create effective user interfaces.
4. To conduct usability testing and other evaluation methods to measure the effectiveness and usability of software applications.
5. To synthesize different design concepts and techniques to create well-designed and user-friendly software interfaces.
6. To evaluate the accessibility of software applications and understand the importance of designing for users with diverse needs.

Syllabus

Unit 1 How to approach design as a developer

Why design for developers, Getting started with design education, Getting started with Figma, Labs on Figma

Unit 2 UX Design for Developers

Understanding design sprints, Understanding the user, Information architecture, Accessibility in design, How to build good digital products, Psychology in UX design, User research, User journey mapping, Wireframing, Prototyping

Unit 3 UI Design for Developers

Visual design principles, Visual hierarchy, Working with text, Layouts and spacing, Working with colours, Working with images, Creating depth in design, Working with components

Text book(s)

About Face; Alan Cooper, Robert Reimann, Christopher Noessel and David Cronin; Wiley Publishing, 2014

Reference book(s)

1. Hands-on UX design for developers; Elvis Canziba; Packt, 2018
2. Refactoring; Adam Wathan and Steve Shoger

Semester 2

Full Stack Web Development

Introduction

Full Stack Web Development is a course designed to teach students how to build dynamic web applications using the MERN stack (MongoDB, Express, React, and Node.js). The course covers both backend and database development, as well as front-end technologies like HTML, CSS, and JavaScript. By the end of the course, students will have built their own unique full stack application, which they can showcase in their portfolio.

Course outcomes

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

1. To describe the architecture and components of a full stack web application, including front-end and back-end technologies and their interactions.
2. To design and develop a database schema using MongoDB, including defining data models, creating indexes, and writing queries.
3. To create RESTful APIs using Node.js and Express, including handling HTTP requests and responses, and interacting with the database.
4. To implement front-end user interfaces using React, including using React components, managing state, and handling user input.
5. To integrate front-end and back-end components to create a fully functional full stack web application, including using asynchronous communication and handling errors and exceptions.
6. To design and implement a unique full stack application as a capstone project, including identifying user requirements, developing a software design, and implementing and testing the application.

Syllabus

Unit 1: Introduction to Full Stack Web Development
Overview of full stack web development, the MERN stack and its components, setting up the development environment (using tools like Node.js, MongoDB, and VSCode), basic backend development concepts (e.g., routing, handling requests, working with databases)

Unit 2: Backend Development and Databases
Designing and implementing a database schema using MongoDB, writing basic queries and data manipulation commands, creating a RESTful API using Node.js and Express, handling HTTP requests and responses, interacting with the database using Mongoose

Unit 3: Front-end Development with React

Overview of React and its components, creating and managing React components, working with state and props, handling user input and events, styling with CSS and Bootstrap

Unit 4: Software Engineering and Project Management
Introduction to Software Engineering and SDLC, Agile and Scrum methodologies, software project management tools (like JIRA/ Trello/ GitHub), version control with Git

Unit 5: Integrating Front-end and Back-end Components
Asynchronous communication between front-end and back-end, handling errors and exceptions, using middleware to process requests, authentication and authorization with JWT

Unit 6: Capstone Project
Design and development of a unique full stack application, identifying user requirements and developing a software design, implementing and testing the application, deployment on cloud platforms like AWS, Heroku or Netlify

Text book(s)

Full-Stack React Projects: Learn MERN stack development by building modern web apps using MongoDB, Express, React, and Node.js; Shama Hoque; Packt Publishing Limited; 2nd edition

Reference book(s)

1. Beginning MERN Stack: Build and Deploy a Full Stack MongoDB, Express, React, Node.js App; Greg Lim
2. Pro MERN Stack: Full Stack Web App Development with Mongo, Express, React, and Node; Vasan Subramanian; Apress; 2nd edition

Professional skills for the workplace

Introduction

Professional Skills for the Workplace is a course designed to help students develop the human skills they need to succeed in the modern workplace. The course is based on the Human Skills Matrix, developed by MIT JWEL, and covers four key quadrants: thinking, leading, interacting, and managing oneself. Students will learn practical strategies for improving their communication, collaboration, problem-solving, and self-management skills.

Course outcomes

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

1. To explain the four quadrants of the Human Skills Matrix and how they relate to success in the workplace.
2. To apply critical thinking skills to solve complex problems and make effective decisions.

3. To lead and collaborate effectively with others, including managing teams and facilitating group discussions.
4. To communicate clearly and persuasively in various professional contexts, including written, verbal, and nonverbal communication.
5. To manage time and prioritize tasks effectively, including setting goals and developing strategies for self-motivation and self-improvement.
6. To synthesize different human skills and apply them to real-world workplace challenges, such as conflict resolution, innovation, and project management.

Syllabus

Unit 1 – Managing ourselves
Self-awareness, Adaptability, Managing emotions, Accountability, Professionalism, Taking initiative, Persistence, Planning and organizing, Integrity

Unit 2 – Interacting
Writing good emails & reports, Use of communication tools, Use of project management tools, Curating relationships on social media through professional networking sites, Negotiation skills, Presentation skills

Univ 3 – Thinking
Ethical dilemmas, being an intrapreneur, Building a growth mindset, Systems thinking

Unit 4 – Leading
Empowering others, Having a strategic vision, Project management, Performance management

Text book(s)

COMMUNICATION SKILLS FOR PROFESSIONALS AND STUDENTS: An Occupational Therapist's Perspective; Dr. Amitabh Kishor Dwivedi; Notion Press; 1st edition

Reference book(s)

The Communication Book: 44 Ideas for Better Conversations Every Day; Mikael Krogerus & Roman Tschäppeler; Portfolio Penguin (19 April 2018)

Learning how to learn

Introduction

Learning How to Learn is a course designed to help students master the concepts and techniques of effective learning, with a focus on online and remote learning environments. The course covers topics such as the science of learning, memory techniques, metacognition, and overcoming procrastination, and provides practical strategies for improving learning outcomes in any subject area.

Course outcomes

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

1. To explain the key principles and processes involved in effective learning, and how they can be applied to any subject area.
2. To apply a variety of memory techniques, such as spaced repetition and visualization, to improve retention and recall of information.
3. To practice metacognitive strategies, such as self-reflection and self-assessment, to monitor and improve learning progress.
4. To identify and overcome common obstacles to effective learning, such as procrastination and distractions.
5. To synthesize different learning techniques and apply them to real-world learning challenges, such as preparing for exams or learning new skills.
6. To evaluate the effectiveness of different learning strategies and make evidence-based decisions about the best approach for a given learning situation.

Syllabus

Unit I: How learning works

Introduction to Focus and Diffuse Modes, the role of Practice in learning,Introduction to Memory & Sleep in learning, Procrastination on learning, case studies to end the unit (on learning languages, creativity, problem solving and much more – how learning happens for these skills)

Unit II: Chunking

Introduction to Chunking, how to form a Chunk, Illusions of Competence, the pitfalls of overlearning & choking

Unit III: Procrastination, Memory and Learning

Avoiding rut thinking, strategies and techniques to tackle procrastination while learning, Process vs. Product, Deep-dive into memory, Long-term memory, Being a life-long learner, The Memory Palace technique

Unit IV: Being a better learner

Creating Visual metaphors, creating analogies for learning, Importance of checklists, Learning vs. Prepping for tests, Case studies

Text book(s)

1. Learning How to Learn: How to Succeed in School Without Spending All Your Time Studying; A Guide for Kids and Teens; Barbara Oakley and Terrence Sejnowski; Tarcher Perigree 2018
2. Make it Stick: The Science of Successful Learning; Peter C Brown, Henry L. Roediger III & Mark A. McDaniel; Harvard University Press; 1st edition

Reference book(s)

1. Mindshift: Break Through Obstacles to Learning and Discover Your Hidden Potential; Barbara Oakley, Penguin, 2017
2. How we learn: The surprising truth about when, where and why it happens; Benedict Carey, Random House, 2014
3. Ultralearning: Accelerate Your Career, Master Hard Skills and Outsmart the Competition; Scott H Young, Harper Collins, 2019

Semester 3

Database Management System

Introduction

This course is designed to teach students the fundamental concepts and principles of Database Management Systems (DBMS) and how to effectively design, implement, and manage databases. Students will learn various database models and acquire hands-on experience in using popular DBMS tools.

Course outcomes

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

1. To understand the fundamental concepts of database management systems including data models, data normalization, and database design.
2. To be able to design and implement a relational database using SQL.
3. To be able to use a popular DBMS tool such as MySQL to create and manage databases.
4. To be able to use SQL to query and manipulate data stored in a database.
5. To be able to apply database management concepts to real-world scenarios and problem-solving.
6. To be able to design and implement a functional database-driven web application.

Syllabus

Unit I – Introduction to DBMS

The Evolution of Database Systems- Overview of a Database Management System-Outline of DatabaseSystem Studies-The Entity-Relationship Data Model: Elements of the E/R Model-Design Principles-The Modeling of Constraints-Weak Entity Sets.

Unit II – The Relational data model and Algebra

Basics of the Relational Model-From E/R Diagrams to Relational Designs-Converting Subclass Structures to Relations-Functional Dependencies-Rules About Functional Dependencies-Design of Relational Database Schemas – Multi valued Dependencies- Relational Algebra: Relational operations-Extended Operators of Relational Algebra- Constraints on Relations.

Unit III – SQL

Simple Queries in SQL-Sub queries-Full-Relation Operations-Database Modifications-Defining a Relation Schema-View Definitions- Constraints and Triggers: Keys and Foreign Keys-Constraints on Attributes and Tuples-Modification of Constraints-Schema-Level Constraints and Triggers - Java Database ConnectivitySecurity and User Authorization in SQL

Unit IV – Index structures and query processing

Index Structures:Indexes on Sequential Files-Secondary Indexes-B-Trees-Hash Tables-Bitmap Indexes-Query Execution: Physical-Query-Plan Operators-One-Pass , two-pass & index based Algorithms, Buffer Management, Parallel Algorithms-Estimating the Cost of Operations-Cost-Based Plan Selection -Order for Joins-Physical-Query-Plan

Unit V – Failure recovery and concurrency control

Issues and Models for Resilient Operation -Undo/Redo Logging-Protecting against Media FailuresConcurrency Control: Serial and Serializable Schedules-Conflict-Serializability-Enforcing Serializability by Locks-Locking Systems With Several Lock Modes-Concurrency Control by Timestamps, validation transaction management: Serializability and Recoverability-View Serializability-Resolving Deadlocks Distributed Databases: commit& lock.

Text book(s)

Database Systems: Models, Languages, Design And Application Programming By Ramez Elmasri, Shamkant B. Navathe, Pearson 6th edition

Technical Communication - English LSRW

Introduction

This is a course aimed at developing the four pillars of Technical English communication: Listening, Speaking, Reading and Writing. The course is designed to benchmark against the CEFR framework, and is tailored to focus on business and technical communication.

Course outcomes

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

1. To comprehend and interpret spoken and written English at the CEFR B2 level
2. To produce spoken and written English at the CEFR B2 level
3. To develop active listening and speaking skills in order to participate in discussions, debates, and presentations
4. To improve reading speed, comprehension, and critical analysis of texts related to business and technical domains
5. To hone writing skills and produce effective business and technical documents such as emails, reports, proposals, and presentations
6. To apply effective communication strategies in a professional setting, including cultural awareness and intercultural communication.

Syllabus

Unit 1: Listening and Speaking

Introduction to listening and speaking skills, , Understanding different accents and intonation, Developing active listening skills, Participating in discussions and debates, Giving presentations and speeches

Unit 2: Reading

Introduction to reading skills, Skimming and scanning for information, Identifying main ideas and supporting details, Understanding tone and purpose, Reading for inference and implication

Unit 3: Writing

Introduction to writing skills, Planning and organizing written work, Writing effective emails and memos

Writing reports and proposals, Writing for specific audiences and purposes

Unit 4: Grammar and Vocabulary

Introduction to grammar and vocabulary, Understanding verb tenses and structures, Practicing correct sentence formation, Building vocabulary through context and word roots, Using idioms and phrasal verbs in communication

Unit 5: Business and Technical Communication

Introduction to business and technical communication, Writing effective resumes and cover letters Conducting effective interviews, Understanding and writing technical documents, Communicating with clients and colleagues in professional settings,

Unit 6: Test Preparation and Practice

Introduction to the CEFR framework, Practice tests and quizzes to assess learning, Review and feedback on written and spoken communication, Goal setting for further improvement, Final project or presentation

Text book(s)

Professional English: for AKTU, Meenakshir Raman and Sangeetha Sharma, Oxford Publication 1st edition

Reference book(s)

Word Power Made Easy; Norman Lewis; Penguin Random House India; Latest edition 2015

Object Oriented Programming

Introduction

This course teaches the principles of Object-Oriented Programming (OOP), which is a key concept in modern programming paradigms. The course is language-agnostic, but students can choose to implement their projects in either C++ or Python.

Course outcomes

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

1. To explain the fundamental concepts and principles of OOP, such as encapsulation, inheritance, and polymorphism.
2. To apply OOP concepts to develop software applications in C++ and Python.
3. To design and implement complex data structures and algorithms using OOP concepts.
4. To develop and manage large-scale software projects using OOP design patterns and software engineering practices.
5. To analyze and evaluate the performance and efficiency of OOP-based programs.
6. To collaborate and communicate effectively in a team environment to develop and maintain OOP-based software projects.

Syllabus

Unit 1 Introduction to Object-Oriented Programming

Basics of OOP, Objects and Classes, Abstraction, Encapsulation, Inheritance, Polymorphism, Procedural Programming vs OOP

Unit 2 Advanced Concepts in OOP

Templates, Overloading, Exception Handling, Operator Overloading, Inheritance and Polymorphism, Advanced Topics in Inheritance

Unit 3 Object-Oriented Design Principles

SOLID Principles, Design Patterns, Design for Reuse, Design for Testability.

Unit 4 Software Engineering Practices

Agile Development, Waterfall Model, Software Development Life Cycle (SDLC), Version Control, Code Reviews, Testing and Debugging.

Unit 5

Advanced Topics in OOP Multithreading, Concurrency, Networking, GUI Programming, Database Programming, Best Practices for OOP

Text book(s)

1. Object-Oriented Programming with C++; E Balagurusamy; McGraw Hill; Eighth edition
2. Python Object-Oriented Programming; Steven F. Lott & Dusty Phillips; Packt Publishing Limited; 4th edition
3. Java The Complete Reference; Herbert Schildt; McGraw Hill; Eleventh edition

Reference book(s)

1. Object Oriented Programming C++; Robert Lafore; Pearson Education India; 4th edition
2. OOPS with C++ and Java; Balagurusamy; McGraw Hill Education 2014 edition
3. Python 3 Object-Oriented Programming; Dusty Phillips; Packt 3rd edition

Semester 4

Data Structure and Algorithms

Introduction

This course is focused on the study of data structures and algorithms, which are fundamental to computer science. It is designed to provide a deep understanding of the theory behind data structures and algorithms, as well as practical implementation techniques. The course is practice-heavy, and students will develop skills in algorithm design and analysis, problem-solving, and programming.

Course outcomes

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

1. To explain the principles and concepts underlying data structures and algorithms, such as complexity analysis, recursion, and graph theory.
2. To design and implement efficient data structures and algorithms, using a variety of techniques and approaches.
3. To analyze the performance and correctness of data structures and algorithms, using mathematical and empirical methods.
4. To apply data structures and algorithms to solve real-world problems, across a range of application areas such as search, sorting, and graph algorithms.
5. To evaluate different data structures and algorithms, based on their suitability for specific problem domains and constraints.
6. To create and present well-structured and well-documented code that implements data structures and algorithms.

Syllabus

Unit I – Introduction
Algorithmic thinking, peak finding, Models of computation, Document distance, Python cost model
Unit II – Sorting and trees (14 hours)
Insertion sort, Merge sort, Heaps and heap sort, Binary search trees, BST sort, AVL trees and sort, Counting sort, radix sort, lower bounds for sorting and searching

Unit III – Hashing
Hashing with chaining, Table doubling, Karp-Rabin, Open addressing, Cryptographic hashing

Unit IV – Numerics
Integer arithmetic, Karatsuba multiplication, Square roots, Newton's method

Unit V – Graphs
Breadth first search, Depth first search, Topological mapping

Unit VI – Shortest paths
Single-source shortest paths problem, Dijkstra, Bellman-Ford, Speeding up Dijkstra

Unit VII – Dynamic Programming
Memoization, subproblems, guessing, bottom-up; Fibonacci, shortest paths, Parent pointers; text justification, perfect-information blackjack, String subproblems, psuedopolynomial time; parenthesization, edit distance, knapsack, Two kinds of guessing; piano/guitar fingering, Tetris training, Super Mario Bros

Unit VIII – Advanced topics
Computational complexity, Algorithms research topics

Text book(s)

Data Structures and Algorithms Made Easy; Narasimha Karumanchi; CareerMonk Publications; 5th edition

Reference book(s)

Introduction to Algorithms; Thomas H. Cormen, Charles E. Leiserson, Ronald L. Rivest, Clifford Stein; PHI Learning Pvt. Ltd. (Originally MIT Press); Third edition

Computer Organization and Architecture

Introduction

This course provides an introduction to computer organization and architecture, which deals with the physical components of a computer system and how they work together to execute instructions. Topics covered include CPU design, memory hierarchy, I/O systems, and assembly language programming.

Course outcomes

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

1. To explain the basic components of computer systems and their functions at a low level, including CPU, memory, and I/O systems.
2. To analyze the performance of computer systems based on metrics such as clock rate and CPI.
3. To design and implement simple CPU and memory systems using hardware description languages such as Verilog.
4. To write and debug assembly language programs that interact with system hardware.

5. To evaluate the trade-offs involved in different design choices for computer systems, such as the size of the instruction set or the level of parallelism.
6. To apply knowledge of computer organization and architecture to optimize code for performance and minimize energy consumption.

Syllabus

Unit I

Basic functional blocks of a computer: CPU, memory, input-output subsystems, control unit. Instruction set architecture of a CPU - registers, instruction execution cycle, RTL interpretation of instructions, addressing modes, instruction set. Case study - instruction sets of some common CPUs.

Unit II

Data representation: signed number representation, fixed and floating point representations, character representation. Computer arithmetic - integer addition and subtraction, ripple carry adder, carry look-ahead adder, etc. multiplication - shift-and-add, Booth multiplier, carry save multiplier, etc. Division - non-restoring and restoring techniques, floating point arithmetic.

Unit III

CPU control unit design: hardwired and micro-programmed design approaches, Case study - design of a simple hypothetical CPU. Memory system design: semiconductor memory technologies, memory organization.

Unit IV

Peripheral devices and their characteristics: Input-output subsystems, I/O transfers - program controlled, interrupt driven and DMA, privileged and non-privileged instructions, software interrupts and exceptions. Programs and processes - role of interrupts in process state transitions. Performance enhancement techniques

Unit V

Pipelining: Basic concepts of pipelining, throughput and speedup, pipeline hazards.
Memory organization: Memory interleaving, concept of hierarchical memory organization, cache memory, cache size vs block size, mapping functions, replacement algorithms, write policy.

Text book(s)

Computer Organization and Design; Patterson; Elsevier; 6th edition

Reference book(s)

1. Computer Architecture, Berhooz Parhami; Oxford University Press (19 April 2012)
2. Computer System Architecture; Mano M Morris; Pearson 3rd edition

Operating Systems

Introduction

This course introduces the concepts and principles of operating systems, including process management, memory management, file systems, and device management. Students will learn about various scheduling algorithms and memory allocation techniques used in operating systems. They will also be introduced to different types of operating systems and will learn about the trade-offs involved in designing and implementing these systems. Through practical assignments and projects, students will gain hands-on experience in implementing basic operating system functionalities. By the end of the course, students will have a solid understanding of the security and privacy issues in modern operating systems.

Course outcomes

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

1. To understand the fundamental concepts of operating systems such as processes, threads, synchronization, and memory management.
2. To analyze the performance of various scheduling and memory allocation algorithms.
3. To compare and contrast different types of operating systems, such as batch, multi-programmed, and real-time systems.
4. To develop an understanding of device management, file systems, and virtualization.
5. To gain practical experience in implementing basic operating system functionalities.
6. To understand the security and privacy issues in modern operating systems.

Syllabus

Unit I - Computer Arithmetic & Processor Organisation

Computer Registers, Classification of Instruction – Size: three, two, one and zero instruction, Addressing Mode. Arithmetic and Logic Circuit Design. Instruction execution cycle: Sequencing of control signals, hardwired control, micro-programmed control, control signals, microinstructions, micro program sequencing, pre-fetching microinstructions. Introduction to graphical processing unit (GPU).

Unit II - Memory Organization

Memory hierarchy, Main memories chip architectures, memory address map, memory assembly to CPU. Auxiliary memory: magnetic tapes, disks (magnetic and SSDs). Associate memory: hardware organization, match logic, read and writes operations. Cache memory. Memory interleaving technique.

Unit III - Parallel Processing

Parallel processing, examples of parallel processing machines. Classification of parallel processing: Handler classification – pipeline processing, vector processing and array processing, Flynn's classification – SISD, SIMD, MISD, MIMD. Pipeline conflicts.

Unit IV - Introduction to Operating Systems and Process Management

Introduction to operating systems. Process Management: what is a Process?, Process state, Process control block. Threads. Cooperating processes. Inter-process communication. CPU scheduling algorithms: Fist come first serve, shortest job first – primitive & non primitive, Round Robin. Deadlock: Necessary conditions for occurrence of deadlocks, Deadlock detection – Resource Allocation Graph. Deadlock Avoidance Algorithms: Banker Algorithm and Safety Algorithm.

Unit V - Memory Management

Memory allocation techniques: Continues (Multiprogramming with fixed number of tasks), Non-continues (Multiprogramming with variable number of tasks), Paging, Demand paging. Page replacement algorithms: First in first out, Least frequently used, Most frequently used, Optimal page replacement. Virtual memory concepts.

Text book(s)

Operating Systems Concepts, Abraham Silberschatz, Peter B. Galvin and Greg Gagne, Wiley, 2012.

Reference book(s)

1. The Design of the Unix Operating System, Maurice Bach, Pearson; 1st edition
2. Operating systems concepts; Avi Silberschatz, Peter Baer Galvin, Greg Gagne; Wiley; Ninth edition

How Human Languages Work

Introduction

In this course, students will explore the fundamental concepts of human language, including phonetics, syntax, semantics, and pragmatics. The course will provide an overview of the structure and function of human language and explore how language is processed in the brain.

Course outcomes

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

1. Understand the basic concepts of human language, including phonetics, syntax, semantics, and pragmatics.
2. Explore the structure and function of human language, including the sound systems of languages, grammatical structures, and the meanings of words and sentences.

3. Understand the social and cultural dimensions of language use, including language variation and change, multilingualism, and language attitudes.
4. Understand how language is processed in the brain, including the neural mechanisms of language comprehension and production.
5. Analyze the ways in which language reflects and shapes cultural and social identities.
6. Develop critical thinking skills by analyzing linguistic data and applying linguistic concepts to real-world problems.

Syllabus

Unit 1: Introduction to Language and Linguistics
Overview of Linguistics and its sub-fields, Phonetics and Phonology, Morphology

Unit 2: Syntax and Semantics
Syntax and Syntactic Structures, Semantics and Semantic Structures, Language Typology

Unit 3: Language Acquisition and Language Change
First and Second Language Acquisition, Language Change and Language Contact, Historical Linguistics

Unit 4: Sociolinguistics and Applied Linguistics
Sociolinguistics and Language Variation, Language Policy and Planning, Language and Identity, Language and Technology

Unit 5: Language in Context
Language and Culture, Language and Gender, Language and Power, Language and Globalization

Unit 6: Language Research and the Future of Linguistics
Research Methods in Linguistics, Current Debates in Linguistics, The Future of Linguistics

Text book(s)

How Languages Work: An Introduction to Language and Linguistics by Carol Genetti; Cambridge University Press; 2nd edition

Reference book(s)

How Language Works; David Crystal; Penguin UK (2007 edition)

Semester 5

Computer Networks

Introduction

This course will cover the fundamental concepts and principles of computer networks, including network architecture, protocols, and services. Students will learn about various network technologies, such as LAN, WAN, and wireless networks, and their applications in different contexts. The course will also address network security and management issues.

Course outcomes

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

1. To identify the different components and layers of computer networks.
2. To explain the functions of different protocols used in computer networks.
3. To analyze the performance and limitations of different network architectures.
4. To design and implement basic network configurations using routers and switches.
5. To troubleshoot common network issues using various network analysis tools.
6. To evaluate the security concerns and design solutions to ensure network security.

Syllabus

Unit 1 Internetworking and Routing

Internet architecture, Unicast IP Forwarding and Routing, Internet Routing in-the-Wild (Measurement), Big Fast Routers, Security Issues in the Internet Architecture, Robustness

Unit 2 – Resource Management

End-to-End Congestion Control, Router-Assisted Congestion Control, Active Queue Management, and Scheduling, Modeling and Measurement, Adaptive Applications and Internet QoS

Unit 3– Network Services

Wireless/Mobile Networking, Naming: DNS, Peer-to-Peer Networking, Distributed Hash Tables, Overlay Routing, Multicast, Network Protection, Reliable Transport and Congestion Control, Unicast Routing, Adaptive and Network-Aware Applications, Traffic Engineering, Flow Modeling, Wireless Protocols, Naming, Web Caching

Text book(s)

1. A S Tanenbaum, Computer Networks, 5th Ed., Pearson, 2010.
2. B.A. Forouzan, TCP/IP Protocol Suite, 4th Ed., TMH, 2010.

Reference book(s)

1. TCP/IP illustrated, Volume 1: The Protocols, W.R. Stevens, 2nd Ed., Addison-Wesley, 2015.
2. Internetworking with TCP/IP Principles, Protocols and Architecture, D E. Comer, 6th Ed., Pearson, 2013.

Formal Language and Automata Theory

Introduction

Formal language and automata theory is a branch of computer science that studies the theoretical foundation of computer science, including the formal languages that computers can recognize and the automata that can recognize those languages. This course is designed to give students an understanding of formal languages, grammars, and automata, and how to use them to solve practical problems.

Course outcomes

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

1. To identify the regular and context-free languages that a given automaton can recognize.
2. To design regular expressions and context-free grammars for given languages.
3. To construct finite automata, pushdown automata, and Turing machines to recognize given languages.
4. To analyze the time and space complexity of algorithms that operate on formal languages.
5. To apply the principles of formal languages and automata to real-world problems, such as pattern matching and parsing.
6. To evaluate and critique different models of computation and their relative strengths and weaknesses.

Syllabus

Unit 1 Automata methods and Finite Automata

Introduction to formal proof, Additional forms of proof, Inductive proofs, The central concepts of Automata theory, Deterministic finite automata, Nondeterministic finite automata, Text search, Finite automata with Epsilon transitions

Unit 2– Regular expressions and languages

Regular expressions, Applications, Algebraic laws for regular expressions, Proving languages not to be regular, Closure properties of regular languages, Decision properties, Equivalence and minimization

Unit 3 Context free Grammar and Languages

Context free grammar, Parse trees, Applications, Ambiguity

Unit 4 Pushdown Automata
The languages of a PDA, Equivalance of PDA and CFG, Deterministic PDA

Unit 5 Intro to Turing machines
Problems that computers cannot solve, The Turing machine, Programming techniques for Turing machine, Extensions to the basic Turing machine

Unit 6 Undecidability and Intractable problems
P and NP, NP-complete problem, A restricted satisfiability problem, Additional NP-complete problems,

Text book(s)

Automata Theory Language & Computation; Hopcraft; Pearson 3rd edition

Reference book(s)

1. Theory of Computer Science: Automata, Languages and Computation; KLP Mishra; Prentice Hall India Learning Private Limited; 3rd edition
2. Switching and Finite Automata Theory; Jha; Cambridge University Press; South Asian edition (8 June 2010)

Introduction to Philosophy

Introduction

In this course, students will be introduced to fundamental concepts in philosophy such as logic, ethics, epistemology, metaphysics, and aesthetics. They will learn about the major philosophical ideas of historical and contemporary thinkers and explore various philosophical arguments.

Course outcomes

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

1. To identify and explain the major philosophical concepts and ideas.
2. To analyze and evaluate philosophical arguments.
3. To develop the ability to think critically and creatively about philosophical problems.
4. To apply philosophical theories and concepts to real-world situations.
5. To engage in philosophical discussions and debates with peers.
6. To appreciate the relevance of philosophy to personal and professional life.

Syllabus

Unit 1: Morality
The status of morality, Objectivism, Relativism, Emotivism

Unit 2: What is knowledge?
The basic constituents of knowledge, the classical account of knowledge, The Gettier problem, Choices

Unit 3: Free will
Determinism, Libetarianism, Compatabilism, Hard determinism, Free will and do we have it?

Unit 4: Obligation to obey the law
The grounds of political obligation, Consent, Fairness, Gratitude and Benefit

Unit 5: Should you believe what you hear?
Reid's challenge to Hume, Reid's Argument, Enlightenment, Intellectual Autonomy

Text book(s)

Philosophy Made Slightly Less Difficult; Garrett J. Deweese & J. P. Moreland; IVP Academic; Second edition

Reference book(s)

1. The Big Questions: A Short Introduction to Philosophy by Robert C. Solomon and Kathleen M. Higgins, published by Wadsworth Publishing (2011).
2. Think: A Compelling Introduction to Philosophy by Simon Blackburn, published by Oxford University Press (2001).
3. Philosophy: The Basics by Nigel Warburton, published by Routledge (2012).
4. The Story of Philosophy by Will Durant, published by Pocket Books (1991).
5. An Introduction to Indian Philosophy; Satishchandra Chatterjee; Rupa & Co 2012 edition

Environmental Science

Introduction

This course provides an introduction to environmental science and how it impacts the world we live in. The course focuses on key environmental issues such as climate change, pollution, biodiversity loss, and sustainable development. The course will also highlight the role of technology and engineering in mitigating environmental issues.

Course outcomes

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

1. To understand the key concepts and principles of environmental science, such as ecosystems, biodiversity, and sustainability.
2. To analyze the impact of human activities on the environment, such as pollution and climate change, and understand the consequences of environmental degradation.
3. To evaluate the effectiveness of different strategies for mitigating environmental problems, such as renewable energy, conservation, and public policy.
4. To apply scientific methods and tools to analyze and solve environmental problems.
5. To assess the role of technology and engineering in environmental sustainability and the development of a green economy.
6. To demonstrate a commitment to environmental stewardship and the promotion of sustainable practices in personal and professional contexts.

Syllabus

Unit 1: Introduction to Environmental Science

Understanding the interdisciplinary nature of environmental science, The history of environmentalism and environmental movements, The role of scientific research in understanding environmental issues, Case studies: analyzing environmental challenges from local to global scales

Unit 2: Ecosystems and Biodiversity

Understanding ecosystem structure and function, Principles of population ecology, Biodiversity: its importance, distribution, and threats, Case studies: conservation efforts, ecosystem management, and restoration

Unit 3: Climate Change and its Impacts

Understanding the causes and impacts of climate change, Climate modeling and prediction, Mitigation and adaptation strategies, Case studies: local and global responses to climate change, renewable energy technologies

Unit 4: Water Resources and Pollution

Understanding the water cycle and its management, Sources and effects of water pollution, Principles of water treatment, Case studies: water resource management, water pollution incidents, remediation of contaminated sites

Unit 5: Land Use and Natural Resource Management

Understanding land use patterns and their impacts, Principles of soil science, Natural resource management: forests, fisheries, wildlife, Case studies: urbanization and land use change, deforestation, sustainable agriculture

Unit 6: Environmental Policy and Management

The role of public policy in addressing environmental issues, Environmental regulations and their implementation, Corporate environmental responsibility, Case studies: environmental policy and decision-making, stakeholder engagement, corporate sustainability

Text book(s)

1. "Living in the Environment" by G. Tyler Miller and Scott E. Spoolman (Cengage Learning, 2019)
2. "Environmental Science: A Global Concern" by William Cunningham and Mary Cunningham (McGraw-Hill Education, 2019)

Reference book(s)

1. "Essential Environment: The Science Behind the Stories" by Jay H. Withgott and Matthew Laposata (Pearson, 2019)
2. "Environmental Science for the AP Course" by Andrew Friedland, Rick Relyea, and David Courard-Hauri (W.H. Freeman, 2019)
3. "The Sixth Extinction: An Unnatural History" by Elizabeth Kolbert (Henry Holt and Co., 2014)

Semester 6

Design and Analysis of Algorithms

Introduction

This course provides an in-depth understanding of fundamental algorithms, algorithm design techniques, and algorithm analysis. Students will gain experience in designing and analyzing algorithms, which will help them solve computational problems more efficiently.

Course outcomes

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

1. To apply algorithmic problem-solving techniques using a variety of algorithm design methods.
2. To analyze the time and space complexity of algorithms and compare the efficiency of different algorithms for the same problem.
3. To select appropriate data structures to optimize algorithms for specific problems.
4. To demonstrate proficiency in dynamic programming, greedy algorithms, and other classical algorithmic techniques.
5. To apply algorithmic solutions to real-world problems and evaluate the quality of the solution.
6. To analyze the limitations and challenges of algorithms in various contexts and assess the ethical implications of algorithm design.

Syllabus

Unit I - Fundamentals of Algorithms and mathematics

Problem, algorithm definitions, Mathematics for algorithmic sets, Functions and relations, Combinations, Vectors and matrices, Linear inequalities and linear equations

Unit II - Analysis of Algorithms

Orders of Magnitude (Asymptotic notations) Growth rates, some common bounds (constant, logarithmic, linear, polynomial, exponential) Average and worst case analysis
Analysing control statements Recurrence Relations- substitution, change of variables, master's method

Unit III - Sorting and searching algorithms

Selection sort, bubble sort, insertion sort Sorting in linear time, count sort Linear search

Unit IV - Divide and conquer algorithms
Quick sort, worst and average case complexity, Merge sort Matrix multiplication, Binary search, Binary search tree

Unit V - Greedy algorithms
General characteristics, Problem solving using Greedy methods, Activity selection problem, MST, The Knapskack problem

Unit VI - String matching
The naive string matching algorithm, The Rabin-Karp algorithm, String Matching with infinite automata

Text book(s)

"Algorithm Design" by Jon Kleinberg and Éva Tardos (Pearson, 2006)

Reference book(s)

1. "The Design of Approximation Algorithms" by David P. Williamson and David B. Shmoys (Cambridge University Press, 2010)
2. "Computational Complexity: A Modern Approach" by Sanjeev Arora and Boaz Barak (Cambridge University Press, 2009)

Tools and Techniques of Creative Thinking

Introduction

This course introduces students to a variety of tools and techniques for generating creative ideas and solving problems. Students will learn strategies for brainstorming, mind mapping, idea generation, and critical thinking. The course will focus on developing practical skills for creative thinking that can be applied in a variety of contexts.

Course outcomes

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

1. To use a variety of creative thinking tools and techniques to generate new ideas
2. To apply critical thinking skills to analyze and evaluate creative ideas and solutions
3. To identify and overcome barriers to creative thinking and problem-solving
4. To apply creative thinking skills in practical situations, such as in business, design, and technology
5. To develop effective communication skills for presenting and selling creative ideas
6. To work collaboratively with others to generate and implement creative solutions

Syllabus

Unit I – Introduction to Principles of Creativity
Mother and father of innovation, Levels of creativity, Creative environments

Unit II – Creativity tools
Creativity tools, Brainstorming techniques, Principles of brainstorming, Flip chart, Post-it, Alphabet brainstorming, Brainwriting, Grid brainstorming

Unit III – Thinking styles
The value of diversity, Principles of various thinking styles, Design thinking, Different thinking styles in practice

Unit IV – Morphological analysis
Principles of morphological analysis, Group application of plotline MA

Unit V – TRIZ theory
Principles and discussion, Contradiction matrix, TRIZ Parameters and Principles

Unit VI – SCAMPER
SCAMPER for architecture, team innovation using SCAMPER, Use of differen thinking styles

Unit VII – Using the tools in combination
Creative problem solving, Double diamond model, Circle brainstorming steps, E-tivity: B-link

Text book(s)

"The Art of Possibility: Transforming Professional and Personal Life" by Rosamund Stone Zander and Benjamin Zander; Penguin; Reprint edition (27 July 2006)

Reference book(s)

1. "Cracking Creativity: The Secrets of Creative Genius" by Michael Michalko (2001); Ten Speed Press; Revised ed. edition (26 June 2001)
2. "A Whack on the Side of the Head: How You Can Be More Creative" by Roger von Oech (2008); Grand Central Publishing; Special edition (5 May 2008)

Compiler Design

Introduction

The Compiler Design course is designed to teach students the principles and techniques used in building compilers for programming languages. Students will learn how compilers work and how to build a simple compiler from scratch.

Course outcomes

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

1. To apply the principles of formal language and automata theory in building compilers for programming languages.
2. To design and implement the lexical analyzer and parser for a programming language.
3. To generate intermediate code from source code using different techniques.
4. To optimize code generation using various optimization techniques.
5. To implement error handling and debugging mechanisms in a compiler.
6. To evaluate and compare different compiler design and optimization techniques.

Syllabus

Unit 1 Introduction and Directed Translator

Language processors, Structure of a compiler, Evolution of programming languages, The science of building a compiler, Applications of computer technology, Syntax – Directed translation, Parsing, A translator for simple expressions, Lexical analysis, Symbol tables, Intermediate code generation

Unit 2 - Lexical analysis

The role of lexical analyzer, Input buffering, Tokens, Lexical-Analyzer generator, Finite Automata, Design of a lexical-analyzer generator

Unit 3 - Syntax analysis

The role of parser, Context-free grammar, Writing a grammar, Top-down parsing, Bottom-up parsing, LR parsing, Parser generators

Unit 4 Syntax directed translation

Evaluation order for SDDs, Applications, Schemes, Implementing L-attributed SDDs

Unit 5 Intermediate Code Generation

Variations of syntax trees, Three address code, Types and declaration, Translation of expressions, Type checking, Control flow, Backpatching, Switch statements

Unit 6 Run time environments

Issues in the design of a code generator, The target language, Addresses in the target code, Basic blocks and flow graphs, Optimization of basic blocks, A simple code generator, Peephole optimization, Dynamic Programming code-generation

Text book(s)

Compilers: Principles Techniques and Tool; Alfred V. Aho, Monica S. Lam, Ravi Sethi, Jeffry D. Ullman; Pearson 2nd edition

Reference book(s)

1. "Engineering a Compiler" by Keith D. Cooper and Linda Torczon (2nd Edition, 2011, Morgan Kaufmann)
2. "Modern Compiler Implementation in Java" by Andrew W. Appel (2nd Edition, 2002, Cambridge University Press)
3. "Introduction to Compiler Construction" by Thomas W. Parsons (2001, Addison-Wesley)
4. "Language Implementation Patterns: Create Your Own Domain-Specific and General Programming Languages" by Terence Parr (2010, Pragmatic Bookshelf)
5. "Writing Compilers and Interpreters: A Software Engineering Approach" by Ronald Mak (3rd Edition, 2009, Wiley)

Fundamentals of Business Management

Introduction

This course provides an introduction to the key principles and practices of modern business management. It is designed to equip students with the foundational knowledge and skills needed to manage people, resources, and processes effectively, and to develop a strategic mindset that enables them to make sound business decisions in a dynamic and complex environment.

Course outcomes

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

1. To understand the fundamental concepts and principles of modern business management, including organizational structures, functions, and processes.
2. To develop a strong foundation in core business disciplines, such as accounting, finance, marketing, and operations management.
3. To acquire the necessary knowledge and skills to manage people effectively, including leadership, motivation, and communication.
4. To understand the importance of strategic thinking and planning in business management, and to develop the ability to formulate and execute effective business strategies.

5. To learn to manage resources efficiently and effectively, including financial, technological, and human resources.
6. To develop critical thinking and problem-solving skills, and to apply them in real-world business scenarios.

Syllabus

Unit 1: Introduction to business management and Marketing
How business operate, Branding, customer centricity, Go-to market strategies, different modes of marketing

Unit 2: Financial Accounting
Balance sheet, Accrual accounting, Income sheet, Cash flows, Ratio analysis

Unit 3: Managing social and human capital
Motivation and reward, Tasks, jobs and system of work, Making good & timely decisions, Designing and changing the organization's architecture

Unit 4: Background to Entrepreneurship
Theories of entrepreneurship, Activity scope and role in modern society, The effects of entrepreneurial activity on economic systems, contributions and benefits of entrepreneurial activity, Entrepreneurship in Oman. Problems faced by small firms, types of entrepreneurs and innovators.

Unit 5: The Entrepreneur
The individual in terms of psychology, personality and trait theories; The individual in terms of motivation and achievement theories; The individual in terms of behaviour and characteristics theories; Differentiated entrepreneur typologies.

Unit 6: Conceiving a Business Idea and Creating a Business Plan
Business Idea - Models for new ventures, idea generation, screening ideas, business analysis, and feasibility studies.
Business Plan - Purpose and benefits, design of a business plan, layout and content, focused recipient, approaching potential investors.

Text book(s)

1. "Principles of Management" by Peter F. Drucker (HarperCollins, 2017)
2. "Fundamentals of Management" by Stephen P. Robbins and David A. DeCenzo (Pearson, 2017)

Reference book(s)

1. "The Lean Startup" by Eric Ries (Crown Business, 2011)
2. "Blue Ocean Strategy" by W. Chan Kim and Renée Mauborgne (Harvard Business Review Press, 2015)
3. "The 7 Habits of Highly Effective People" by Stephen R. Covey (Simon & Schuster, 2013)
4. "Good to Great" by Jim Collins (Harper Business, 2001)

Semester 7

This section of the doc contains a tentative list of courses offered (not a detailed syllabus) for the electives.

Academic Elective 1

1. Cloud computing
2. Distributed systems
3. Data Mining and Warehousing

Academic Elective 2

1. Cryptography
2. Internet of Things
3. System Design

Foundation Elective

1. Human Mind and Behaviour
2. Organization Behaviour
3. Foreign language (Global options provided such as French/ Spanish/ German/ Japanese etc.)
4. Design Thinking 101

Skilling Elective

1. Unix Shell Programming
2. AWS and AWS Security
3. Data Modeling and Visualization