11th Comp New Ch4 ShortQ.pdf

Primitive Computational Structures

Q.1 What are primitive computational structures?
Ans: Primitive computational structures are the basic building blocks of computer programs. They include data types, operations, and control structures like loops and conditionals. These elements help build complex systems by combining simple parts.

Q.2 Why are primitives important in computer science?
Ans: They form the foundation for all software and algorithms. Primitives allow abstraction and help developers design efficient systems. Understanding them makes learning programming and problem solving easier.

Lists

Q.3 What is a list in data structure?
Ans: A list stores multiple items in order, where each item has an index. It allows easy access, insertion, and deletion of data. Lists are used to manage collections of related items.

Q.4 How are lists created in Python?
Ans: In Python, lists are created using square brackets []. Items are separated by commas inside the brackets.
For example: fruits = ["apple", "banana", "cherry"].

Q.5 What is dynamic size in Python lists?
Ans: Dynamic size means a list can grow or shrink as needed. You can add or remove items without setting a fixed size first. This makes lists more flexible than arrays in other languages.

Q.6 What is index-based access in lists?
Ans: Index-based access lets you get or change an item using its position. The first item is at index 0, the second at index 1, and so on. This helps in quickly accessing specific data.

Q.7 What does it mean that lists are ordered?
Ans: Ordered means the sequence of items in a list is preserved. If you add "A" before "B", they will stay in that order unless changed. This helps maintain data relationships.

Q.8 How do you add an item to the end of a list?
Ans: Use the append() method to add an item to the end. For example: my_list.append("new item"). This changes the list permanently.

Q.9 How do you insert an item at a specific position in a list?
Ans: Use the insert() function with the index and value. Example: my_list.insert(1, "new item") inserts at position 1. Other items shift to make space.

Q.10 How do you remove an item by value from a list?

Ans: Use the remove() method. Example: my_list.remove("banana") removes the first occurrence of "banana". It gives an error if the item is not found.

Q.11 How do you remove an item by index from a list?
Ans: Use the pop() method. Example: my_list.pop(0) removes the item at index 0. If no index is given, it removes the last item.

Q.12 How do you check if an item exists in a list?
Ans: Use the in keyword. Example: if "apple" in my_list: checks if "apple" is present. It returns True or False.

Q.13 What are two applications of lists?
Ans: Lists store and manage data like names, numbers, or tasks. They also help implement stacks and queues, which are used in many computing tasks.

Stack

Q.14 What is a stack?
Ans: A stack is a data structure where items are added and removed from the top only. It follows the **LIFO (Last-In, First-Out)** principle. Like stacking plates, the last one added is the first taken off.

Q.15 What is the LIFO principle?
Ans: LIFO stands for **Last-In, First-Out**. In a stack, the last item added is the first one removed. This rule ensures a specific order of processing.

Q.16 What are the two main operations of a stack?
Ans: The main operations are:

- **Push:** Adds an item to the top.
- **Pop:** Removes an item from the top.

These follow the LIFO rule.

Q.17 Give a real-life example of a stack.
Ans: A browser's back button uses a stack to keep track of visited pages. When you click back, the most recent page is removed from the stack. This helps navigate in reverse order.

Queue

Q.18 What is a queue?
Ans: A queue is a data structure where items are added at the back and removed from the front. It follows the **FIFO (First-In, First-Out)** principle. Like a line at a ticket counter.

Q.19 What is the FIFO principle?
Ans: FIFO stands for **First-In, First-Out**. The first item added is the first one removed. It ensures fair processing order in queues.

Q.20 What are the two main operations of a queue?
Ans: The main operations are:

- **Enqueue:** Add an item to the back.
- **Dequeue:** Remove an item from the front.

These follow the FIFO rule.

Q.21 Give a real-life example of a queue.
Ans: Print jobs in a printer use a queue. The first document sent is printed first, and others wait in order. This prevents confusion and keeps processing fair.

Trees

Q.22 What is a tree data structure?
Ans: A tree organizes data hierarchically starting from a root node. Each node can have child nodes, forming a branching structure. It shows parent-child relationships clearly.

Q.23 What is a root node in a tree?
Ans: The root node is the topmost node in a tree. All other nodes branch out from it. It is like the CEO in an organizational chart.

Q.24 What is a leaf node in a tree?
Ans: A leaf node is a node with no children. It appears at the ends of branches. Like a file in a folder that does not contain any subfolders.

Q.25 What is the height of a tree?
Ans: Height is the number of levels in a tree. It measures how deep the tree goes from the root to the farthest leaf. A taller tree takes longer to search.

Q.26 What is a balanced tree?
Ans: A balanced tree has nearly equal heights on both sides. It helps keep searching and updating fast. An unbalanced tree may slow down operations.

Q.27 Name two applications of trees.
Ans: Trees are used to represent file systems and family trees. They also help in decision-making processes like decision trees in AI.

Introduction to Graphs

Q.28 What is a graph in data structure?
Ans: A graph consists of vertices (nodes) connected by edges. It represents relationships between objects. Graphs can model networks like roads, social connections, or web links.

Q.29 How is a tree different from a graph?
Ans: A tree is hierarchical with a root and no cycles. A graph has no root and can have cycles. Trees are a special type of graph.

Q.30 What are directed and undirected graphs?
Ans: In a directed graph, edges have direction (like one-way streets). In an undirected graph, edges go both ways. Directed graphs model Twitter follows; undirected graphs model Facebook friendships.