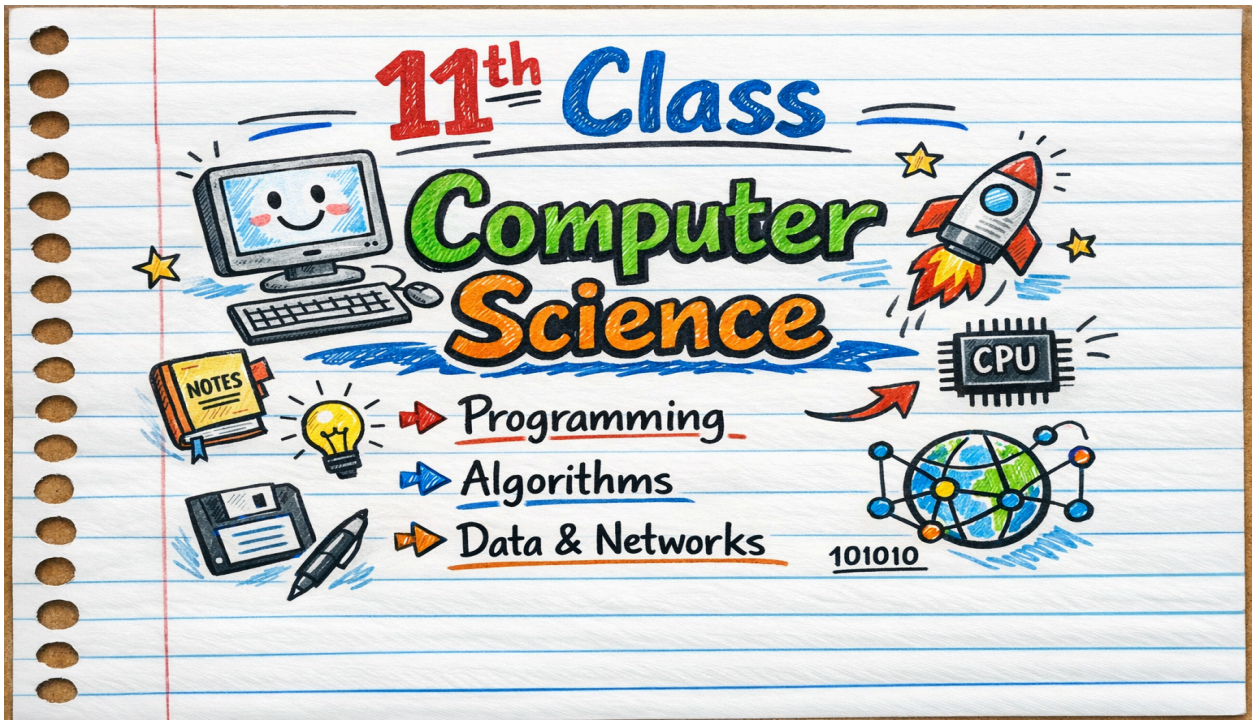

Class: 11th

Subject: Computer Science

Unit 4 Exercise: Computational Structures

📘 EXERCISE

Q1: Multiple Choice Questions

1. The function used to add an item at the end of a list in Python:

(a) insert()

(b) append() ✔

(c) remove()

(d) pop()

2. The purpose of the in keyword used with a Python list:

(a) Adds an item to the list

(b) Removes an item from the list

(c) Checks if an item exists in the list ✔

(d) Returns the length of the list

3. An operation that removes an item from the top of the stack:

(a) Push

(b) Pop ✔

(c) Peek

(d) Add

4. The operation used to add an item to a queue:

(a) Dequeue

(b) Peek

(c) Enqueue ✔

(d) Remove

5. True statement about the height of a tree:

StudyNotes360.com

(a) Number of edges from the root to the deepest node ✔

(b) Number of nodes from the root to the deepest node

(c) Number of children of the root node

(d) Always equal to the number of nodes in the tree

6. A scenario where a graph data structure is most suitable:

(a) Managing a to-do list

(b) Modeling a line of customers in a store

(c) Representing connections in a social network ✔

(d) All of the above

Q2: Short Questions

1. Explain how the insert() function works in Python lists. Provide an example.

The insert() function is used to add an element at a specific position (index) in a list. When a new element is inserted, the existing elements are shifted to the right to make space.

Example:

Python

```
items = ["A", "C", "D"]

items.insert(1, "B")
```

StudyNotes360.com

```
print(items)  # ['A', 'B', 'C', 'D']
```

2. Explain the potential issues when two variables reference the same list. Provide an example.

When two variables refer to the same list, they both point to the same memory location. Any change made using one variable will also affect the other. This can create unexpected results or bugs in a program.

Example:

Python

```
list1 = [1, 2, 3]

list2 = list1

list2.append(4)

print(list1)  # [1, 2, 3, 4]
```

3. Define a stack and explain the LIFO principle.

A stack is a linear data structure where insertion and deletion are performed at one end called the top. It follows the LIFO (Last In First Out) principle, meaning the last element added is the first one to be removed.

Example:

A stack of books where the last book placed on top is removed first.

StudyNotes360.com

4. Differentiate between Enqueue and Dequeue operations of a queue.

- **Enqueue** is an operation used to add an element at the rear (end) of the queue.
- **Dequeue** is an operation used to remove an element from the front of the queue.
- **Enqueue** increases the size of the queue, while dequeue decreases it.

Example:

Python

```
queue = ["A", "B"]

queue.append("C")   # enqueue

queue.pop(0)        # dequeue
```

5. Name two basic operations performed on stack.

The two basic operations are:

- Push ⟶ used to add an element to the top
- Pop ⟶ used to remove the top element

6. What is the difference between enqueue() and dequeue()?

Enqueue is used to insert an element at the rear of the queue, while dequeue is used to remove an element from the front.

StudyNotes360.com

These operations follow the FIFO principle.

Example:

Python

```
queue = []

queue.append("Ali")   # enqueue

queue.append("Ahmed")

queue.pop(0)        # dequeue
```

Q3: Long Questions

🌟 Q1. Discuss the dynamic size property of lists in Python. How does this property make lists more flexible?

The dynamic size property of Python lists means that a list can automatically grow or shrink during program execution. Unlike arrays in some programming languages, where the size must be fixed at the time of creation, Python lists do not require a predefined size.

This means we can easily add new elements or remove existing elements without worrying about memory allocation. Python internally manages memory for lists, making them very convenient and efficient to use.

Because of this property, lists are considered flexible data structures. They allow programmers to handle data that changes in size, such as

user input, records, or real-time data processing. This flexibility makes lists suitable for many applications like storing dynamic data, implementing stacks and queues, and managing collections of items.

Example:

Python

```
items = ["pen", "book"]

# Adding elements

items.append("eraser")

items.append("scale")

# Removing element

items.remove("book")

print(items)
```

Explanation:

Initially, the list contains two items. Then new items are added using append(), and one item is removed using remove(). The list automatically adjusts its size without any error or limitation.

Summary:

The dynamic size property makes Python lists highly flexible, easy to use, and suitable for handling changing data efficiently in real-world programs.

🌟 Q2. Explain the operations on stack with real-life example and Python code.

A stack is a linear data structure in which elements are added and removed from one end called the top. It follows the LIFO (Last In, First Out) principle, meaning the last element inserted is the first one to be removed.

Basic Operations of Stack

There are two main operations performed on a stack:

1. Push Operation

Push is used to add an element to the top of the stack. When a new element is pushed, it becomes the new top element.

2. Pop Operation

Pop is used to remove the top element from the stack. The last inserted element is removed first.

Real-Life Example

A stack of books or plates is a common example.

When you place books on top of each other, the last book you place is the first one you remove. You cannot remove a book from the middle; you must remove from the top only.

Python Code Example

Python

```
# Create an empty stack

stack = []

# Push operation (adding elements)

stack.append("Book A")

stack.append("Book B")

stack.append("Book C")

print("Stack after push:", stack)

# Pop operation (removing top element)

removed_item = stack.pop()

print("Removed item:", removed_item)

print("Stack after pop:", stack)
```

Explanation of Code

First, an empty list is created to represent a stack.

Then elements are added using append() (push operation).

Finally, the pop() function removes the top element (last added item), showing LIFO behavior.

Summary:

Stack operations (push and pop) are simple but very important. They are widely used in applications like undo operations, function calls, and expression evaluation. The LIFO principle makes stacks useful where the latest data needs to be accessed first.

🌟 Q3. Write a simple program to implement a queue (insertion and deletion).

A queue is a linear data structure that follows the FIFO (First In First Out) principle. In a queue, elements are inserted at the rear (end) and removed from the front (beginning).

In Python, a simple queue can be implemented using a list where:

- append() is used for insertion (enqueue)
- pop(0) is used for deletion (dequeue)

Python Program

Python

```
# Simple queue implementation using list

queue = []

# Insertion (Enqueue)

queue.append("Ahmed")
```

```
queue.append("Fatima")

queue.append("Sara")

print("Queue after insertion:", queue)

# Deletion (Dequeue)

removed_item = queue.pop(0)

print("Removed item:", removed_item)

print("Queue after deletion:", queue)
```

Explanation:

First, an empty list is created to act as a queue.

Then elements are added using append() which works as enqueue operation.

After that, pop(0) is used to remove the first element, which is the dequeue operation.

This follows the FIFO rule, where the first inserted element is removed first.

Output

```
Queue after insertion: ['Ahmed', 'Fatima', 'Sara']
```

Removed item: Ahmed

Queue after deletion: ['Fatima', 'Sara']

Summary:

This program shows a simple implementation of a queue using Python lists. It demonstrates both insertion (enqueue) and deletion (dequeue) while following the FIFO principle.

🌟 Q4. Define Tree and explain its properties (Proper Exam Answer)

Definition of Tree

A tree is a non-linear data structure used to organize data in a hierarchical form. It consists of nodes (data elements) connected by edges (links). A tree starts from a single node called the root node, and then branches into multiple levels of nodes.

It is called a hierarchical structure because it represents parent-child relationships between nodes.

Example:

A computer file system (folders and subfolders) and a family tree are common examples of tree structures.

✦ Properties of Trees

1. Root Node

The root node is the topmost node of a tree. It is the starting point of the tree and does not have any parent node. All other nodes originate from the root.

Example:

In a computer system, the main drive (like C drive) acts as the root.

2. Edges and Nodes

- Nodes are the individual elements in a tree that store data.
- Edges are the lines that connect nodes with each other.

A node that has no child is called a leaf node, similar to a file that does not contain any subfolder.

3. Height of Tree

The height of a tree is the longest path from the root node to the farthest leaf node. It tells how deep or tall the tree is.

Example:

If a tree has many levels of nodes, its height is greater.

4. Balanced Tree

A tree is called balanced if the left and right sides (subtrees) of each node have almost the same height. This helps in making operations like searching faster and more efficient.

Summary:

A tree is an important hierarchical data structure used to represent structured data. Its properties such as root node, nodes and edges, height, and balanced structure help in organizing and accessing data efficiently in applications like file systems, databases, and organizational charts.

🌟 Q5. What is a graph? Explain differences between directed and undirected graphs.

Definition of Graph

A graph is a non-linear data structure that consists of a set of vertices (nodes) and edges. The vertices represent objects or entities, and the edges represent the relationship or connection between these vertices.

Graphs are used to represent real-world networks such as social media connections, road maps, and computer networks.

Example:

In a social network, each person is a vertex, and friendship between them is represented by an edge.

Directed Graph (Digraph)

A directed graph is a graph in which edges have a specific direction. This means the relationship flows from one vertex to another in only one direction.

Example:

If A ⟶ B, it means A can go to B, but B cannot go back to A unless a reverse edge exists.

Real-life example:

One-way roads in traffic systems.

Undirected Graph

An undirected graph is a graph in which edges do not have direction. The connection between two vertices works both ways.

Example:

If A is connected to B, then B is also connected to A.

Real-life example:

Friendship in social networks (if A is friend of B, then B is also friend of A).

Differences between Directed and Undirected Graphs

In a directed graph, edges have a direction, meaning relationships are one-way. For example, in a Twitter follow system, A can follow B, but B may not follow A.

In an undirected graph, edges do not have direction, meaning relationships are two-way. For example, in Facebook friendship, if A is friends with B, then B is also friends with A.

Directed graphs are used where relationships are one-sided, while undirected graphs are used where relationships are mutual.

Summary:

A graph is an important data structure used to represent relationships between objects. Directed graphs show one-way relationships, while undirected graphs show two-way relationships, making them suitable for different real-world applications.

Note:

This chapter is designed to provide a solid foundation of knowledge, with the goal of deepening understanding and encouraging further exploration of the subject. The content has been carefully selected to support effective learning and inspire students to engage with the topic more deeply.

Author: Muhammad Asghar

Purpose: To contribute to education by offering insightful, valuable content that enhances learning and understanding.

Copyright & Usage Policy

© 2026 **Studynotes360.com**. All Rights Reserved.

No part of these notes may be reproduced, redistributed, or used for commercial purposes without explicit written permission from the author. These notes are intended solely for personal study and educational use.