Data Structures

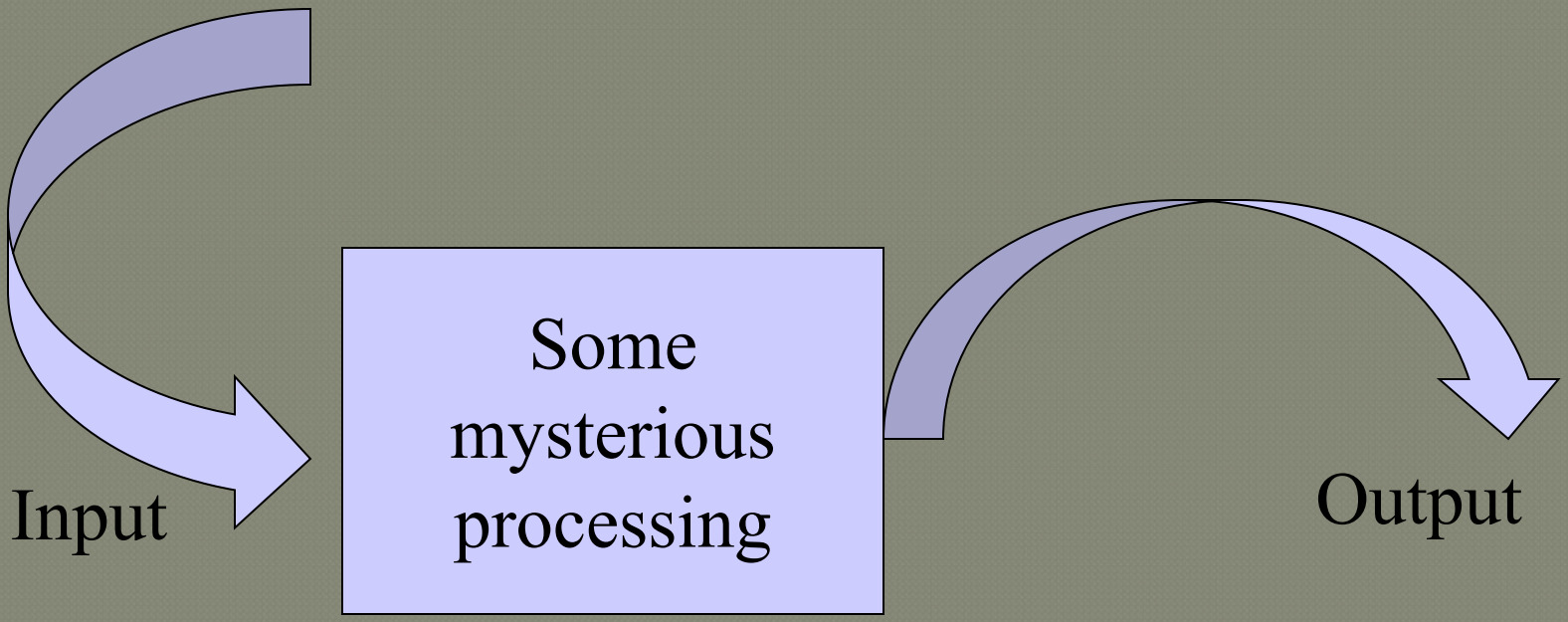
Unit -I

Course Contents

- Data Types
  - Abstract data types (ADT)
  - Pointers
  - Pointers to Arrays, Structures and Functions
- Linked Lists
- Stack
- Queues
- Trees
- Graphs

What is a Computer Program?

- To exactly know, what is data structure? We must know:
  - *What is a computer program?*



Introduction to Data Structures

- What is Data ?
  - Any useful information – that can be stored or memorized for future reference
    - In an organization it can be
      - Employee's name, age, department, address so on
    - In Railway reservation office
      - Passenger name, traveling date, traveling time, seat number so on
    - In Computers
      - Set of integers, Structure(s) or any other user defined data type

Definition

- An organization of information, usually in memory, for better algorithm efficiency

  such as queue, stack, linked list, heap, dictionary, and tree,

- In computer science, a **data structure** is a particular way of storing and organizing data in a computer's memory so that it can be used efficiently.

- Data may be organized in many different ways; the logical or mathematical model of a particular organization of data is called a **data structure.**

- Different operations are defined for each type of data structure to add, delete or modify its content

Data Structures: More specifically

- A data structure is a way of grouping fundamental types (like integers, floating point numbers, and arrays) into a bundle that represents some identifiable thing.

Why Study?

- Designed to develop students understanding the impact of ***structuring data to achieve efficiency*** of a solution to a problem

- After completion you will be familiar with important and most often used data structuring techniques.

- It will enable you to understand the manner in which data is organized and presented later.

Objectives of the course

- Present in a systematic fashion the most commonly used data structures, emphasizing their abstract properties.
- Discuss typical algorithms that operate each kind of data structure, and analyze their performance.
- Compare different Data Structures for solving the same problem, and choose the best

3 steps in the study of data structures

- Logical or mathematical description of the structure
- Implementation of the structure on the computer
- Quantitative analysis of the structure, which includes determining the amount of memory needed to store the structure and the time required to process the structure

- Data may be organized in many ways
  - E.g., arrays, linked lists, trees etc.
- The choice of particular data model depends on two consideration:
  - It must be rich enough in structure to mirror the actual relationships of data in the real world
  - The structure should be simple enough that one can effectively process the data when necessary

Data strucure Classification

- **Primitive vs Non primitive**
  Based on data type used to organize the data

  primitive : Uses primitive type such as int float etc.

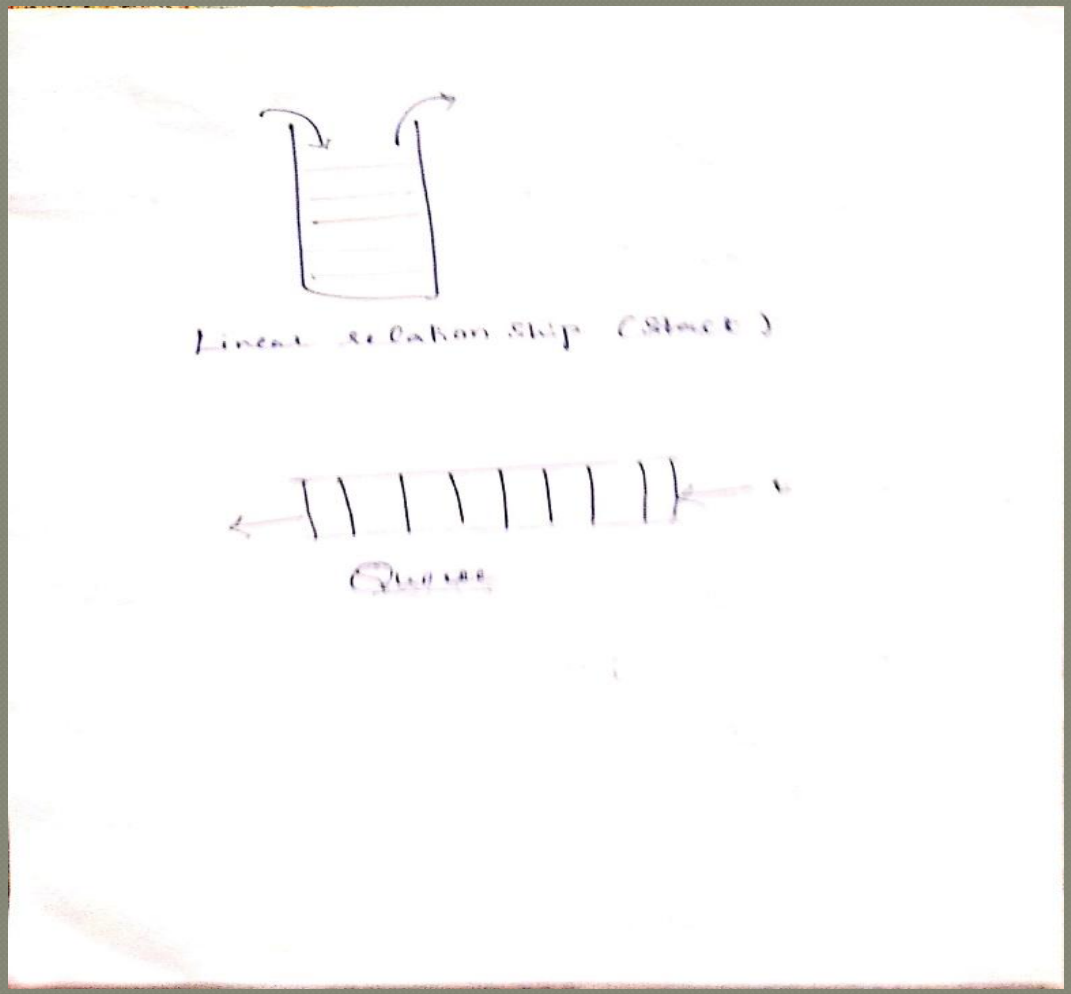
  Non primitive: Uses non primitive type such as derived or user defined types like array, union, structure

- **Linear vs Non Linear**
  Based on organization fashion.

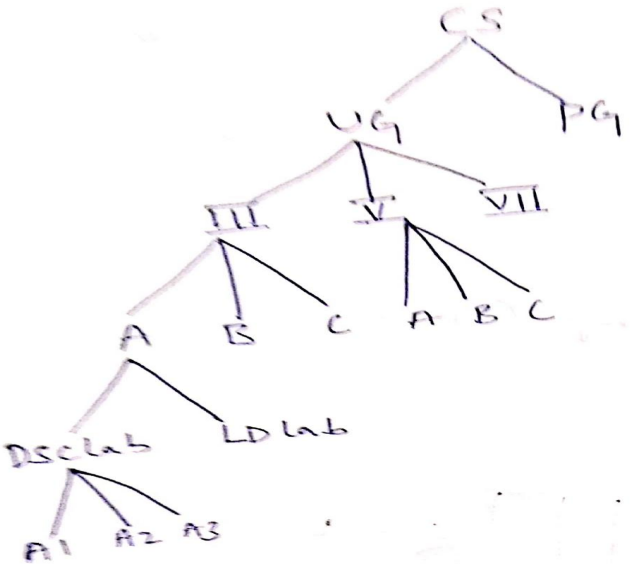
  Linear Data structure: A data structure is said to be linear if its elements form a sequence. The elements of linear data structure are representd by means of sequential memory locations
  eg: stack, queue

  Non linear Data structure: Data items are organized in non linear fashion, i.e , not in sequential order
  A data structure is said to be non-linear if its elements show a hierarchical /adjacency relationship between each other. All elements assign the memory as random form and you can fetch data elements through random access process.

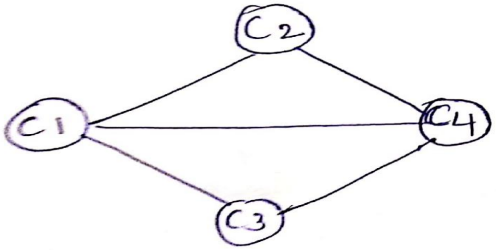
  eg : Tree, graphs.



Tree



Hierarchical Relationship



Graph

Adjacency relationship

Common operations on list of items

- Retrieval
- Searching
- Sorting
- Insertion
- Deletion
- Updation
- Merging
- Traversing(visiting the elements)

A Real Life Example

Electronic Phone Book
Contains different **DATA**:

- names
- phone number
- addresses

Need to perform certain **OPERATIONS**:

- add
- delete
- look for a phone number
- look for an address

How to organize the data so to optimize the efficiency of the operations

- **Lisa**
- **Michele**
- **John**
- **110**
- **622-9823**
- **112-4433**
- **75**
- **Bronson**
- **Paola**

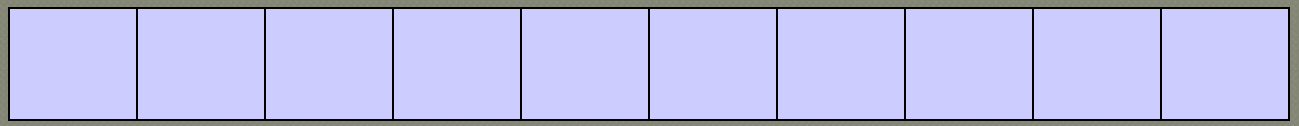
Two basic representations for storing the data

- Arrays
- Linked Lists

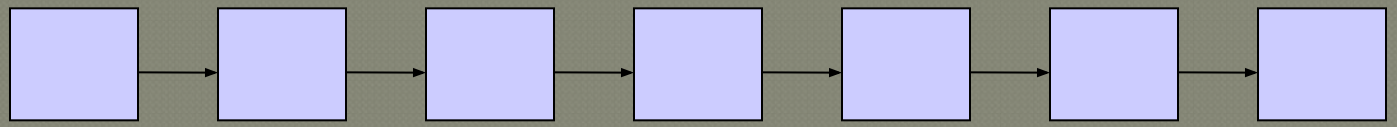
- Issues
  - Space needed
  - Operations efficiency (Time required to complete operations)
    - Retrieval
    - Insertion
    - Deletion

What data structure to use?

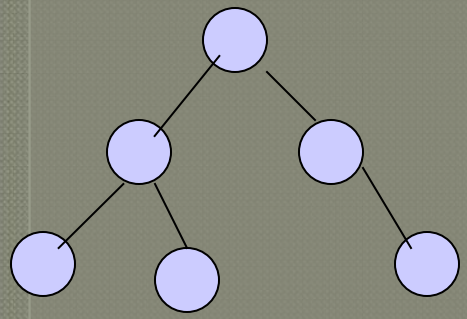
Data structures let the input and output be represented in a way that can be handled efficiently and effectively.



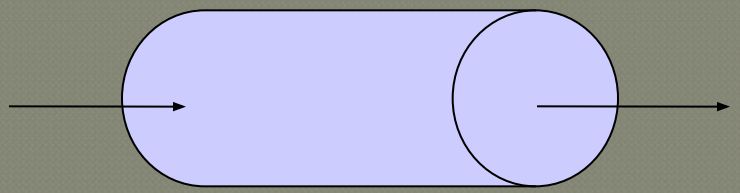
array



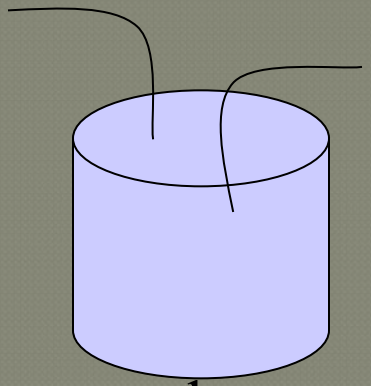
Linked list



tree



queue



stack

Abstract Data Types (ADT)

- This is a higher level, domain and operations independent specification of data
  - While working on an "integer" type we don't worry about its low level implementation (can be 32 bit, or 64 bit number)
  - **Distance = rate * time** (rate and time can be of type integer or real numbers)
  - Employee record means same to the bosses of two different organizations, no matter how differently they are stored by the admin officers

Abstract Data Type ADT

- A data type whose properties (domain and operations) are specified independently of any particular implementation

- ADT is a mathematical model which gives the a set of utilities available to the user but never states the details of its implementation.

Algorithms

- The operations defined on the data structures are generally knows as algorithms
- For each data structure there has to be defined algorithms
- Algorithms are normally used to add, delete, modify, sort items of a data structure
- All programming languages are generally equipped with conventional data structures and algorithms.
- New data structures and the algorithms for manipulation can be defined by the user

The first Data Structure

- An Array!

Word about Arrays!

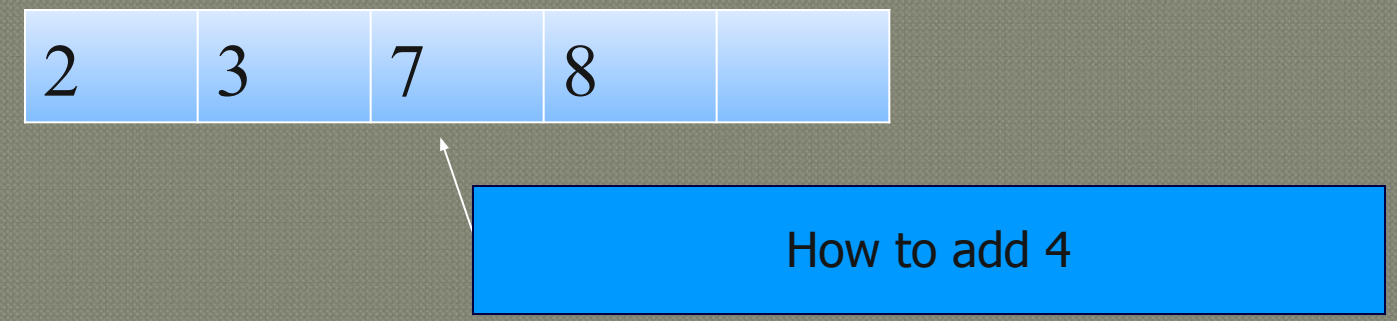
- Lets Get Started:

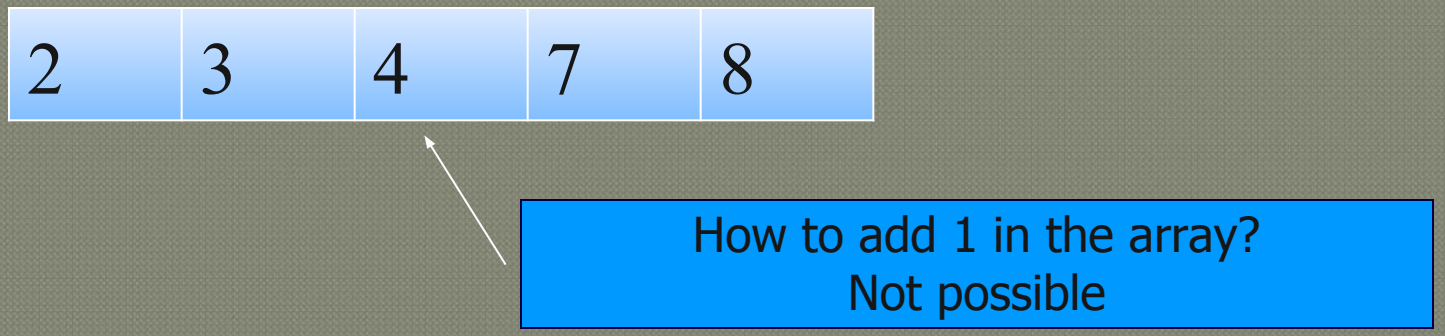
- Arrays are data structures
  - Finite
  - Contiguous
  - Fast
  - Direct Access
  - All elements of same data type
  - Insertion / Deletion ??? HOW??

Arrays

- How to Input arrays
- How to process arrays
- How to insert an item in an array
- How to delete an item from an array
- How to pass an array

Array





The first Data Structure

- An Array!

  The simplest form of an Array is a *one dimensional array* that may be defined as a finite ordered set of homogenous elements

- For Example

  int a[100];

Basic Operations

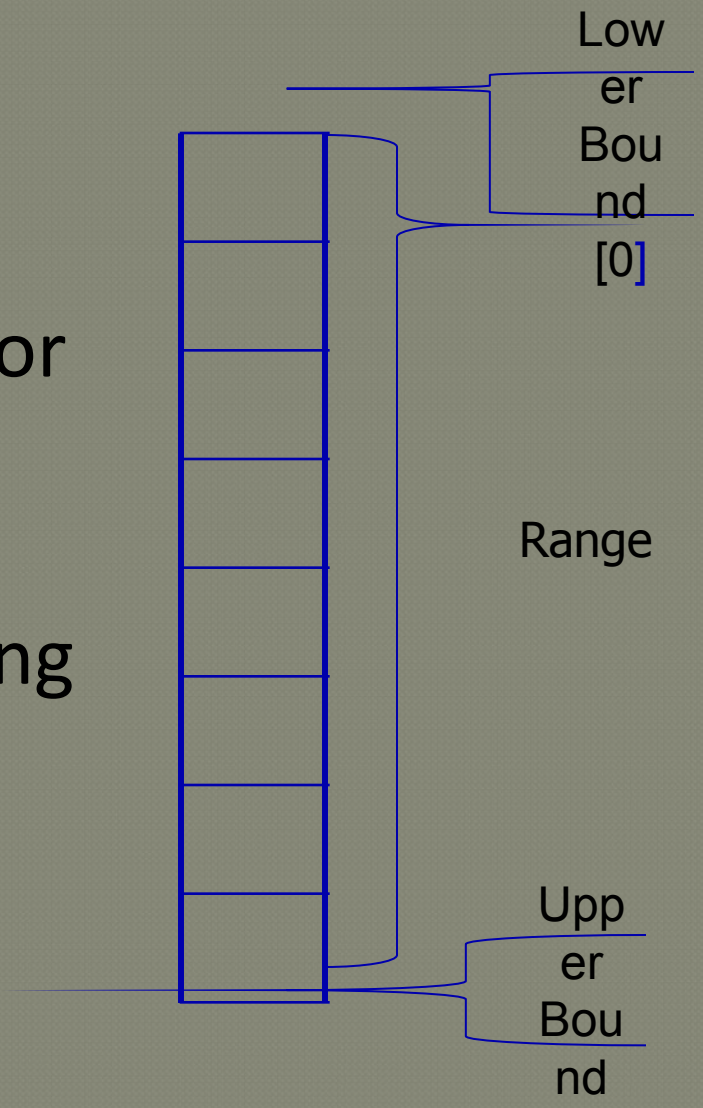
- Extraction
  - A function that accepts an array “a” and an index “i”, and returns an element of the array.
  - Example

    a[i]
- Storing
  - It accepts array “a” an index “i” and an element x.
  - Example
  - a[i]=x

One Dimensional Array

- range = *upper - lower+1*

- Neither the *upper bound* nor the *lower bound* can be changed and as well as the *range* can be changed during the program execution.



Implementation of 1Dimenional Array

- int b[100];
- Reserves 100 successive locations, each large enough to contain a single integer.
- The address of the first of these locations is called the *base Address: base(b)*
- Reference to element b[0] is to the element at location *base(b)*
- Reference to b[1] is to the element at

  base(b) + 1 esize*

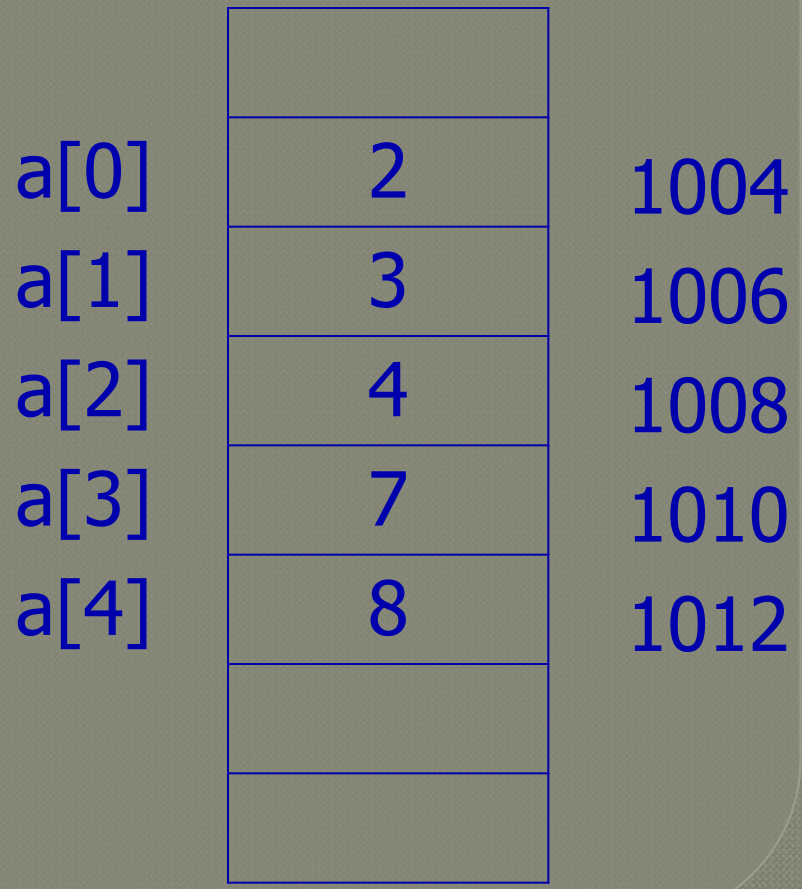
Hence

b gives you the starting memory address of the array b

Memory view of an array

int a[5]

2	3	4	7	8



Draw backs of array representation

- Size is always fixed(Static or Dynamic allocation)
- Insertion and deletion requires shifting of data. So, extra time will be spent in movement of data.
- By deleting the data item we can't delete the memory allocated for that item. Memory gets deleted only when the contiguous block of memory is deallocated(implicitly(static)/ Explicitly(dynamic))
- We can't make the list to grow dynamically with array representation.

Example

- To under stand the difficulties of with arrays ,insertion and deletion operations on arrays are discussed in class(Refer program)