Experiment 1

Name: Aaryan Sharma

Roll no. 16010123012

Q1. Advantages of Lists

Lists are a data structures with various advantages:

- Direct Access: Elements in a list can be accessed directly using their index.
- Dynamic Sizing: Lists can grow or shrink dynamically, which is useful for applications where the number of elements can change over time.
- Flexibility: Lists can store elements of the same type or different types providing versatility in data storage.
- Versatile Operations: Lists support a wide range of operations, including insertion, deletion, sorting and searching.
- Compatibility: Lists can be easily converted to other data structures.
- Efficient Iteration: Lists allow for efficient traversal and iteration over elements.
- Commonly used: Lists are supported by most programming languages,
- making them readily available.

Q3. ADT Functions for Lists

Abstract Data Types (ADTs) define the expected behaviour and operations of a data structure without specifying the implementation details.

- createList(): Initializes a new, empty list.
- insertAtPosition(list, position, element): Inserts an element at a specified position in the list.
- deleteAtPosition(list, position): Removes the element at a specified position in the list.
- getElementAtPosition(list, position): Returns the element at a specified position.
- IsEmpty(): Returns true if the list is empty, false otherwise.

These ADT functions provide a comprehensive set of operations for managing lists effectively in various programming contexts.

Q2. Comparison of ADT, Data Types, and Data Structures

Feature	ADT	Data Type	Data Structure
Definition	Abstract model of a data collection with defined operations	Classification of data based on its value and operations	Concrete implementation of an ADT for storing and organizing data
Focus	Behaviour and operations	Data representation and manipulation	Storage and organization
Implementation	No concern with implementation	Defined by programming languages	Concerned with specific implementation details
Use Case	Software design & modularity	Variable definition & type safety	Efficient data management
Relationship	ADT specifies the interface, data structure implements it	Data type defines the data, ADT defines how to use it	Data structure is a specific way to implement an ADT
Examples	List, Stack, Queue	Integer, Float, String, Boolean	Array, Linked List, Stack, Queue