Data Structures
CS-2001

Introduction

Guidelines

- Be attentive in class and always expect a quiz
- There will be no retake of any quiz/activity, whatever circumstances are
- Strictly follow your deadlines otherwise be ready for penalties
- Follow SLATE/Classromm and report your query in time otherwise don’t complaint

Guidelines

- There will be no re-evaluation for any marks, when report time is over
- If you don’t want to study leave the class, let others study.
- You will be given zero (negative marks) for copy cases (Both parties)

Important

Classroom Conduct

- All students are expected to behave as scholars at a leading institute of technology. This includes arriving on time, not talking during lecture (unless addressing the instructor), and not leaving the classroom before the end of the lecture.
- Disruptive students will be warned and potentially dismissed from the classroom

Academic Dishonesty

- Academic dishonesty in any portion of the academic work for a course shall be grounds for awarding a grade of F for the entire course

Some Rules

- Raise your hand before asking any question and then WAIT for the permission

- Never ever Miss a class

- Never ever “sleep” in the class

- Never even think about using mobile during the class

Distribution (Tentative)

- Attendance 100%
- Weight-ages

Assignments / Home Work	5 %	3 to 5
Quizzes	5%	7 to 10
Sessional-I	15%	1
Sessional-II	15%	1
Project	10%	2 to 3 phases
Final	50%	1

Assignment/Quizzes

- A number of assignments and quizzes will be taken
- Announced and/or unannounced quizzes may be given to students any time during the lecture

Submissions

- All submissions (Assignments, Home Works, Project) would be on Slate/Classroom, unless otherwise announced.
- The file naming convention should be as:
  Roll Number_Assignment_Number_Course
- For example: 23_1234_Assignment_2_DS

Prerequisite

- Enforced
  - CS1004 Object Oriented Programming
  - CL1004 Object Oriented Programming

Why should we study this course?

- Well, because it is the core computer sciences course
- Any other reason to study this course?
- We want to make a successful career after graduation
- The most common programming interviews questions
  - Linked lists
  - Strings
  - Binary Trees
  - Arrays
  - Queues

Source: http://maxnoy.com/interviews.html

Course Objective

- Introduce the basic concepts of data structures /ADTs, and use them efficiently in algorithms for solving various problems using C/C++

- What should you expect in this course?
  - Extensive programming
  - A lot of thinking

- What should you learn by the end of this course
  - Ability to understand common programming problems and design and implement efficient data structures to solve them

Books

- Data Structures
  - Seymour Lipschutz
- Data Structures Using C and C++
  - Y. Langsam, M. J. Augenstein, A. M. Tenenbaum

Reference Books

- Introduction to Algorithms
  - Thomas H. Cormen et al
- Data Structures and Algorithms
  - A. V. Aho, J. E. Hopcroft, J. D. Ullman

Course Outline

- In this course you will learn
  - Introduction
  - Algorithm Analysis
  - Abstract Data Types (ADTs)
  - Lists
  - Stacks
  - Queues
  - Trees
  - Hashing
  - Sorting
  - Graph

What is a data structure exactly?

Definition:
A data structure is a particular way of organizing data in a computer so that it can be used effectively.

What is a Data Structure?

Definition:

A data structure is a specialized format for organizing, managing, and storing data in a computer so that it can be accessed and modified efficiently.

Importance:
Data structures are crucial in developing efficient software and solving problems.

Choosing the right data structure can improve the performance of software applications, optimize memory usage, and enhance processing speed.

They are foundational to designing efficient algorithms, enabling programs to handle large amounts of data effectively.

For instance, searching for an item in an unsorted list is inefficient, but using a structured approach like a binary search tree can significantly speed up the process.

Definition

- Data may be organized in different ways
- Data structure is the logical or mathematical model of a particular organization of data
- Must be rich enough to mirror the actual relationships of the data in the real world sample

What is a Data Structure?

- Data structures are used in almost every program or software system
- In computer sciences, a data structure is a **particular way of storing and organizing data in a computer** so that it can be used efficiently
  - Efficient data structures → Efficient algorithms
- **Focus:** Efficiency and performance
  - Time and space analysis of algorithms

Source: http://en.wikipedia.org/wiki/Data_structure

Classification of Data Structures

1. **Primitive Data Structures**: These are the most basic data types provided by a programming language. Examples include:
   1. Integer (int)
   2. Float (decimal numbers)
   3. Character (char)
   4. Boolean (true/false)
2. **Non-Primitive Data Structures**: These are more complex data structures that are derived from primitive data types. Examples include:
   1. Arrays
   2. Lists (Linked Lists)
   3. Stacks
   4. Queues
   5. Trees
   6. Graphs

Linear vs. Non-Linear Data Structures:

•**Linear Data Structures**: Elements are arranged sequentially, and each element is connected to its previous and next element. Examples include:

- Arrays: A collection of elements identified by index or key.
- Linked Lists: A sequence of elements where each element points to the next.
- Stacks: A collection of elements that follows the Last In, First Out (LIFO) principle.
- Queues: A collection of elements that follows the First In, First Out (FIFO) principle.

•**Non-Linear Data Structures**: Elements are not arranged sequentially. Examples include:

- Trees: A hierarchical structure where data is organized in nodes, and each node can have multiple children.
- Graphs: A collection of nodes (vertices) connected by edges, used to represent networks.

Why Learn Data Structures?

Understanding data structures is fundamental for programming, software development, and computer science in general.

Efficient Problem Solving:

Data structures provide the means to manage and organize data efficiently, which is crucial for developing optimized solutions.

Algorithm Design:

Most algorithms rely heavily on data structures.Understanding the right data structure to use in a specific scenario is key to writing efficient algorithms.

Real-World Applications:

Data structures are widely used in real-world applications such as search engines (e.g., Google), databases, operating systems, and even artificial intelligence.

Critical for Interviews:

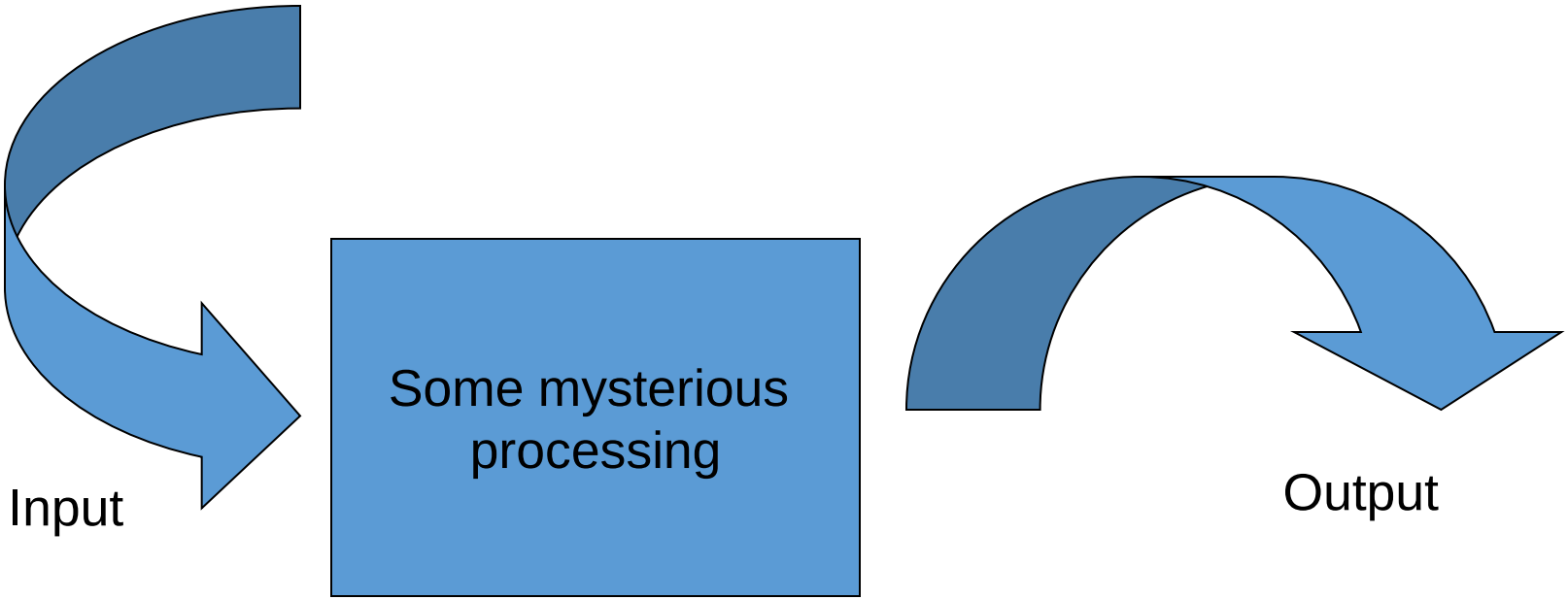
Mastery of data structures is often a significant focus of technical interviews in the software industry.

Abstract data type

- In computer science, an **abstract data type** (ADT) is a mathematical model for data types, where **a data type** is defined by its behavior (semantics) from the point of view of a user of the data, specifically in terms of possible values, possible operations on data of this type, and the behavior of these operations
- ADT users are NOT concerned with how the task is done but rather what it can do.
- An abstract data type is a data declaration packaged together with the operations that are meaningful for the data type.

To answer that, we must first understand:

What is a computer program?



How to solve the following problems:

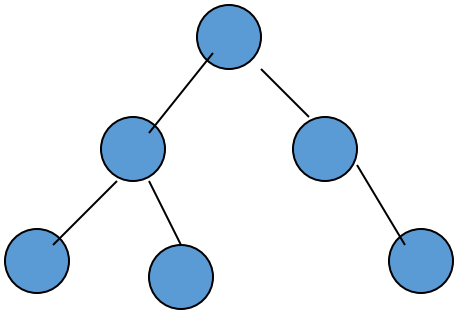
1. Input 3 numbers, print out the maximum.

2. Input 30000 numbers, print out the largest 10 numbers.

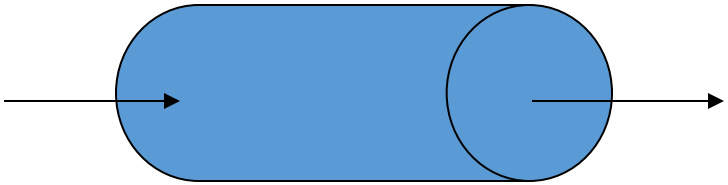
Data structures let the input and output be represented in a way that can be handled efficiently and effectively.

array

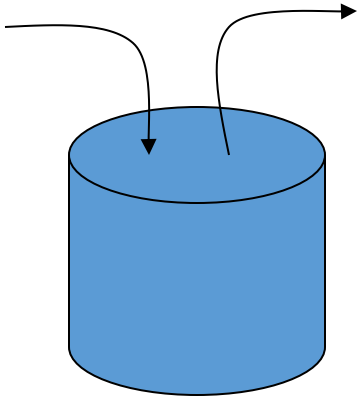
Linked list



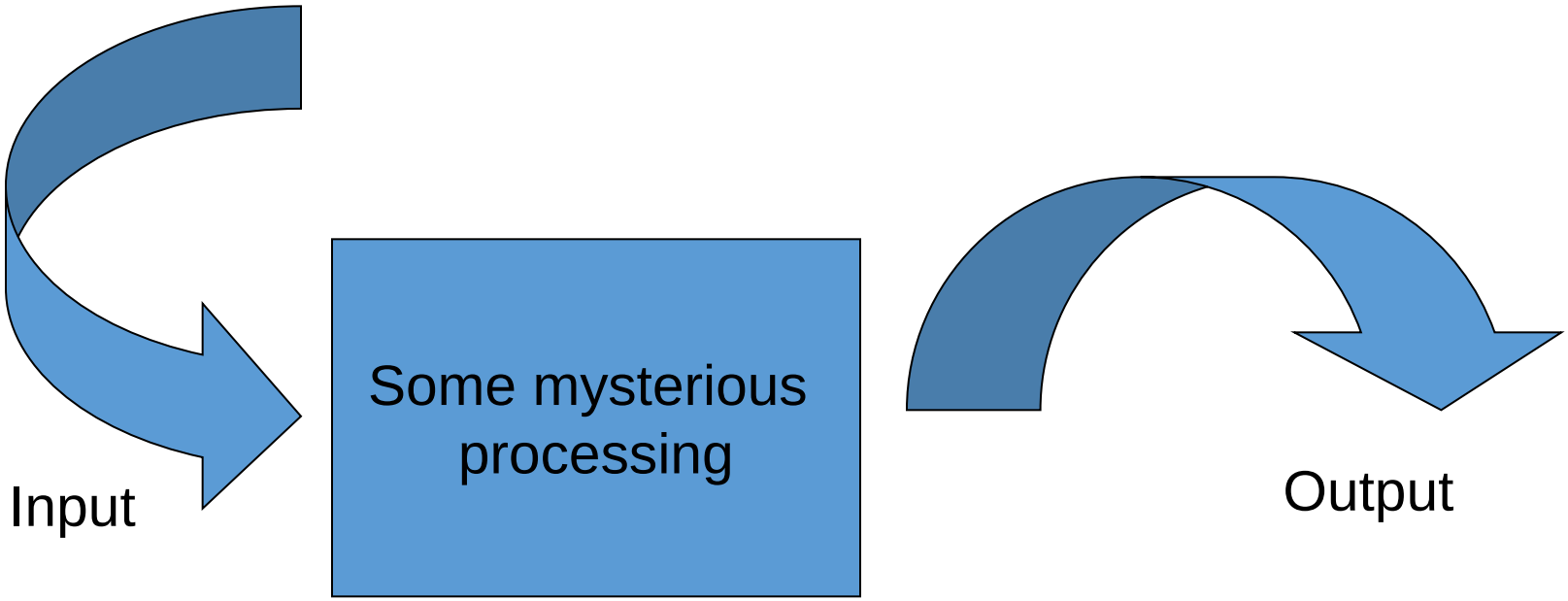
tree



queue

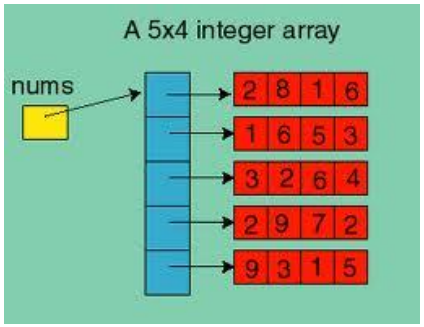


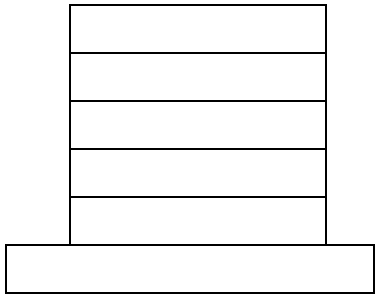
stack

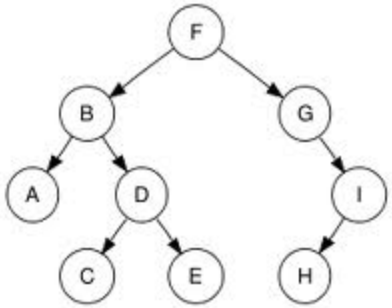


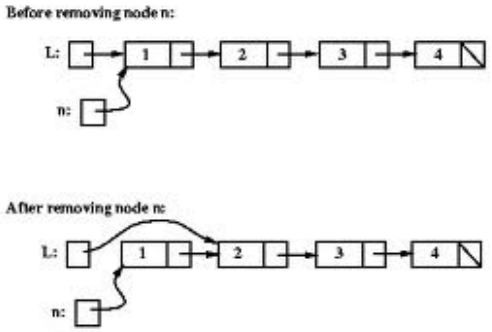
Data structures+Algorithms=Programs

Some Example Data Structures









Arrays Stack Tree Linked List

Data structure = representation and operations associated with a data type

Data Structure Applications

- Arrays
  - Consecutive memory locations
  - lists (one dimensional arrays)
  - matrices (two dimensional arrays)
  - database applications
  - to implement other data structures, such as heaps, hash tables, queues, stacks, strings
- Stacks
  - expression evaluation and syntax parsing
- Queues
  - scheduling
  - transportation
  - operations management

Data Structure Applications

- Trees
  - efficient searching of data (Binary search tree)
  - manipulate hierarchical data
  - manipulate sorted data
- Linked lists
  - can be used to implement several other common abstract data structures, such as
    - stacks
    - queues
    - symbolic expressions, and etc

Arrays

Linear Arrays

Array

Ali				
Salam		A	B	C
Usman				
Danial				
Saeed		Y		
Zaka		X		
Ali		Y		
Salam		X		
Usman		Y		
Danial		X		
Saeed		Y		
		X		

Arrays

Sale			
Item 1	2	8	85
Item 2	5	8	5
Item 3	5	8	5
Item 4	4	8	5
Item 5	5	5	5

Two Dimensional Arra

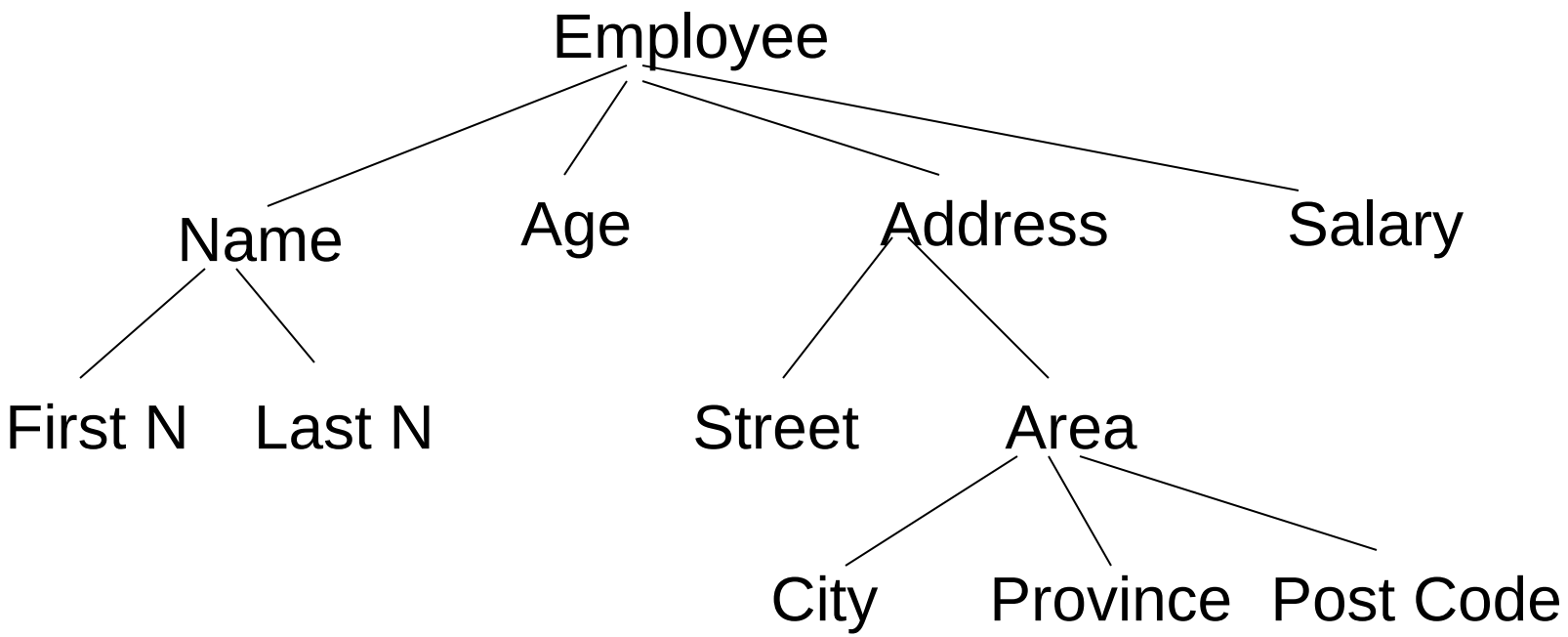
Arrays

	Customer	Salesperson
1	Jamal	Tony
2	Sana	Tony
3	Saeed	Nadia
4	Farooq	Owais
5	Salman	Owais
6	Danial	Nadia

Link List

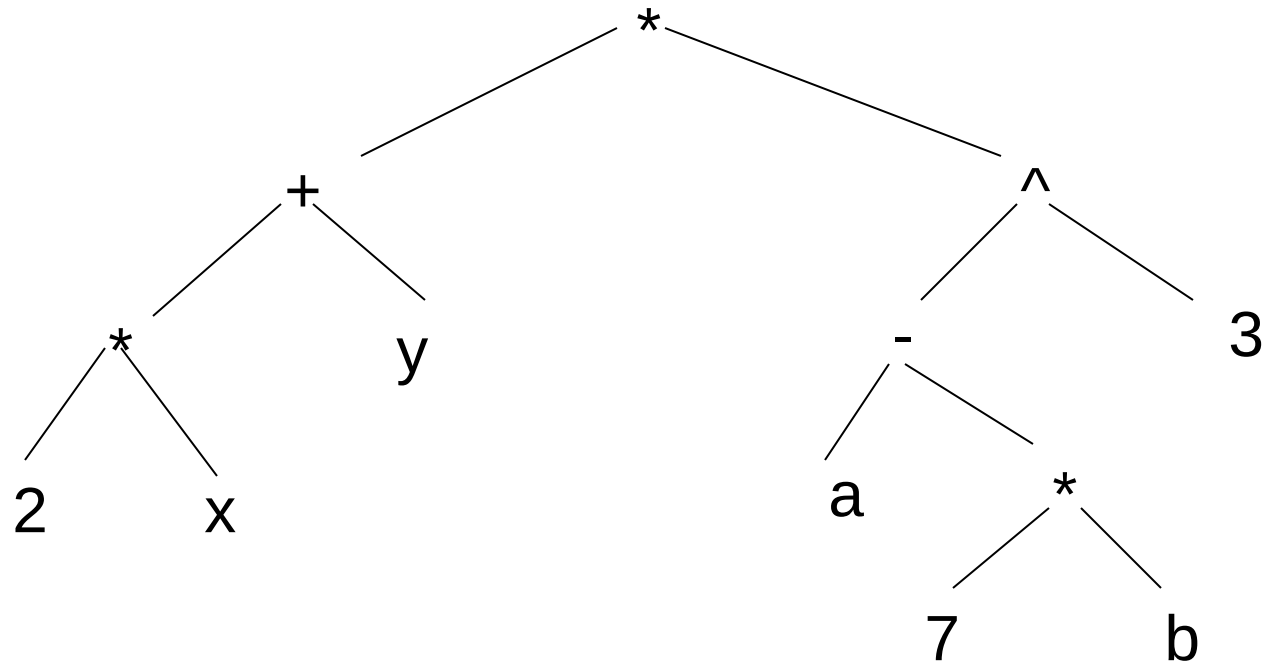
Jamal	1			
Sana	1			
		Saeed	2	
Farooq	3			
Salman	3		Salman	3

Trees

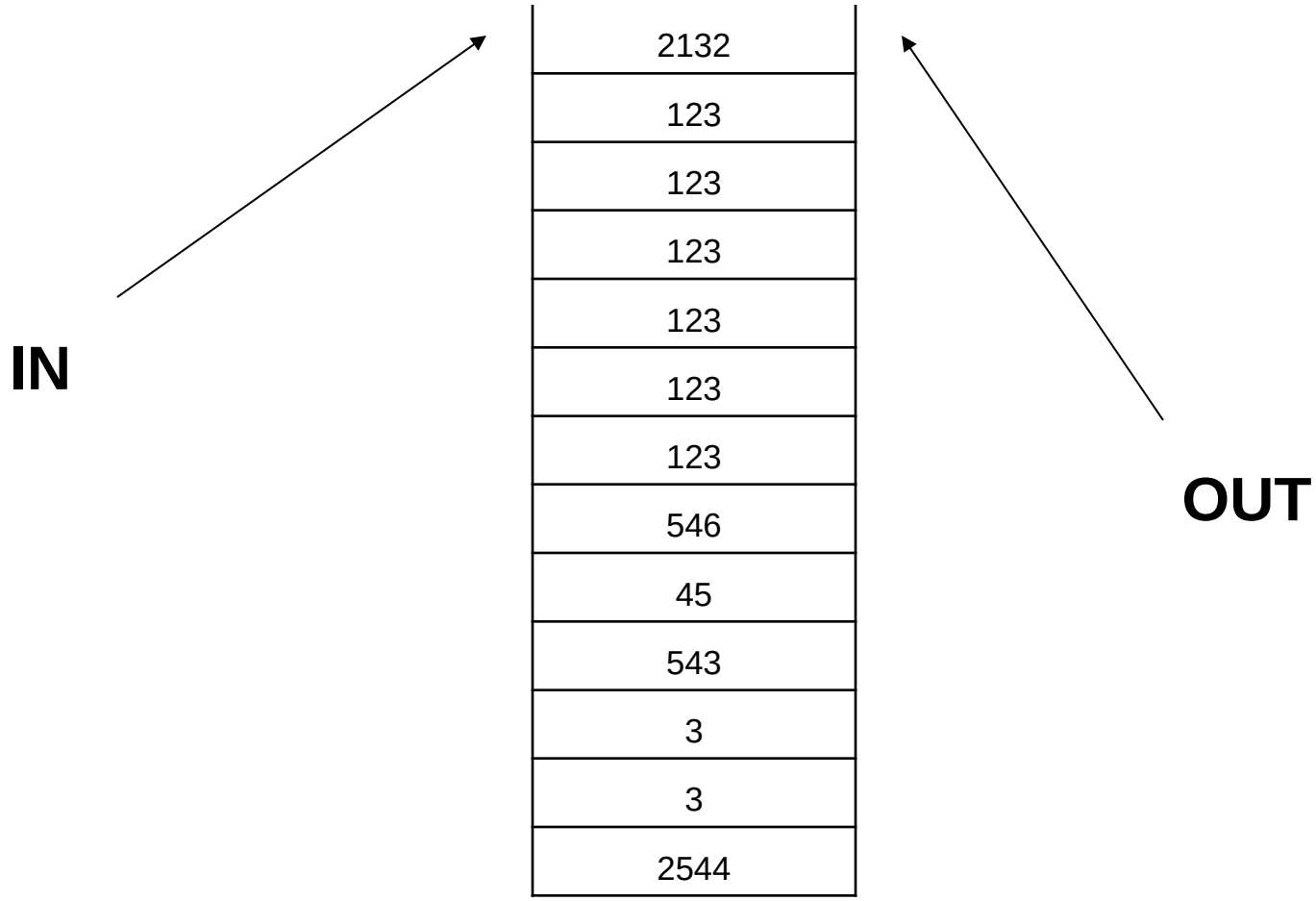

Employee
Name
Age
Address
Salary
First N
Last N
Street
Area
City
Province
Post Code

Trees

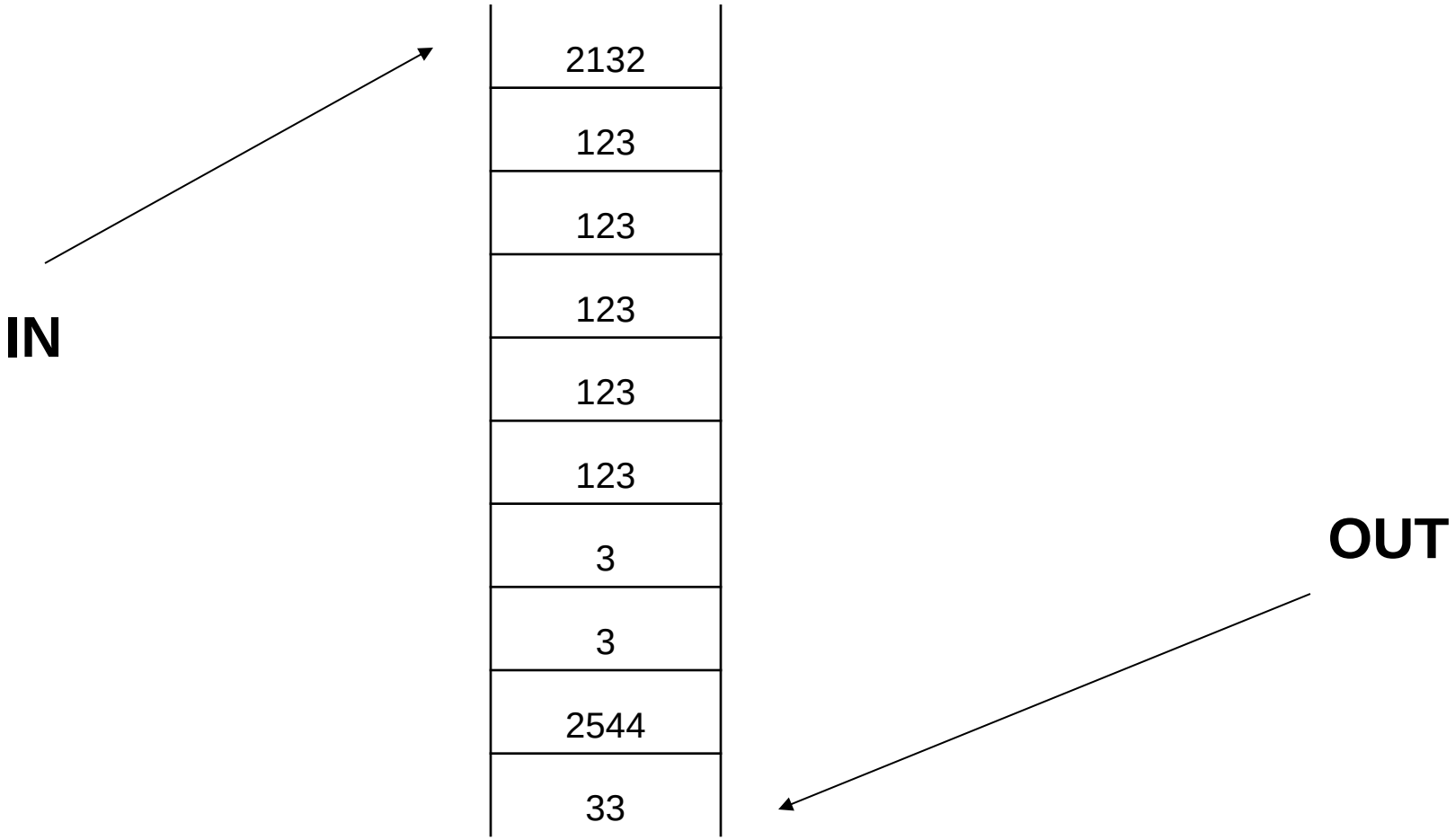
$(2x+y)(a-7b)^3$



Stacks



Queue



The Need for Data Structures

- Goal: to **organize data**

- Criteria: to facilitate **efficient**
  - **storage** of data
  - **retrieval** of data
  - **manipulation** of data

- Design Issue:
  - **select and design** appropriate data types.
    (This is the real essence of OOP.)

Data Structure Operations

- Update
- Searching
- Insertion
- Deletion

Dictionary Operations

- Sorting
- Merging

Thank you