Class 11 Computer Chapter 4: Computational Structures

EXERCISE QUESTIONS

Multiple Choice Questions

Q1 **The function used to add an item at the end of a list in Python:**

a insert()
b append()
c remove()
d pop()

✓ **Answer:** b) append()

Q2 **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

✓ **Answer:** c) Checks if an item exists in the list

Q3 **An operation that removes an item from the top of the stack:**

a Push
b Pop
c Peek
d Add

✓ **Answer:** b) Pop

Q4 **The operation used to add an item to a queue:**

a Dequeue
b Peek
c Enqueue
d Remove

✓ **Answer:** c) Enqueue

taleemzone.com

Q5 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

✓ **Answer:** a) Number of edges from the root to the deepest node

Q6 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

✓ **Answer:** c) Representing connections in a social network

Short Questions

Q1 Explain how the insert() function works in Python lists. Provide an example.

Ans: The insert() function in Python lists adds an element at a specific position.

Syntax:
list.insert(index, element)

- index → position where the element is inserted
- element → item to insert

Example:

```python
fruits = ['apple', 'banana', 'cherry']
fruits.insert(1, 'grape')
print(fruits)
```

Output:

```
['apple', 'grape', 'banana', 'cherry']
```

taleemzone.com

Q2 **Explain the potential issues which could arise when two variables reference the same list in a program. Provide an example.**

Ans: If two variables point to the same list, any change made using one variable will also affect the other because they both share the same data.

Example:

```python
a = [1, 2, 3]
b = a
b.append(4)
print(a)  # Output: [1, 2, 3, 4]  (a is changed too!)
```

Q3 **Define a stack and explain the Last-In, First-Out (LIFO) principle.**

Ans: A stack is a data structure where you can only add or remove items from the top. The last item added is the first one removed. This is called LIFO. For example, like stacking plates — the last plate you put on top is the first one you take off.

Q4 **Differentiate between the Enqueue and Dequeue operations of a queue.**

ANSWER

Aspect	Enqueue	Dequeue
Definition	Adds an element to the rear of the queue	Removes an element from the front of the queue
Type of Operation	Insertion	Deletion
Effect on Queue	Increases the size of the queue	Decreases the size of the queue

Q5 **Name two basic operations performed on a stack.**

Ans: The two basic operations on a stack are:

- **Push:** Add an item to the top.
- **Pop:** Remove an item from the top.

Q6 **What is the difference between enqueue() and dequeue()?**

ANSWER

Function	enqueue()	dequeue()
Purpose	Adds an element to the rear of the queue	Removes an element from the front of the queue
Operation	Insertion	Deletion
Effect	Increases queue size	Decreases queue size
Used In	Adding tasks or data to process later	Processing or retrieving the next task/data

Long Questions

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

Ans: In Python, lists are dynamic in size, meaning their length can grow or shrink during program execution. Unlike arrays in some other programming languages (e.g., C or Java), where the size must be defined at the time of declaration, Python lists can automatically adjust their size as elements are added or removed.

Key Points of Dynamic Size Property:

- **Automatic Memory Management**
  Python handles memory allocation for lists internally. When a list grows beyond its current capacity, Python automatically allocates more memory and copies the list to a new location.
- **No Predefined Size**
  You don't need to declare how many elements a list will hold.

Example:

```python
my_list = []          # Empty list
my_list.append(10)
my_list.append(20)
```

- **Insertion and Deletion**
  You can insert or delete elements at any position. Python internally resizes the list if necessary.
- **Heterogeneous Elements**
  Lists can hold items of different data types, which adds to their flexibility.

How It Makes Lists More Flexible:

- **Ease of Use:** Developers don't have to worry about memory limits or resizing the structure manually.

- **Efficient for Varying Data:** Lists are ideal when the number of elements isn't fixed or known beforehand.
- **Simplifies Code:** You can write cleaner and more concise code without managing capacity or array bounds.
- **Dynamic Operations:** Lists allow stacking (append), queueing (pop, insert), and slicing operations without reallocation concerns.

Example:

```python
data = []  # Empty list
# Dynamically adding elements
for i in range(5):
data.append(i * 10)
print(data)  # Output: [0, 10, 20, 30, 40]
```

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

Ans: A stack is a data structure that works on the LIFO (Last-In, First-Out) principle. The last item added is the first one removed.

Real-Life Example:
Think of a stack of plates in a restaurant. When a new plate is added, it goes on top. When someone takes a plate, they take the top one.

Stack Operations:

- **Push:** Add an item to the top.
- **Pop:** Remove an item from the top.

Python Code Example:

```python
stack = []
# Push operation
stack.append("A")
stack.append("B")
stack.append("C")
print("Stack:", stack)
# Pop operation
print("Popped:", stack.pop())
print("Stack after pop:", stack)
```

Output:

```
Stack: ['A', 'B', 'C']
Popped: C
Stack after pop: ['A', 'B']
```

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

Ans: A queue follows the FIFO (First-In, First-Out) rule — the first item added is the first one removed. We can use a Python list to implement a queue using append() for insertion and pop(0) for deletion.

Python Code Example:

PYTHON

```
queue = []
# Insertion (Enqueue)
queue.append("Apple")
queue.append("Banana")
queue.append("Cherry")
print("Queue after enqueues:", queue)
# Deletion (Dequeue)
print("Dequeued:", queue.pop(0))
print("Queue after dequeue:", queue)
```

Output:

```
Queue after enqueues: ['Apple', 'Banana', 'Cherry']
Dequeued: Apple
Queue after dequeue: ['Banana', 'Cherry']
```