

Computational Structures

Q. What is a list? Describe its properties and applications.

List

A **list** is a data structure in Python that can store multiple values. The list is **mutable** that means its elements can be created, accessed, modified or removed easily. Each item stored in a list is called an **element**. Lists can store elements of different data types, including integers, floats, strings etc. Each element in the list can be accessed with reference to its position in the list. This position is called **index**. Each element in the list has a unique index. The index of first element is 0 and the index of last element is length -1. The value of the index is written in brackets along with the name of the list.

The following example shows a list **Num** with three elements:



Figure: A list **Num** with three elements

List Properties

List has the following properties:

1. Dynamic Size

A list in Python can change its size. The new items can be added to the list and existing items can be removed easily. The size of the list automatically changes.

2. Index-Based Access

Every item in a list has a unique index that indicates its position in the list. The index is used to access specific items in the list easily.

3. Ordered Collection

The order in which items are added to a list is preserved. It means that the first item added to the list will stay in that position unless it is changed.

Applications of Lists

Lists are one of the most commonly used data structures in Python due to their flexibility and ease of use. Some important applications of lists are described below:

1. Data Storage and Manipulation

Lists are commonly used to store and manage collections of data like records, entries or values. The data in the lists can be inserted, deleted and accessed easily.

2. Stack and Queue Implementations

Lists can be used to implement the data structures known as stack and queue.

Stack

It is a data structure that follows the **Last In, First Out (LIFO)** principle. This means the last element added to stack is the first one to be removed.

Queue

It is a data structure that follows the **First In, First Out (FIFO)** principle. This means the first element added to queue is the first one to be removed.

Q. Describe some common operations of a list with examples.

Some common operations of a list include insertion, deletion and searching.

1. Insertion

The insertion operation is used to add a new item to a list. An item can be inserted at different positions in the list. The **insert()** method in Python is used to insert an item at a specific index in a list.

Syntax

The syntax of **insert()** method is as follows:

```
list_name.insert(index, element);
```

list_name

It indicates the name of the list.

index

It indicates the index where the element will be inserted in the list.

Example

```
country = ['Turkey', 'Iran', 'Oman']  
country.insert(0, 'Pakistan')  
print(country)
```

Output: ['Pakistan', 'Turkey', 'Iran', 'Oman']

2. Deletion

The deletion operation is used to remove an item from a list. The deletion of an item can be done using the value or index of the item.

a. Removing by Value

The **remove()** method is used to delete an item from the list by value. It removes the first occurrence of the specified value from the list.

Syntax

The syntax of **remove()** method is as follows:

```
list_name.remove(val)
```

list_name

It indicates the name of the list.

val

It indicates the value of the list element to be removed.

Example

```
fruit = ['apple', 'banana', 'cherry', 'potato', 'kiwi']  
fruit.remove('potato')  
print(fruit)
```

Output: ['apple', 'banana', 'cherry', 'kiwi']

b. Removing by Index

The **pop()** method is used to delete an item from the list by index. The function deletes the last item from the list if the index is not given.

Syntax

The syntax of **pop()** method is as follows:

`list_name.pop(ind)`

list_name It indicates the name of the list.

ind It indicates the index of the value to be removed. It is optional.

Example

```
even = [2, 4, 6, 7, 8, 10]
even.pop(3)
print(even)
```

Output: [2, 4, 6, 8, 10]

3. Searching

The searching operation is used to find an item in a list. The **in** operator can be used to check whether a specific value exists in the list. It returns **True** if the value exists and returns **False** if the value does not exist. It is case-sensitive and only checks for exact matches.

Syntax

The syntax of **in** operator is as follows:

`value in list_name`

value It is the item to be searched in the list.

list_name It indicates the name of the list.

Example

```
fruits = ["apple", "banana", "cherry"]
if("mango" in fruits):
    print("Mango is available")
else:
    print("Sorry, mango is not available")
```

Output: Sorry, mango is not available

PROGRAM 4.1 Write a program that inputs a number from the user and searches the number in a list.

```
mylist = [10, 20, 30, 40, 50]
n = int(input("Enter a number of search: "))
if(n not in mylist):
    print(n, "is not in the list.")
else:
    s = mylist.index(n)
    print("The number ", n, "is at index", s)
```

Output:

Enter a number to search: 30
The number 30 is at index 2

Q. What is a stack? Discuss its basic operations.

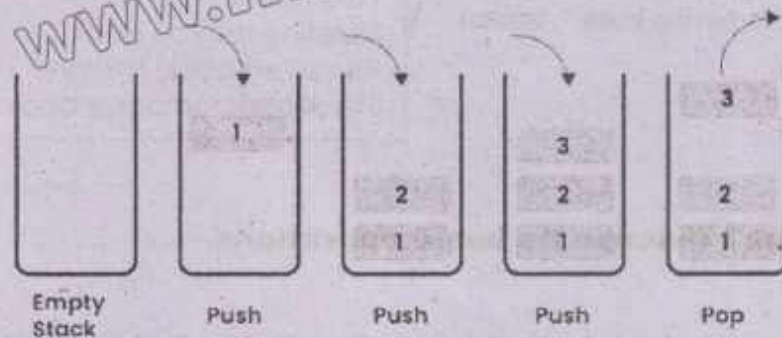
Stack

A **stack** is a data structure that follows the **Last In, First Out (LIFO)** principle. This means that the last element added to stack is the first one to be removed. It can be considered as a container where both insertion and deletion of elements occur at the top end.

Stack Operations

Two basic operations in a stack are as follows:

- **Push:** This operation is used to add an item to the top of the stack.
- **Pop:** This operation is used to remove an item from the top of the stack.



Q. Explain the operations on stack with real life example and python code.

Stack in Python

A **stack** can be created in Python using a list. The basic stack operations can be implemented using **append()** and **pop()** method.

The Push Operation

The push operation of stack can be implemented in Python using **append()** method. It adds an item to the top of the stack.

Syntax

The syntax of this method is as follows:

`list_name.append(val)`

list name It indicates the name of the list.

val It indicates the value that will be appended to the list.

The Pop Operation

The pop operation of stack can be implemented in Python using **pop()** method. It deletes the item from the top of the stack.

Syntax

The syntax of this method is as follows:

`list_name.pop()`

list name It indicates the name of the list.

Real-life Example

A simple real-world example of a stack is a pile of books. The last book placed on the top of the pile is the first one to be removed. The books at the bottom cannot be taken without first removing the books on the top. This approach is called **Last In First Out (LIFO)**.

Python Code for Book Stack

```
books = []
print("Initial stack of books:", books)
print("Adding books to stack...")
books.append("Computer")
```

```

print("Stack after adding one book: ", books)
books.append("Python")
print("Stack after adding two books: ", books)
print("Deleting top book... ")
top = books.pop()
print("Removed book:", top)
print("Stack after removing book:", books)

```

Output:

```

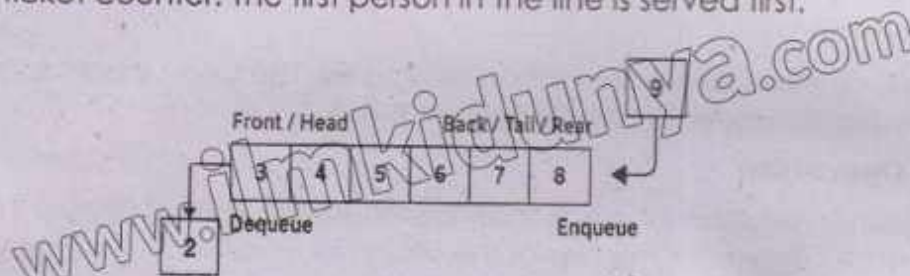
Initial stack of books: []
Adding books to stack...
Stack after adding one book: ['Computer']
Stack after adding two books:
['Computer', 'Python']
Deleting the top book...
Removed book: Python
Stack after removing book: ['Computer']

```

Q. What is a queue? Discuss its basic operations.

Queue

A **queue** is a data structure that follows the **First In First Out (FIFO)** principle. This means that it stores new item at the end of all previously entered items. Similarly, the item that is stored at first position is removed first. It can be considered as a line in front of a bank or a ticket counter. The first person in the line is served first.



Queue Operations

Two basic operations in a queue are as follows:

- **Enqueue:** This operation is used to add an item to the end of the queue.
- **Dequeue:** This operation is used to remove an item from the front of the queue.

Some other operations include the following:

- Checking if the queue is empty or full.
- Retrieving the element at the front of the queue without removing it.
- Determining the size of the queue.

Q. How can a queue be created in Python? Give an example.

Queue in Python

A **queue** can be created in Python using built-in **Queue** class of **queue** module. It provides various methods to create a queue and perform different operations on it. The **Queue** class must be imported in Python program in order to create a queue.

The following line of imports **Queue** class from Python's built-in **queue** module:

```
from queue import Queue
```

Methods of queue Module

Some important methods in queue module are as follows:

1. The Queue() Method

The **Queue()** method creates an object of **Queue()** class. The syntax of this function is as follows:

obj.Queue();

obj It indicates the name of the object to be created.

2. The put() Method

The **put()** method adds an item to the queue. The syntax of this method is as follows:

obj.put(item);

obj It indicates the name of the Queue object.

item It indicates the item to be added to the queue.

3. The get() Method

The **get()** method retrieves an item from the queue. The syntax of this method is as follows:

obj.get(item);

obj It indicates the name of the Queue object.

item It indicates the item to be added to the queue.

Example

```
from queue import Queue
q = Queue()
q.put("Ali")
q.put("Usman")
q.put("Furqan")
front = q.queue[0]
print(front, "is at the front of the queue.")
removed = q.get()
print(removed, "is removed from the queue.")
q.put("Sara")
updated = list(q.queue)
print(updated)
```

Output:

```
Ali is at the front of the queue.
Ali is removed from the queue.
['Usman', 'Furqan', 'Sara']
```

Explanation

from queue import Queue

This line imports **Queue** class.

q = Queue()

This line creates a new empty queue object **q**.

q.put("Ali")

q.put("Usman")

q.put("Furqan")

These lines add three people to the queue.

front = q.queue[0]

This line accesses the first item in the queue ("Ali") without removing it.

print(front, "is at the front of the queue.")

This line displays the first item in the queue ("Ali").

removed = q.get()

This line removes the first item in the queue ("Ali") and stores it in variable **removed**.

print(removed, "is removed from the queue.")

This line displays the value of **removed** variable.

q.put("Sara")

This line adds "Sara" to the end of the queue.
`updated = list(q.queue)`

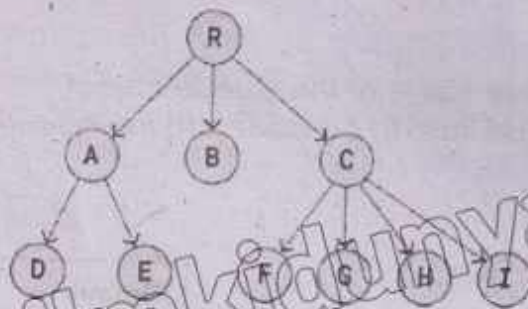
This line stores the whole queue in `updated` variable.
`print(updated)`

This line adds `updated` variable.

Q. What is a tree? Explain with an example.

Tree

A **tree** is a data structure that organizes data in a hierarchical way. It consists of nodes and edges where each node stores data and is connected to other nodes by edges. It forms a branching structure like a tree.



Example

A tree is similar to a family tree where the oldest ancestor represents root node. It is the starting point of the hierarchy. Each individual in a tree may have descendants that form subsequent levels of the hierarchy. It is a non-linear structure that represents parent-child relationships.



Figure: Family tree

Q. Describe the properties and applications of trees.

Properties of Trees

Some important properties of trees are as follows:

1. Root Node

The **root** is the top most node in a tree. It acts as the entry point to the entire structure. It is similar to the main folder in a computer where all other folders and files are contained.

2. Edges and Nodes

Nodes are the individual elements in the tree. The nodes are connected by lines called **edges**. A node without any child nodes is called a **leaf** similar to a file in a folder that does not contain any other files.

3. Height

The height of a tree is the longest path from the root node down to the deepest leaf node. It indicates the height or depth of the tree.

4. Balanced Trees

A tree is considered balanced if the branches on the left and right sides are nearly the same height.

Applications of Trees

Some important applications of trees are as follows:

1. File Systems

Pre-order tree traversal is useful for creating backups of file systems. It is a method of traversing a tree where each node is visited before its children. It ensures that directories are backed up before their contents.

2. File System Deletion

Post-order traversal is a tree traversal method where the child nodes are visited before the parent node. It is used in file systems to ensure that files and directories are deleted in the correct order. It first deletes all sub directories and files before deleting the parent directory.

3. Hierarchical Data Representation

Trees are used in representing data with a clear hierarchical relationship such as organizational charts and family trees.

4. Decision Making

Trees such as decision trees are used in algorithms to make decisions based on various conditions and outcomes.

Q. What is a graph? Explain with an examples.

Graph

A **graph** is a data structure that consists of a set of **vertices** or **nodes** connected by **edges**. Graphs are used to represent networks of connections where each connection is a relationship between two vertices. These vertices can represent real-world entities such as cities, people or concepts. The edges represent the relationships, connections or pathways between them.

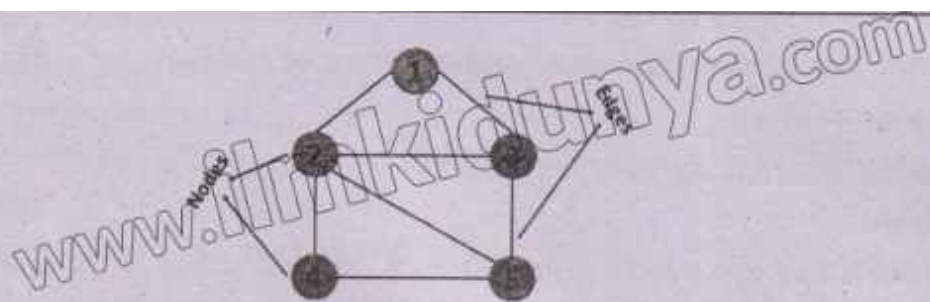


Figure: Graph

A graph does not have a single root node and it does not follow a hierarchical structure. It can have multiple connections between nodes.

Example 1

Suppose the user is mapping out all cities in Pakistan and the roads that connect them. Each city is a vertex and each road between two cities is an edge. This forms a graph because cities can be connected in multiple ways and there is no single starting or ending point.

Example 2

In a social network, each user is a node and a friendship or connection between users is an edge. A user can be connected to many other users that forms a graph. There is no single starting or ending point.

Q. Describe different characteristics and properties of graphs.

Characteristics of Graphs

Some important characteristics of graphs are as follows:

1. Vertices (Nodes)

These are the individual points or entities in a graph. For example, each user in a social network graph is represented by a vertex.

2. Edges (Links)

These are the connections between vertices. For example, each road in a transport system graph is an edge that connects two cities.

Properties of Graphs

Some important properties of graphs are as follows:

1. Degree

This is the number of edges connected to a vertex. For example, the degree of vertex will be 3 if a city is connected to three other cities.

2. Weight

In some graphs, edges have weights that represent the values like distances or costs. For example, if a road between two cities is 50 kilometers long, its edge may have a weight of 50.

3. Direction

Edges can be either directed or undirected. Directed edges have a one-way connection such as a one-way street. The undirected edges represent a two-way connection.

Q. Describe different types of graphs.

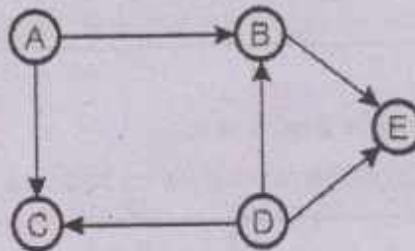
Graphs can be classified into many types based on their structure and properties. The main types of graphs are directed, undirected and weighted.

1. Directed Graphs

A **directed graph** is a type of graph where each edge has a direction. It means that it goes from one vertex to another vertex in a specific way.

Example

The following graph is a directed graph. The arrows show the direction of link between two vertices.

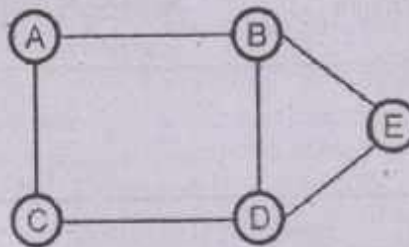


2. Undirected Graphs

An **undirected graph** is a type of graph in which the edges have no direction. This means that if there is a connection between two vertices, you can travel in both directions.

Example

The following graph is an undirected graph. Suppose the user needs to go from A to D. There is no restriction on the direction.

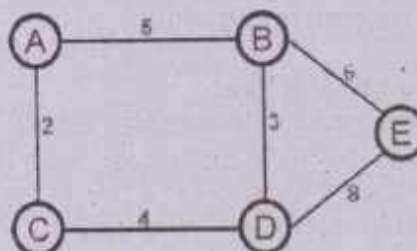


3. Weighted Graphs

A **weighted graph** is a type of graph in which each edge has an associated numerical value called a **weight** or **cost**. It represents the distance, time or cost required to travel from one vertex to another.

Example

The following example shows a graph for the map of a city. Each road on the map has a different distance or travel time. This information helps the user to determine the shortest or quickest route between two points.



Q. Differentiate between directed and undirected graph.

The difference between directed and undirected graph is as follows:

Directed Graph	Undirected Graph
1. The edges in directed graph have directions from one vertex to another.	1. The edges in undirected graph have no directions.
2. It allows one-way movement from one node to another.	2. It allows two-way movement from one node to another.
3. It is less flexible than undirected graph.	3. It is more flexible than directed graph.
4. The vertices are connected using arrows in directed graph.	4. The vertices are connected using lines in undirected graph.

Q. Differentiate between graph and tree.

The difference between graph and tree is as follows:

Tree	Graph
1. A tree is a hierarchical structure.	1. A graph is a network-like structure.
2. It has a single root node.	2. It does not have any root node.
3. It has exactly one path between any two nodes.	3. It can have multiple paths between nodes.
4. It is usually directed from parent to child nodes.	4. It can be directed or undirected.
5. It cannot have cycles or loops.	5. It can have cycles or loops.
6. It is used to represent file systems, organizational charts and family trees.	6. It is used to represent social networks, maps, web links and transport systems.

Exercise Solution

Multiple-Choice Questions (MCQs)

- The function used to add an item at the end of a list in Python:
a) insert() b) append() c) remove() d) pop()
- 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
- An operation that removes an item from the top of the stack:
a) Push b) Pop c) Peek d) Add
- The operation used to add an item to a queue:
a) Dequeue b) Peek c) Enqueue d) Remove
- True statement about the height of a tree:
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

Answers

1. b	2. c	3. b	4. c	5. a	6. c
------	------	------	------	------	------

Short Questions

Q.1. Explain how the insert function works in python list. Provide an example.

The insert() function in Python is used to add an item at a specific index in a list. The syntax of this function is list_name.insert(index, element). An example of insert() function is as follows:

```
country = ["Turkey", "Iran", "Oman"]
country.insert(0, "Pakistan")
print(country)
```

Q.2. Explain the potential issues which could arise when two variables reference the same list in a program? Provide an example.

When two variables reference the same list, they both point to the same memory location. This means that changes made through one variable will affect the other. It will often occur unintentionally.

```
a = [1, 2, 3]
b = a
b.append(4)
print("a:", a) # Output: a: [1, 2, 3, 4]
print("b:", b) # Output: b: [1, 2, 3, 4]
```

Q.3. Define a stack and explain the Last In, First Out (LIFO) principle.

Stack is a data structure that follows the **Last In, First Out (LIFO)** principle. This means the last element added to stack is the first one to be removed.

Q.4. Differentiate between the Enqueue and Dequeue operations of queue.

Enqueue and Dequeue are the two main operations of a queue. Enqueue adds an element to the rear of the queue while Dequeue removes an element from the front. Enqueue increases the queue size whereas Dequeue decreases it.

Q.5. Name two basic operations performed on stack.

Two basic operations performed on stack are push and pop. Push operation is used to add an item to the top of the stack. Pop operation is used to remove an item from the top of the stack.

Long Questions

Q.1. Discuss the dynamic size property of lists in Python. How does this property make lists more flexible?

The lists in Python have a dynamic size. It means that they can grow or shrink automatically as elements are added or removed. The user does not need to declare a fixed size when creating a list. The dynamic size property makes the lists highly flexible for the following reasons:

1. **No Predefined Size:** The lists can be defined without any predefined size. It gives flexibility to add any number of elements to the list.
2. **Automatic Memory Management:** Python internally resizes the list's storage as needed. It handles memory allocation for the user.
3. **Ease of Use:** Lists support operations like append(), insert(), pop() and remove(). These operations allow dynamic manipulation of content at runtime.

4. **Suitable for Varying Data:** It is ideal for situations where the number of items is not known in advance such as reading user input.

Q.2. Explain the operations on stack with real life example and python code. (See chapter)

Q.3. Write, a simple program to implement a queue (insertion and deletion). (See chapter)

Q.4. Define Tree and explain its properties. (See chapter)

Q.5. What is graph? Explain differences between directed and undirected graph.
(See chapter)

SHORT QUESTIONS

Q.1. What are the properties of a list in Python?

The properties of a list in Python include dynamic size, index-based access and ordered collection. Dynamic size means that a list can change its size. Index-based access means that every item in a list has a unique index that indicates its position in the list. Ordered collection means that the order in which items are added to a list is preserved.

Q.2. What does dynamic size mean in the context of lists?

It means that a list in Python can change its size. The new items can be added to the list and existing items can be removed easily. The size of the list automatically changes.

Q.3. List some common operations that you can perform on lists in Python.

Some common operations that you can perform on lists in Python include insertion, deletion, searching and sorting.

Q.4. How do you remove an item from a list in Python?

You can remove an item from a list in Python using `remove()` or `pop()` function. The `remove()` function deletes an item from the list by value. It removes the first occurrence of the specified value from the list. The `pop()` function is used to delete an item from the list by index. The function deletes the last item from the list if the index is not given.

Q.5. How can you remove an item from a list by value?

You can remove an item from a list by value using `remove()` function. It removes the first occurrence of the specified value from the list. For example, `fruit.remove("potato")` removes "potato" from fruit list.

Q.6. How can you remove an item from a list by index?

You can remove an item from a list by index using `pop()` function. It deletes the value at the specified index of the list. The function deletes the last item from the list if the index is not given. For example, `even.pop(3)` removes the item at index 3 from even list.

Q.7. How do you search for an item in a list in Python?

The `in` operator is used to search a value in the list. It returns `True` if the value exists and returns `False` if the value does not exist.

Q.8. What is a stack in data structures?

A stack is a data structure that follows the Last In, First Out (LIFO) principle. This means that the last element added to stack is the first one to be removed. It can be considered as a container where both insertion and deletion of elements occur at the top end.

Q.9. Name two basic operations in a stack.

Two basic operations in a stack are push and pop. The push operation is used to add an item to the top of the stack. The pop operation is used to remove the top item.

Q.10. How do you implement a stack using a list in Python?

You can implement a stack using a list in Python by using the `append()` method to add items to the stack and the `pop()` method to remove items from the stack.

Q.11. Can you access items in the middle of a stack?

No, you cannot access items in the middle of a stack directly. You can only access the top item of the stack.

Q.12. What is a queue in data structures?

A queue is a data structure that follows the First In First Out (FIFO) principle. This means that it stores new item at the end of all previously entered items. Similarly, the item that is stored at first position is removed first. It can be considered as a line in front of a bank or a ticket counter. The first person in the line is served first.

Q.13. Write two basic operations in a queue.

Two basic operations in a queue are enqueue and dequeue. The enqueue operation is used to add an item to the end of the queue. The dequeue operation is used to remove an item from the front of the queue.

Q.14. How do you implement a queue using the queue module in Python?

You can implement a queue using queue module in Python by creating a Queue object and using its put() and get() methods to add and remove items.

Q.15. How do you add items to a queue in Python?

You add items to a queue in Python using put() method. For example, q.put("Ahmed") will add "Ahmed" to the queue named q.

Q.16. How do you remove items from a queue in Python?

You remove an item from a queue in Python using get() method. For example, q.get() removes the first added item from the queue named q.

Q.17. What is the difference between a stack and a queue?

The stack data structure follows Last In, First Out (LIFO) principle. This means that the last element added to stack is the first one to be removed. The queue follows the First In First Out (FIFO) principle. This means that the first element added to queue is the first one to be removed.

Q.18. What is a tree in data structure?

A tree is a data structure that organizes data in a hierarchical way. It consists of nodes and edges where each node stores data and is connected to other nodes by edges. It forms a branching structure like a tree.

Q.19. What is the root node in a tree?

The root is the top most node in a tree. It acts as the entry point to the entire structure. It is similar to the main folder in a computer where all other folders and files are contained.

Q.20. What are nodes and edges in a tree?

Nodes are the individual elements in the tree. The nodes are connected by lines called edges. A node without any child nodes is called a leaf similar to a file in a folder that does not contain any other files.

Q.21. How do you determine the height of a tree?

The height of a tree is determined by finding the longest path from the root node down to the deepest leaf node.

Q.22. What is a balanced tree?

A tree is a balanced if the branches on the left and right sides are nearly the same height.

Q.23. What is pre-order tree traversal?

Pre-order tree traversal is useful for creating backups of file systems. It is a method of traversing a tree where each node is visited before its children. It ensures that directories are backed up before their contents.

Q.24. What is post-order tree traversal?

Post-order traversal is a tree traversal method where the child nodes are visited before the parent node. It is used in file systems to ensure that files and directories are deleted in the correct order. It first deletes all sub directories and files before deleting the parent directory.

Q.25. What is a graph?

A graph is a data structure that consists of a set of vertices or nodes connected by edges. Graphs are used to represent networks of connections where each connection is a relationship between two vertices.

Q.26. Give a real-world example of a graph.

Suppose the user is mapping out all cities in Pakistan and the roads that connect them. Each city is a vertex and each road between two cities is an edge. This forms a graph because cities can be connected in multiple ways and there is no single starting point.

Q.27. Can graphs have multiple paths between two vertices?

Yes, graphs allow multiple paths between two vertices. It is useful for modeling transport routes or communication channels.

Q.28. What is meant by the degree of a vertex?

The degree of a vertex is the number of edges connected to it. For example, the degree of vertex will be 3 if a city is connected to three other cities.

Q.29. What is the difference between directed and undirected edges?

Directed edges have a one-way connection that means the relationship or pathway is only in one direction. Undirected edges represent a two-way connection.

Q.30. How do graphs differ from trees?

Tree is a hierarchical structure whereas graph is a network-like structure. Tree has a single root node but graph does not have any root node. Tree has exactly one path between any two nodes but graph can have multiple paths between nodes.

Multiple Choice Questions

1. A data structure used to store multiple items in a sequence is called:
a. Dictionary b. Loop c. List d. Function
2. Each piece of data in a list is known as a:
a. Variable b. Element c. Operator d. Function
3. Lists are created using:
a. Curly braces {} b. Parentheses ()
c. Angle brackets <> d. Square brackets []
4. How are elements separated in a Python list?
a. Semicolons b. Commas c. Periods d. Colons
5. Every item in a list has a position, called an _____.
a. Index b. Value c. Label d. Node
6. What is the index of the first element in a list?
a. 0 b. 1 c. -1 d. First
7. Which of the following are properties of a list in Python?
a. Dynamic Size b. Index-Based Access
c. Ordered Collection d. All
8. Which property allows Python lists to increase or decrease in size during execution?
a. Static Size b. Indexed Storage
c. Dynamic Size d. Constant Size
9. In Python, what type of collection is a list?
a. Random b. Unordered c. Ordered d. None
10. Which method is used to insert an element at a specific position in a Python list?
a. add() b. insert() c. append() d. put()
11. Which of the following can be used to insert 5 into the third position (Index 2) in list1?
a. list1.insert(3, 5) b. list1.insert(2, 5)
c. list1.add(3, 5) d. list1.append(3, 5)
12. Which method removes an item from the list by value?
a. pop() b. delete() c. remove() d. discard()
13. Which of the following removes "Buy snacks" from the party_list?
a. party_list.delete("Buy snacks") b. party_list.remove("Buy snacks")
c. party_list.pop("Buy snacks") d. party_list.clear("Buy snacks")

14. Which method is used to remove an item by its index?
a. remove() b. pop() c. delete() d. clear()
15. Which of the following removes the first occurrence of an item from a list?
a. append() b. pop() c. remove() d. clear()
16. Which operator is used to check if an item exists in a Python list?
a. find b. in c. has d. exist
17. Which types of elements can a Python list contain?
a. Only integers b. Only strings
c. Only characters d. Any type of data
18. Which data structure can be implemented using a Python list to follow Last In First Out (LIFO) order?
a. Queue b. Dictionary c. Stack d. Tuple
19. Which data structure can be implemented using a list in First In First Out (FIFO) order?
a. Set b. Queue c. Stack d. Matrix
20. In a stack data structure, from which end are elements added and removed?
a. Bottom b. Middle c. Top d. Random
21. Which operation adds an item to the top of the stack?
a. insert() b. pop() c. push d. delete()
22. Which operation removes an item from the top of the stack?
a. append() b. remove() c. push d. pop
23. Which Python list method is used to implement the "push" operation in a stack?
a. insert() b. append() c. pop() d. extend()
24. Which Python list method is used to implement the "pop" operation in a stack?
a. delete() b. insert() c. pop() d. remove()
25. What does the following code do? stack = []
a. Creates a queue b. Creates an array
c. Creates an empty stack d. Deletes a list
26. A line of customers at a bank is an example of which data structure?
a. Stack b. Queue c. Tree d. Array
27. In a queue data structure, new elements are always added to the:
a. Front b. Middle c. Rear d. Any
28. Which operation adds an item to the end (rear) of a queue?
a. dequeue b. push c. enqueue d. pop
29. The operation to remove an element from the front of a queue is called:
a. pop b. Dequeue c. remove d. delete
30. Which Python module is commonly used to implement a queue?
a. stack b. collections c. queue d. list
31. Which method is used to add (enqueue) an item to a queue in Python?
a. add() b. push() c. put() d. append()
32. Which method is used to remove an item from a queue in Python?
a. remove() b. get() c. delete() d. pop()
33. A tree is a _____ data structure.
a. Linear b. Hierarchical c. Sequential d. Random
34. The topmost node in a tree is called:
a. Leaf node b. Parent node
c. Root node d. Branch node
35. The individual elements in a tree are called:
a. Leaves b. Edges c. Nodes d. Roots
36. The lines that directly connect nodes in a tree are called:
a. Path b. Edge c. Link d. Branch
37. In tree data structures, nodes without any children are called:
a. Root nodes b. Internal nodes
c. Leaf nodes d. Edge nodes
38. The number of edges from root node to the deepest leaf is called the _____ of tree.
a. Height b. Depth c. Distance d. Width

39. Which tree traversal is most suitable for creating file system backups?
a. In-order
b. Pre-order
c. Post-order
d. Level-order
40. The correct traversal for safely deleting a directory and its contents is:
a. In-order
b. Pre-order
c. Post-order
d. Breadth-first
41. Which data structure uses tree-based branching to evaluate conditions and make decisions?
a. Stack
b. Queue
c. Decision trees
d. Circular list
42. A _____ is a data structure that consists of a set of vertices (or nodes) connected by edges.
a. Graph
b. Tree
c. Array
d. Stack
43. Which of the following is a data structure that consists of a set of vertices (or nodes) connected by edges?
a. Tree
b. Array
c. Graph
d. Stack
44. Another name for a vertex in a graph is called:
a. Leaf
b. Node
c. Root
d. Point
45. What are the individual points or entities in a graph called?
a. Edge
b. Path
c. Vertices
d. Degrees
46. A connection or relationship between two vertices in a graph is called a:
a. Node
b. Edge
c. Root
d. Path
47. Which structure is best for modeling a network of friends connected with each others?
a. Stack
b. Tree
c. Graph
d. Queue
48. Which of the following is NOT typically represented using a general graph structure?
a. Social networks
b. Web page links
c. Family trees
d. transport systems
49. In a graph representing a social network, what do vertices typically represent?
a. Friendships
b. People
c. Messages
d. Groups
50. In a transport system graph, what represents the roads between cities?
a. Vertices
b. Lists
c. Edges
d. Trees
51. Which of the following data structures allows multiple paths between nodes and permits cycles?
a. Tree
b. Graph
c. Stack
d. Queue
52. In a tree data structure, how many distinct paths exist between any two nodes?
a. Multiple paths
b. No path
c. Exactly one
d. Random paths
53. Which data structure is most suitable for representing hierarchical relationships like family trees or organizational charts?
a. Stack
b. Queue
c. Tree
d. Graph
54. The number of edges connected to a vertex in a graph is called its:
a. Weight
b. Degree
c. Path
d. Cycle
55. If a city is connected to three other cities in a graph, its vertex degree is:
a. 1
b. 2
c. 3
d. 4
56. Which of the following is NOT a type of graph?
a. Undirected
b. Weighted
c. Directed
d. Linear
57. What type of edge has a one-way connection in a graph?
a. Undirected edge
b. Weighted edge
c. Directed edge
d. Circular edge
58. If an edge allows travel only from node A to node B, it is called:
a. Undirected
b. Weighted
c. Reversible
d. Directed
59. What type of edge has a two-way connection in a graph?
a. Undirected
b. Weighted
c. Directed
d. Circular

60. Which type of graph has edges that do not indicate a specific direction?

- a. Directed graph
b. Weighted graph
c. Undirected graph
d. Cyclic graph

61. A road map where edges show travel time is an example of _____ graph.

- a. Unweighted
b. Weighted
c. Complete
d. Bipartite

Answers

1. c	2. b	3. d	4. b	5. a	6. a
7. d	8. c	9. c	10. b	11. b	12. c
13. b	14. b	15. b	16. b	17. d	18. c
19. b	20. c	21. c	22. d	23. b	24. c
25. c	26. b	27. c	28. c	29. b	30. c
31. c	32. b	33. b	34. c	35. c	36. b
37. c	38. a	39. b	40. c	41. c	42. a
43. c	44. b	45. c	46. b	47. c	48. c
49. b	50. c	51. b	52. c	53. c	54. b
55. c	56. d	57. c	58. d	59. a	60. c
61. b					