Data Structure
&
Algorithm Analysis

Chandan Mazumdar

Professor

Department of Computer Science & Engineering

Jadavpur University

Calcutta 700 032

Program Outcomes for all JU Engineers

- Engineering Knowledge
- Problem Analysis
- Design / Development of Solutions
- Conduct Investigations of Complex Problems
- Modern Tool Usage
- The Engineer and Society
- Environment and Sustainability
- Ethics
- Individual and Team work
- Communication
- Project Management and Finance
- Life-long Learning

Program Specific Outcomes for CSE Engineers

- **Software Development:** Designing Algorithm, Analyzing Complexity, and Developing cost-effective system
- **Hardware Design:** Designing Cost-effective energy efficient hardware
- **Societal Outreach:** Applying computational methods to address diverse needs of the community for improving the quality of life and environment.
- **Professionalism:** Developing sense of responsibilities, professional ethics, communication skills, competence, environmental awareness and self-learning

Three Domains of Learning

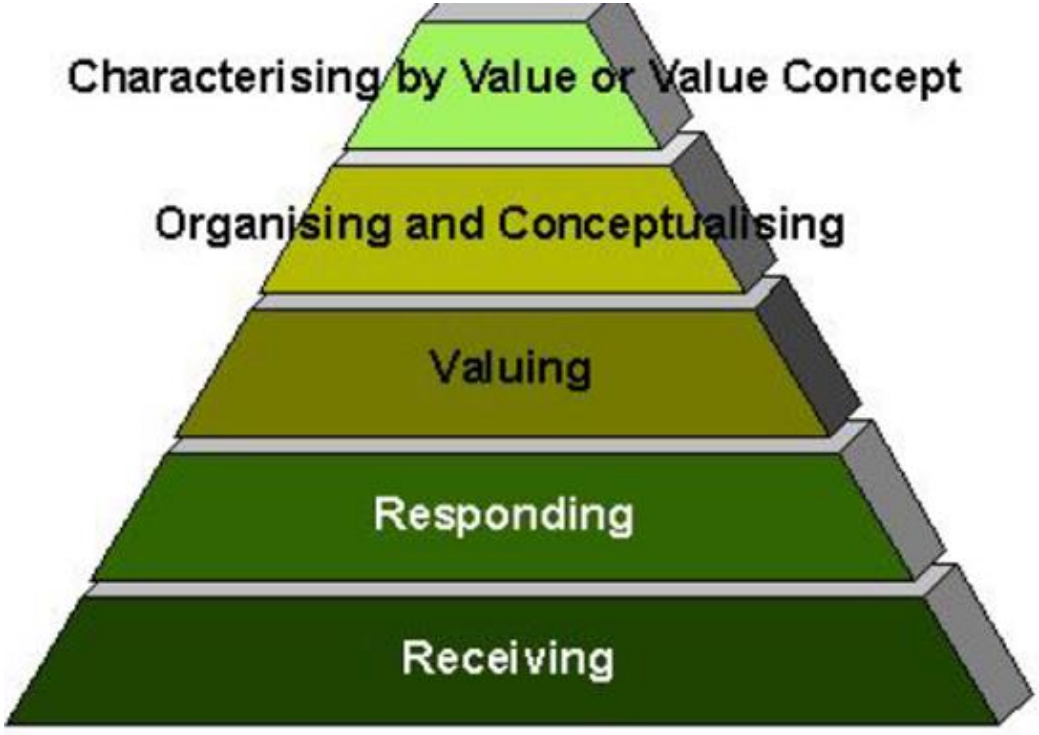
- **Cognitive Learning -** The mental or intellectual thinking behaviors demonstrated by an individual

- **Affective Learning -** An individual's emotions, attitudes, appreciations, interests, and/or values about "something" or someone

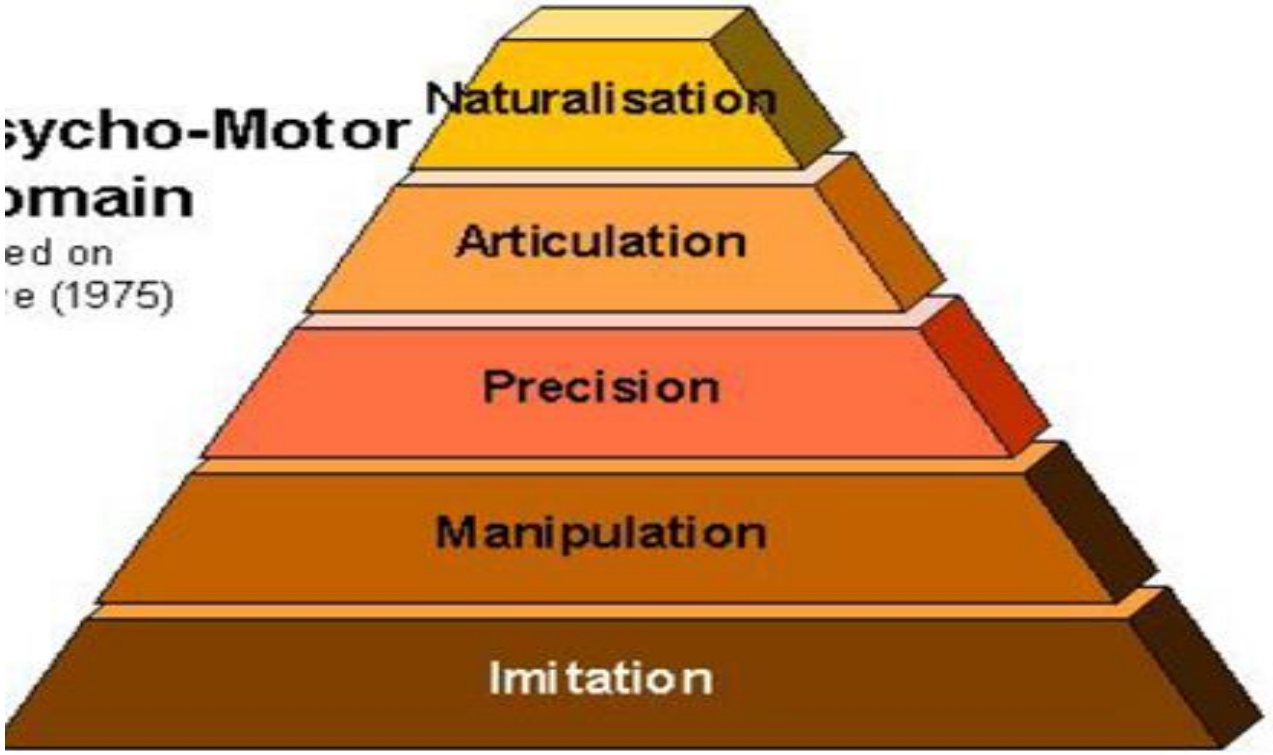
- **Psychomotor Learning -** Physical activities involving gross and/or fine motor skills, such as coordination, dexterity, strength, manipulation, and speed

Knowledge Levels (K1 – K6)

Affective Levels (A1 – A5)



Psycho-motor/Skill Levels (S1 – S5)



Course Outcomes of

Data Structures and Algorithms

- **CO1** Understand Data Type Abstraction for Linear and Non-linear Data Arrangements K2
- **CO2** Implement Static and Dynamic, Linear and Non-linear Data Structures K3-K5
- **CO3** Design algorithms using elementary algorithm design strategies K5
- **CO4** Analyze and implement algorithms and data structures for time and memory usage efficiency K5, K6
- **CO5** Solve computing problems by selecting / developing data structures and algorithms for efficient implementation K3-K6

Syllabus

- 1. Information, Data, Data Types, Abstract Data Type (ADT), Data Structure, Static and Dynamic Data Structures, Implementation Methods [CO1]

- 2. Array as an ADT, Single and Multidimensional Arrays, Structures, ADT Polynomial, Sparse Matrix and List using Arrays. [CO1, CO2]

- 3. Pointers, ADT Linked List, Singly Linked List, Doubly Linked List, Multi-linked List. Implementations using Pointers and Arrays. Application in implementing Polynomials, Sparse Matrix, etc. [CO1, CO2, CO4]

- 4. Algorithm Design Methodologies – Divide & Conquer, Greedy Algorithms, Dynamic Programming, Backtracking, Exhaustive Search, Probabilistic Algorithms. [CO3, CO4]

Syllabus (contd)

- 5. Analysis of Algorithms, Big O notation, Introduction to analysis of Sequential, Iteration and Recursive Algorithms with examples. Measurement of program efficiency. [CO3, CO4]

- 6. Development, Implementation, Analysis and Measurement of Searching and Sorting Algorithms – Linear and Binary Search, Insertion Sort, Selection Sort, Merge Sort, Quick Sort, Heap Sort, Counting Sort. [CO3, CO4, CO5]

- 7. ADT Stack and Queue – Implementation using Arrays and Pointers, Priority Queue, Applications. Implementation of Recursive Functions using Stack [CO1, CO2, CO4, CO5]

- 8. ADT Tree, Binary Tree, Binary Search Tree, Height Balanced Tree, 2-3 Tree, B-Tree,. Applications. [CO1, CO2, CO4, CO5]

Syllabus (contd)

- 9. ADT Graph, Representations of Graph Data Structures, Graph Algorithms – Depth-First and Breadth-First Search, Spanning Tree – Kruskal and Prim's Algorithm, Finding Minimum Cost Paths, Applications.

- 10. ADT Hash Table – Hash Functions, Synonyms, Collisions, Example Hash Functions, Collision Resolution Strategies, Applications. [CO1, CO2, CO4, CO5]

- 11. Advanced Topics – B+ Tree, Bloom Filters, Applications. [CO1- CO5]

Books

- 1. Fundamentals of Data Structures in C – Horowitz, Sahni, Anderson-Fred, Latest Edition. (* Textbook)
- 2. Data Structures and Algorithm Analysis in C
- by Mark Alan Weiss, 2nd ed., Pearson Education (#)
- 3. Data Structues and Algorithms
- by Aho, Hopcroft & Ullman
- 4. Data Structures and Program Design
- by Kruse et. al., PHI
- 8. Algorithms + Data Structures = Programs (#)
- by N. Wirth, PHI
- 9. How to solve it by Computers (#)
- by Dromey, PHI
- # A few chapters will be referred

Data Structures and Algorithms Lab

This Lab course comprises of a series of assignments on Data Structures and Algorithms to be implemented in C language on Linux Operating System. The assignments will follow the progress in the Theory course. They will include implementation of ADTs, Algorithms and applications to solve real-life problems. Measurement of performance of the programs developed will also be part of the assignments.

Course Outcomes of DSA Lab

- **CO1**. Implement a given ADT as a data structure in C language
- **CO2.** Implement a given Algorithm developed using a design strategy in C language
- **CO3.** Choose the appropriate data structure and algorithm design method for a specified application to be developed in structured and modular form
- **CO4.** Apply systematic testing and debugging methods on developed applications and measure the performance of them
- **CO5.** Write Reports in acceptable format for the applications developed

Why C?

- It is the de facto substandard of programming languages
  - C runs on everything.
  - C lets you write programs that use very few resources.
  - C gives you near-total control over the system, down to the level of pushing around individual bits with your bare hands.
  - C imposes very few constraints on programming style: unlike higher level languages, C doesn't have much of an ideology. There are very few programs you can't write in C.
  - Many of the programming languages people actually use (Visual Basic, perl, python, ruby, PHP, etc.) are executed by interpreters written in C (or C++, an extension to C).

Why C?

- You will learn discipline
  - C makes it easy to shoot yourself in the foot
  - You can learn to avoid this by being careful about where you point it.
  - Pain is a powerful teacher of caution.

Why not C?

- It's missing a lot of features of modern program languages, including:
  - A garbage collector
  - Minimal programmer-protection features like array bounds-checking or a strong type system.
  - Non-trivial built-in data structures.
  - Language support for exceptions, namespaces, object-oriented programming, etc.

The C Programming Language (2nd Edition), by Brian W. Kernighan and Dennis M. Ritchie. Prentice Hall, 1988.