School Connect Online

SCO INTERNATIONAL CODING OLYMPIAD

CLASS 12 SAMPLE QUESTION PAPER

Practice Paper Set B • with answer key and explanations

Designed as a website-ready sample paper for practice, revision, and transparent exam preparation.

- balanced practice across programming concepts, data structures, databases and web development
- answers and explanations included for student self-study and teacher-led review
- question blocks use compact numbering; code and passage content remain inside the main question area

Python	C	Swift	PHP	SQL
Debugging	Data Science	AI Basics	Web Apps	Practice

PDF-ready • Editable Word document • Official SCO cover-page style • Text-based, not a full-page image

Exam Name	SCO International Coding Olympiad
Class / Grade	Class 12
Question Paper Set	Set B
Academic Year	2026-27
Duration	60 minutes
Total Questions	50
Question Type	Objective multiple-choice questions
Calculator	Not allowed unless specifically announced by SCO

Guidelines for the Candidate

• Read every question carefully and select only one correct option.

• All questions are compulsory. There is no negative marking in this practice-ready version unless the official event notification states otherwise.

• Use the question paper for reasoning and rough work only; mark responses on the answer sheet or online test interface as instructed.

• Code snippets should be interpreted exactly as printed. Assume standard language behavior unless the question states otherwise.

• For security, AI and data-science questions, choose the option that follows safe engineering practice and validated reasoning, not only surface-level syntax.

Section A: Advanced Programming Concepts

Q1. Consider an array of n elements. If you want to insert a new element at the beginning of the array, what is the time complexity of this operation (in Big-O notation) in the worst case?

A) O(1)

B) O(log n)

C) O(n)

D) $O(n^2)$

Q2. Given the following Python code snippet, what is the time complexity of finding the maximum element in an unsorted array?

```
def find_max(arr):
    maximum = arr[0]
    for num in arr:
        if num > maximum:
            maximum = num
    return maximum
```

A) O(1)

B) O(log n)

C) O(n)

D) O(n log n)

Q3. In a singly linked list, what is the time complexity of inserting a new node at the beginning of the list?

A) O(n)

B) O(log n)

C) O(1)

D) $O(n^2)$

Q4. Consider a singly linked list with n nodes. What is the worst-case time complexity to search for an element in the list?

A) O(1)

B) O(log n)

C) O(n)

D) $O(n^2)$

Q5. Given a binary search tree (BST), which traversal method will output the nodes in ascending order?

A) Pre-order Traversal

B) Post-order Traversal

C) In-order Traversal

D) Level-order Traversal

Q6. In a binary search tree (BST), if the tree becomes skewed (all nodes have only one child), what is the worst-case time complexity for searching an element?

A) O(1)
B) O(log n)
C) O(n)
D) O(n log n)

Q7. Consider a graph with V vertices and E edges. What is the worst-case time complexity of a Breadth-First Search (BFS) algorithm when implemented using an adjacency list?
A) O(V)
B) O(E)
C) O(V + E)
D) O(V * E)

Q8. Which of the following algorithms is typically used to detect a cycle in a directed graph?
A) Depth-First Search (DFS) with recursion stack
B) Breadth-First Search (BFS)
C) Dijkstra's Algorithm
D) Kruskal's Algorithm

Q9. In Python, the built-in list type is implemented as a dynamic array. Compared to a linked list, which of the following statements is true regarding memory usage and access?
A) Python lists require more memory per element than linked lists but allow O(1) random access.
B) Python lists use less memory per element than linked lists and allow O(1) random access.
C) Python lists require more memory per element than linked lists and allow O(n) random access.
D) Python lists use less memory per element than linked lists but allow O(n) random access.

Q10. Consider the SQL query below, which retrieves data from a table named "Students" with columns "Name" and "Score". What does the following query do?

```
SELECT Name, Score
FROM Students
ORDER BY Score DESC;
```

A) It selects all students and sorts them by name in ascending order.
B) It selects all students and sorts them by score in descending order.
C) It selects only the highest score from the table.
D) It deletes the student records with the lowest scores.

Q11. Consider the following Python function that processes an array by adding 1 to odd numbers and doubling even numbers:

```
def process_data(nums):
    result = []
    for num in nums:
        if num % 2 == 0:
            result.append(num * 2)
        else:
            result.append(num + 1)
```

```
    return result
data = [1, 2, 3, 4, 5]
print(process_data(data))
```

What is the output of this code?

A) [2, 4, 4, 8, 6]

B) [2, 4, 4, 8, 5]

C) [1, 4, 3, 8, 6]

D) [2, 2, 3, 4, 6]

Q12. Consider the recursive function to compute the factorial of a number:

```
def factorial(n):
    if n == 0:
        return 1
    else:
        return n * factorial(n-1)
print(factorial(5))
```

What is the output, and what is the time complexity of this recursive approach?

A) 120, O(1)

B) 120, O(n)

C) 24, O(n)

D) 720, $O(n^2)$

Q13. Given the following Python code to compute the slope (m) and intercept (b) of a simple linear regression line:

```
import numpy as np
x = np.array([1, 2, 3, 4, 5])
y = np.array([2, 4, 6, 8, 10])
m = np.sum((x - np.mean(x)) * (y - np.mean(y))) / np.sum((x - np.mean(x))**2)
b = np.mean(y) - m * np.mean(x)
print(round(m,2), round(b,2))
```

What is the output?

A) 1.00, 1.00

B) 2.00, 3.00

C) 2.00, 0.00

D) 1.50, 2.50

Q14. Examine the following Python code:

```
import numpy as np
data = [10, 20, 30, 40, 50]
std_val = np.std(data)
print(round(std_val,2))
```

What is the printed output (rounded to two decimals)?

A) 10.00

B) 12.00

C) 14.14

D) 15.00

Q15. What is the output of the following Python code?

```
total = 0
for i in range(1, 4):
    for j in range(1, 4):
        total += i * j
print(total)
```

A) 36
B) 42
C) 30
D) 48

Section B: Data Structures, Algorithms and Data Science

Q16. Consider the following Python function:

```
def second_largest(arr):
    unique_arr = list(set(arr))
    unique_arr.sort()
    return unique_arr[-2] if len(unique_arr) >= 2 else None
print(second_largest([3, 5, 1, 2, 5, 3]))
```

What is the output?

A) 1
B) 2
C) 3
D) 5

Q17. Given the following pseudo-Python code for a singly linked list:

```
class Node:
    def __init__(self, data):
        self.data = data
        self.next = None
head = Node(10)
head.next = Node(20)
head.next.next = Node(30)
current = head
count = 0
while current:
    count += 1
    current = current.next
print(count)
```

What is the output?

A) 1
B) 2
C) 3
D) 4

Q18. Examine the following Python code:

```
def fib(n):
    if n <= 1:
        return n
    else:
        return fib(n-1) + fib(n-2)
print(fib(6))
```

What is the output?

A) 5

B) 8

C) 13

D) 21

Q19. Given the following deterministic DFS implementation, what is the output?

```
def dfs(graph, start):
    visited = []
    stack = [start]
    while stack:
        vertex = stack.pop()
        if vertex not in visited:
            visited.append(vertex)
            for nxt in reversed(graph[vertex]):
                if nxt not in visited:
                    stack.append(nxt)
    return visited
graph = {'A': ['B', 'C'], 'B': ['D'], 'C': ['E'], 'D': [], 'E': []}
print(dfs(graph, 'A'))
```

A) ['A', 'B', 'D', 'C', 'E']

B) ['A', 'C', 'E', 'B', 'D']

C) ['A', 'C', 'B', 'D', 'E']

D) ['A', 'B', 'C', 'E', 'D']

Q20. Consider the following Python function to compute the median of a list:

```
def median(lst):
    lst = sorted(lst)
    n = len(lst)
    mid = n // 2
    if n % 2 == 0:
        return (lst[mid-1] + lst[mid]) / 2
    else:
        return lst[mid]
data = [5, 3, 8, 1, 9]
print(median(data))
```

What is the output?

A) 3

B) 5
C) 6
D) 8

Q21. Consider the following Python code snippet:

```
def shout(func):
    def wrapper(*args, **kwargs):
        result = func(*args, **kwargs)
        return result.upper()
    return wrapper
@shout
def greet(name):
    return f"Hello, {name}"
print(greet("Alice"))
```

What is the output of this code?
A) Hello, Alice
B) HELLO, ALICE
C) hello, alice
D) An error occurs

Q22. What will be the output and memory behavior of the following Python code?

```
def count_up_to(n):
    i = 1
    while i <= n:
        yield i
        i += 1
gen = count_up_to(5)
print(list(gen))
```

A) [1, 2, 3, 4, 5] and uses a lot of memory
B) [1, 2, 3, 4, 5] and uses memory efficiently
C) [1, 2, 3, 4] and uses memory efficiently
D) [1, 2, 3, 4, 5] and uses more memory than a list

Q23. Given the Swift code below, what is the purpose of the Summable protocol?

```
protocol Summable {
    static func +(lhs: Self, rhs: Self) -> Self
}
extension Int: Summable {}
extension Double: Summable {}
func add<T: Summable>(_ a: T, _ b: T) -> T {
    return a + b
}
```

A) It restricts the generic function to work only with integers.
B) It ensures that any type passed to add supports the addition operator.
C) It forces all types to be converted to strings before addition.

D) It makes the function work only with Swift arrays.

Q24. Consider the following Swift code snippet:

```
var optionalString: String? = "Hello"
if let unwrappedString = optionalString {
   print(unwrappedString + " World!")
} else {
   print("No value")
}
```

What will this code print?

A) Hello

B) Hello World!

C) No value

D) An error occurs

Q25. Consider the following C code snippet:

```
#include <stdio.h>
int main() {
   int arr[] = {10, 20, 30, 40, 50};
   int *ptr = arr;
   ptr += 3;
   printf("%d\n", *ptr);
   return 0;
}
```

What is the output of this code?

A) 10

B) 20

C) 40

D) 50

Q26. Examine the following C code:

```
#include <stdio.h>
#include <stdlib.h>
int main() {
   int *arr = (int *)malloc(5 * sizeof(int));
   for (int i = 0; i < 5; i++) {
      arr[i] = i * 2;
   }
   printf("%d\n", arr[3]);
   free(arr);
   return 0;
}
```

What value is printed, and why is free(arr) important?

A) 6, and free(arr) deallocates the memory to prevent memory leaks.

B) 6, and free(arr) is not necessary.

C) 8, and free(arr) increases the speed of the program.

D) 6, and free(arr) is used to reallocate memory.

Q27. Given the following PHP code snippet:

```
<?php
$fruits = array("apple", "banana", "cherry");
$fruits[] = "date";
echo $fruits[1];
?>
```

What is the output?

A) apple

B) banana

C) cherry

D) date

Q28. Consider the following SQL query executed on two tables: Students(StudentID, Name) and Scores(StudentID, Score). What does the query do?

```
SELECT Students.Name, AVG(Scores.Score) AS AverageScore
FROM Students
JOIN Scores ON Students.StudentID = Scores.StudentID
GROUP BY Students.Name
ORDER BY AverageScore DESC;
```

A) It selects the names of students and their average score, sorted in descending order of the average score.

B) It selects all students with the highest score only.

C) It deletes records from the Scores table.

D) It returns student names in alphabetical order.

Q29. A developer is designing a RESTful API for a book store application. Which HTTP method should be used to update the details of an existing book?

A) GET

B) POST

C) PUT

D) DELETE

Q30. A PHP developer is writing code to query a MySQL database. Which of the following practices best helps prevent SQL injection attacks?

```
<?php
// Unsafe query
$query = "SELECT * FROM users WHERE username = '".$_POST['username']."'";
?>
```

A) Directly echoing the SQL query.

B) Using parameterized queries or prepared statements with PDO or MySQLi.

C) Storing SQL queries in plain text files.

D) Concatenating user input directly into the query.

Section C: Multi-language Programming and Web Applications

Q31. Consider the following Python code that uses a decorator to modify the behavior of a function:

```
def shout(func):
    def wrapper(*args, **kwargs):
        result = func(*args, **kwargs)
        return result.upper()
    return wrapper
@shout
def greet(name):
    return f"Hello, {name}"
print(greet("Alice"))
```

What is the output, and what is the primary purpose of the decorator?

A) "Hello, Alice" – The decorator does nothing

B) "HELLO, ALICE" – The decorator converts the result to uppercase

C) "hello, alice" – The decorator converts the result to lowercase

D) An error occurs due to improper use of decorators

Q32. Review the following Swift code:

```
var optionalString: String? = "Hello, World!"
if let unwrapped = optionalString {
    print(unwrapped)
} else {
    print("No value")
}
```

What does this code print, and why is optional binding used?

A) Prints "No value" because the optional is nil

B) Prints "Hello, World!" because the optional is safely unwrapped

C) Causes a runtime error due to unwrapping

D) Prints nothing due to incorrect syntax

Q33. Consider the following C code:

```
#include <stdio.h>
int main() {
    int arr[] = {10, 20, 30, 40, 50};
    int *ptr = arr;
    ptr += 3;
    printf("%d\n", *ptr);
    return 0;
}
```

What is the output of this code, and why?

A) 10 – The pointer remains at the start

B) 20 – The pointer moves one position

C) 40 – The pointer moves to the fourth element

D) 50 – The pointer moves to the last element

Q34. A PHP developer wants to securely retrieve user information from a MySQL database. Which code snippet best prevents SQL injection?

A)

```
$username = $_POST['username'];
$query = "SELECT * FROM users WHERE username = '$username'";
$result = mysqli_query($conn, $query);
```

B)

```
$username = $_POST['username'];
$query = "SELECT * FROM users WHERE username = ?";
$stmt = mysqli_prepare($conn, $query);
mysqli_stmt_bind_param($stmt, "s", $username);
mysqli_stmt_execute($stmt);
$result = mysqli_stmt_get_result($stmt);
```

C)

```
$username = $_GET['username'];
$query = "SELECT * FROM users WHERE username = '$username'";
$result = mysqli_query($conn, $query);
```

D)

```
$username = $_POST['username'];
$query = "DELETE FROM users WHERE username = '$username'";
mysqli_query($conn, $query);
```

Q35. Given the following SQL query:

```
SELECT department, COUNT(*) AS EmployeeCount
FROM Employees
WHERE salary > 50000
GROUP BY department
HAVING COUNT(*) > 5
ORDER BY EmployeeCount DESC;
```

What does this query accomplish?

A) Retrieves all employees with a salary over 50000

B) Counts the number of employees in each department with a salary over 50000, only for departments with more than 5 such employees, sorted by count descending

C) Deletes departments with fewer than 5 employees

D) Updates the salary of employees in departments with more than 5 employees

Q36. What is the output of the following Python code snippet?

```
nums = [1, 2, 3, 4, 5]
squared_even = list(map(lambda x: x**2, filter(lambda x: x % 2 == 0, nums)))
print(squared_even)
```

A) [1, 9, 25]

B) [4, 16]

C) [2, 4, 6, 8, 10]

D) [1, 4, 9, 16, 25]

Q37. A developer is building a RESTful API for a social media platform. To update a user's profile information, which HTTP method should be used?

A) GET

B) POST

C) PUT

D) DELETE

Q38. Consider the following Swift code snippet:

```
func divide(_ numerator: Double, _ denominator: Double) -> Double? {
    if denominator == 0 {
        return nil
    }
    return numerator / denominator
}
if let result = divide(10, 2) {
    print(result)
} else {
    print("Error: Division by zero")
}
```

What is printed, and why is error handling important here?

A) 5.0; it prevents crashes by safely handling division by zero

B) "Error: Division by zero"; because the denominator is zero

C) 10.0; because Swift ignores the denominator

D) An error occurs due to division by zero

Q39. Examine the following C code snippet:

```
#include <stdio.h>
#include <string.h>
int main() {
    char str[] = "Hello, World!";
    char *ptr = str;
    ptr += 7;
    printf("%s\n", ptr);
    return 0;
}
```

What is the output, and what concept does this illustrate?

A) "World!"; it illustrates pointer arithmetic and string slicing

B) "Hello,"; it illustrates memory allocation

C) "Hello, World!"; it illustrates the complete string copy

D) "World"; it illustrates string termination

Q40. Consider the following Python code that analyzes student scores:

```
import pandas as pd
data = {'Student': ['Alice', 'Bob', 'Charlie', 'David'],
        'Score': [85, 92, 78, 90]}
df = pd.DataFrame(data)
mean_score = df['Score'].mean()
df['AboveAverage'] = df['Score'] > mean_score
print(df)
```

What is the output, and what does the new column "AboveAverage" represent?

A) It prints the DataFrame with a new column that shows True if a student's score is above the mean and False otherwise.

B) It prints only the mean score.

C) It deletes the "Score" column from the DataFrame.

D) It sorts the DataFrame by the "AboveAverage" column.

Achievers Section: Applied Coding and Reasoning

Q41. A web application needs to log every call to its endpoint functions. A developer writes the following decorator in Python:

```
def log_call(func):
    def wrapper(*args, **kwargs):
        print(f"Function '{func.__name__}' was called")
        result = func(*args, **kwargs)
        return result
    return wrapper
@log_call
def process_order(order_id):
    return f"Order {order_id} processed"
print(process_order(101))
```

What is the output of this code, and what is the purpose of the decorator?

A)

Order 101 processed

B)

Function 'process_order' was called

Order 101 processed

C)

Order 101 processed

Function 'process_order' was called

D) An error occurs due to improper decorator usage

Q42. A developer optimizes a recursive Fibonacci function using memoization:

```
def fib(n, memo={}):
    if n in memo:
        return memo[n]
    if n <= 1:
        return n
```

```
    memo[n] = fib(n-1, memo) + fib(n-2, memo)
    return memo[n]
print(fib(10))
```

What is the output and the benefit of using memoization in this function?

A) 55; Memoization reduces redundant calculations, improving time complexity.

B) 89; Memoization increases memory usage unnecessarily.

C) 55; Memoization makes the function recursive but slower.

D) 34; Memoization has no impact on this simple function.

Q43. A data scientist is analyzing a CSV file containing student test scores. They use the following code:

```
import pandas as pd
data = {
    'Class': ['A', 'B', 'A', 'B', 'C'],
    'Score': [88, 92, 76, 81, 95]
}
df = pd.DataFrame(data)
result = df.groupby('Class')['Score'].mean().round(2)
print(result)
```

What does this code output and what does it compute?

A) It prints the total scores per class.

B) It prints the average score for each class, rounded to two decimals.

C) It prints the highest score in each class.

D) It prints a sorted list of all scores.

Q44. A Swift developer writes the following code to filter an array of integers and multiply each even number by 2:

```
let numbers = [1, 2, 3, 4, 5, 6]
let processed = numbers.filter { $0 % 2 == 0 }.map { $0 * 2 }
print(processed)
```

What is the output of this code?

A) [2, 4, 6, 8, 10, 12]

B) [4, 8, 12]

C) [2, 8, 12]

D) [1, 3, 5]

Q45. Consider the following C code snippet:

```
#include <stdio.h>
#include <stdlib.h>
int main() {
    int *arr = (int *)malloc(5 * sizeof(int));
    for (int i = 0; i < 5; i++) {
        arr[i] = (i + 1) * 10;
    }
    int *ptr = arr;
    ptr += 2;
    printf("%d\n", *ptr);
```

```
    free(arr);
    return 0;
}
```

What does this code print, and why is dynamic memory allocation important here?

A) 10; Dynamic memory is not used efficiently.

B) 30; Dynamic memory allocation allows allocation of memory during runtime.

C) 50; Memory is allocated but not freed correctly.

D) 40; Dynamic memory allocation automatically sorts the array.

Q46. A PHP developer writes code to safely retrieve user details from a MySQL database. Which snippet best prevents SQL injection?

A)

```
$username = $_POST['username'];
$query = "SELECT * FROM users WHERE username = '$username'";
$result = mysqli_query($conn, $query);
```

B)

```
$username = $_POST['username'];
$query = "SELECT * FROM users WHERE username = ?";
$stmt = mysqli_prepare($conn, $query);
mysqli_stmt_bind_param($stmt, "s", $username);
mysqli_stmt_execute($stmt);
$result = mysqli_stmt_get_result($stmt);
```

C)

```
$username = $_GET['username'];
$query = "SELECT * FROM users WHERE username = '$username'";
$result = mysqli_query($conn, $query);
```

D)

```
$username = $_POST['username'];
$query = "DELETE FROM users WHERE username = '$username'";
mysqli_query($conn, $query);
```

Q47. A company wants to give a bonus to employees whose salary is above the average salary. Consider the following SQL query:

```
SELECT name, salary
FROM employees
WHERE salary > (SELECT AVG(salary) FROM employees);
```

What does this query do?

A) Retrieves all employees with salaries below average.

B) Retrieves the names and salaries of employees who earn more than the average salary.

C) Updates employee salaries.

D) Deletes employees with below-average salaries.

Q48. A developer creates a REST API endpoint using Node.js that returns a JSON response containing product details. Which HTTP status code should be returned for a successful GET request, and what is the correct content type header?

A) 404 and "text/html"

B) 200 and "application/json"

C) 500 and "application/xml"

D) 302 and "text/plain"

Q49. A Python developer implements Dijkstra's algorithm to find the shortest path in a weighted graph. Consider this simplified code snippet:

```
import heapq
def dijkstra(graph, start):
    distances = {vertex: float('inf') for vertex in graph}
    distances[start] = 0
    pq = [(0, start)]
    while pq:
        current_distance, current_vertex = heapq.heappop(pq)
        if current_distance > distances[current_vertex]:
            continue
        for neighbor, weight in graph[current_vertex]:
            distance = current_distance + weight
            if distance < distances[neighbor]:
                distances[neighbor] = distance
                heapq.heappush(pq, (distance, neighbor))
    return distances
graph = {
    'A': [('B', 5), ('C', 1)],
    'B': [('A', 5), ('C', 2), ('D', 1)],
    'C': [('A', 1), ('B', 2), ('D', 4), ('E', 8)],
    'D': [('B', 1), ('C', 4), ('E', 3), ('F', 6)],
    'E': [('C', 8), ('D', 3)],
    'F': [('D', 6)]
}
print(dijkstra(graph, 'A'))
```

What does this code output, and what is the significance of the priority queue (heap) in Dijkstra's algorithm?

A) It outputs the shortest distances from 'A' to all other vertices; the heap ensures that the next closest vertex is processed efficiently.

B) It outputs the longest distances; the heap is used for sorting alphabetically.

C) It outputs the sum of all weights; the heap is not significant.

D) It outputs only the distance to vertex 'F'; the heap causes delays.

Q50. A developer trains a logistic regression model to classify emails as spam (1) or not spam (0) using scikit-learn. Consider this code snippet:

```
from sklearn.linear_model import LogisticRegression
from sklearn.model_selection import train_test_split
from sklearn.metrics import accuracy_score
import numpy as np
# Sample dataset: Features (word frequency) and labels
X = np.array([[0.1, 0.3], [0.4, 0.5], [0.2, 0.1], [0.6, 0.8], [0.5, 0.4]])
y = np.array([0, 1, 0, 1, 1])
X_train, X_test, y_train, y_test = train_test_split(X, y, test_size=0.4, random_state=42)
model = LogisticRegression()
model.fit(X_train, y_train)
predictions = model.predict(X_test)
print(accuracy_score(y_test, predictions))
```

What is the role of train_test_split and accuracy_score in this code, and what does the output represent?

A) train_test_split divides the dataset into training and testing sets; accuracy_score calculates the percentage of correct predictions on the test set.

B) train_test_split sorts the data; accuracy_score measures the time taken for predictions.

C) train_test_split is used for hyperparameter tuning; accuracy_score prints the model's loss.

D) train_test_split duplicates the dataset; accuracy_score compares different models.

Answer Key

Q.No.	Ans	Q.No.	Ans	Q.No.	Ans	Q.No.	Ans	Q.No.	Ans
1	C	11	A	21	B	31	B	41	B
2	C	12	B	22	B	32	B	42	A
3	C	13	C	23	B	33	C	43	B
4	C	14	C	24	B	34	B	44	B
5	C	15	A	25	C	35	B	45	B
6	C	16	C	26	A	36	B	46	B
7	C	17	C	27	B	37	C	47	B
8	A	18	B	28	A	38	A	48	B
9	A	19	A	29	C	39	A	49	A
10	B	20	B	30	B	40	A	50	A

Detailed Explanations

Q1. Answer: C. In an array, inserting an element at the beginning requires shifting all existing elements one position to the right. In the worst-case scenario, this involves moving n elements, making the time complexity O(n).

Q2. Answer: C. The code iterates through each element of the array exactly once, comparing it with the current maximum. Therefore, the time complexity is O(n).

Q3. Answer: C. Inserting a node at the beginning of a singly linked list is done by creating the node and updating the head pointer. This operation takes constant time, O(1), regardless of the list's length.

Q4. Answer: C. In the worst case, you may have to traverse the entire linked list to find the element or determine it's not present. Hence, the search operation in a singly linked list is O(n).

Q5. Answer: C. In-order traversal of a BST visits nodes in the order: left subtree, node, then right subtree. This results in the nodes being visited in ascending order.

Q6. Answer: C. A skewed BST essentially behaves like a linked list, where in the worst-case you have to traverse all n nodes. Thus, the search time complexity degrades to O(n).

Q7. Answer: C. BFS visits each vertex once and inspects all the edges in the adjacency list. Thus, the worst-case time complexity is O(V + E).

Q8. Answer: A. Cycle detection in a directed graph is commonly performed using DFS along with a recursion stack to track the vertices in the current path. If a vertex is encountered that is already in the recursion stack, a cycle is detected. BFS, Dijkstra's, and Kruskal's algorithms are not typically used for this purpose.

Q9. Answer: A. Python's list type is implemented as a dynamic array, which provides O(1) random access. However, dynamic arrays typically allocate extra memory to allow for efficient resizing, which can result in higher memory overhead per element compared to a linked list, which only stores pointers and values.

Q10. Answer: B. The SQL query selects the columns "Name" and "Score" from the "Students" table and orders the results by "Score" in descending order (highest score first). It does not filter or delete any records.

Q11. Answer: A. For each element: 1 becomes 1+1=2 (odd), 2 becomes 2×2=4 (even), 3 becomes 3+1=4, 4 becomes 4×2=8, and 5 becomes 5+1=6. Thus, the output is [2, 4, 4, 8, 6].

Q12. Answer: B. Factorial of 5 = 5×4×3×2×1 = 120. The recursive function calls itself n times, resulting in a time complexity of O(n).

Q13. Answer: C. The relationship is y = 2x. Thus, the slope m = 2.00 and intercept b = 0.00. The code rounds these to two decimals, so the output is "2.00 0.00."

Q14. Answer: C. The mean is 30. The variance is ((400+100+0+100+400)/5 = 200), so the standard deviation is sqrt(200) ≈ 14.14.

Q15. Answer: A. Calculation: For i=1: (1×1 + 1×2 + 1×3) = 6; i=2: (2×1 + 2×2 + 2×3) = 12; i=3: (3×1 + 3×2 + 3×3) = 18. Total = 6+12+18 = 36.

Q16. Answer: C. The array [3, 5, 1, 2, 5, 3] becomes unique [1, 2, 3, 5] when converted to a set and sorted. The second largest is 3.

Q17. Answer: C. The code creates three nodes (10, 20, 30) and then counts them using a while loop. Therefore, the output is 3.

Q18. Answer: B. The Fibonacci sequence is: 0, 1, 1, 2, 3, 5, 8,... For n=6 (with 0-indexing, fib(6)=8), the output is 8.

Q19. Answer: A. The reversed neighbor push puts B on top of the stack first, so the traversal is A, B, D, C, E.

Q20. Answer: B. The sorted list is [1, 3, 5, 8, 9]. With 5 elements (odd number), the median is the middle element, which is 5.

Q21. Answer: B. The @shout decorator wraps the greet function, so when greet("Alice") is called, its return value is converted to uppercase by the wrapper function. Therefore, the output is "HELLO, ALICE".

Q22. Answer: B. The generator count_up_to yields numbers one at a time, so it produces [1, 2, 3, 4, 5] when converted to a list. Generators are memory-efficient because they generate items on the fly instead of storing the entire sequence in memory.

Q23. Answer: B. The generic constraint T: Summable allows add to use + only for types declared to support the required operator.

Q24. Answer: B. The code uses optional binding (if let) to safely unwrap the optional. Since optionalString contains "Hello", it is unwrapped and concatenated with " World!", resulting in "Hello World!".

Q25. Answer: C. ptr initially points to arr[0]. The statement ptr += 3; moves the pointer to the fourth element (index 3) in the array, which is 40. Thus, the output is 40.

Q26. Answer: A. The array values are [0, 2, 4, 6, 8]. arr[3] is 6. Calling free(arr) releases the allocated memory, preventing memory leaks.

Q27. Answer: B. The PHP array initially contains "apple" (index 0), "banana" (index 1), and "cherry" (index 2). After appending "date", "banana" remains at index 1. Thus, the output is "banana".

Q28. Answer: A. The query performs an inner join between Students and Scores based on StudentID, calculates the average score for each student, groups the results by student name, and sorts them in descending order by average score.

Q29. Answer: C. The PUT method is typically used in RESTful APIs to update an existing resource. GET retrieves data, POST creates new resources, and DELETE removes a resource.

Q30. Answer: B. Using parameterized queries or prepared statements ensures that user input is properly escaped and treated as data, not executable code. This practice is essential to prevent SQL injection vulnerabilities. Options A, C, and D do not secure the query against injection.

Q31. Answer: B. The @shout decorator wraps the greet function so that its return value is passed through the wrapper, which converts it to uppercase. Thus, greet("Alice") returns "HELLO, ALICE".

Q32. Answer: B. The code uses optional binding (if let) to safely unwrap the optional string. Since optionalString contains "Hello, World!", it is unwrapped and printed, avoiding potential runtime errors.

Q33. Answer: C. The pointer ptr initially points to arr[0]. After ptr += 3;, it points to arr[3], which holds the value 40. Thus, the code prints 40.

Q34. Answer: B. Option B uses prepared statements with parameter binding, which ensures that the user input is treated as data, not as part of the SQL command. This technique is crucial for preventing SQL injection attacks.

Q35. Answer: B. The query filters employees with a salary above 50000, groups the results by department, counts the number of such employees, and then only includes departments with more than 5 employees (using HAVING). Finally, it orders the results by the employee count in descending order.

Q36. Answer: B. The code first filters the list to include only even numbers ([2, 4]) and then maps these numbers to their squares ($2^2 = 4$, $4^2 = 16$). Thus, the output is [4, 16].

Q37. Answer: C. PUT is the HTTP method used to update an existing resource (e.g., a user's profile). GET retrieves data, POST is used for creating new resources, and DELETE removes a resource.

Q38. Answer: A. The function divide returns 10 / 2 = 5.0, since the denominator is not zero. The optional binding (if let) safely unwraps the result, preventing runtime errors that would occur if division by zero were attempted.

Q39. Answer: A. The pointer ptr is moved 7 positions forward, so it points to the beginning of "World!" in the string. Printing ptr outputs "World!". This demonstrates pointer arithmetic in C.

Q40. Answer: A. The code calculates the mean score (85+92+78+90)/4 = 86.25. The new column "AboveAverage" is a boolean that is True for scores above 86.25 and False otherwise. Thus, it indicates which students scored above the average. The DataFrame with this extra column is printed.

Q41. Answer: B. The decorator @log_call wraps the process_order function. When process_order(101) is called, the wrapper first prints "Function 'process_order' was called" and then calls the original function, returning "Order 101 processed". Option B correctly reflects the sequence of outputs.

Q42. Answer: A. The Fibonacci number for n=10 is 55. Memoization stores previously computed results, which avoids recalculating Fibonacci numbers multiple times, thus significantly reducing the time complexity from exponential to linear.

Q43. Answer: B. The code groups the DataFrame by 'Class' and computes the mean score for each group. The result is rounded to two decimal places. This operation produces the average score for classes A, B, and C, which is then printed.

Q44. Answer: B. The code first filters the array to include only even numbers ([2, 4, 6]). Then, it maps each even number to its double, resulting in [4, 8, 12]. Therefore, the output is [4, 8, 12].

Q45. Answer: B. The array is filled with values [10, 20, 30, 40, 50]. The pointer ptr is advanced by 2 positions, pointing to arr[2] (which is 30). The code then prints 30. Dynamic memory allocation with malloc allows allocating memory during runtime, making the program flexible.

Q46. Answer: B. Option B uses prepared statements with parameter binding, ensuring that user input is properly escaped and treated as data rather than part of the SQL command, thus preventing SQL injection.

Q47. Answer: B. The subquery (SELECT AVG(salary) FROM employees) computes the average salary. The main query then selects employees whose salary is greater than this average, retrieving their names and salaries.

Q48. Answer: B. A successful GET request typically returns HTTP status code 200. When returning JSON data, the content type header should be set to "application/json". This ensures that clients correctly interpret the response as JSON.

Q49. Answer: A. The code calculates the shortest distance from the start vertex 'A' to all other vertices in the graph. The priority queue (implemented as a heap) is crucial because it always extracts the vertex with the smallest tentative distance, ensuring efficient processing of nodes. Option A accurately describes the output and the role of the heap.

Q50. Answer: A. The function train_test_split splits the dataset into training and testing sets so that the model can be trained on one portion of the data and then evaluated on unseen data. accuracy_score then calculates the proportion of correctly predicted labels in the test set. This output represents the model's prediction accuracy.