School Connect Online

SCO INTERNATIONAL OLYMPIAD

GRADE 9 CODING OLYMPIAD

Sample question paper with answer key and explanations

Designed as a practice paper from prior-year SCO coding pathways for guided preparation

- grade-fit coding guidance for Class 9 / secondary-level learners globally
- programming concepts, application development, advanced Python, data analytics, Swift, Objective-C, PHP, SQL, and project-based coding
- practice roadmap, assessment readiness, secure coding awareness, and future-ready computational growth

Python	Algorithms	Swift	Objective-C	SQL
Data Science	Web Apps	Debugging	AI Basics	Security

SCO International Coding Olympiad

Class 9 | Question Paper Set B - Sample Practice Paper | SCO International Coding Olympiad

Detail	Description
Exam Name	SCO International Coding Olympiad
Class / Grade	Class 9
Paper Type	Objective Type - Multiple Choice Questions
Total Questions	50
Duration	60 minutes
Answering Rule	Choose ONE correct option for each question.
Learning Focus	Programming concepts, application development, Python, data analytics, Swift, Objective-C, PHP, SQL, debugging and project-based reasoning.
Download Use	PDF-ready website download for students, teachers, schools and parents.

Candidate Guidelines

- Read each question carefully before selecting the answer.
- Only one option is correct for each question.
- Calculator use is not required for this paper.
- Code snippets are designed to test logic, debugging, syntax awareness and computational thinking.
- For school-supervised exams, follow the instructions of the invigilator or the SCO online exam interface.
- The answer key and explanations are provided for academic review and post-practice learning.

Academic and global alignment note

This Class 9 paper is aligned to age-appropriate computational thinking, algorithms, programming, data and analysis, secure web-development awareness and project-based coding readiness.

Question blocks use compact question-number labels so the space is reserved for code, tables, scenarios and explanations.

School Connect Online

Question Paper

Section 1: Programming Basics

Q1.

What will be the output of the following Python function?

```
def mystery(x, y=[]):
    y.append(x)
    return y
print(mystery(5))
print(mystery(10))
```

A) [5] [10]
B) [5] [5, 10]
C) [5] [10]
D) [5] [10, 5]

Q2.

Which of the following sorting algorithms has the best worst-case time complexity?

A) Merge Sort
B) Bubble Sort
C) Quick Sort
D) Selection Sort

Q3.

In Swift, what is the purpose of Automatic Reference Counting (ARC)?

A) It manually manages memory allocation.
B) It automatically deallocates objects when they are no longer needed.
C) It prevents memory leaks by garbage collecting unused variables.
D) It allows direct access to system memory.

Q4.

What will be the output of the following SQL query?

```
SELECT COUNT(DISTINCT subject) FROM exams WHERE score > 80;
```

A) Counts only unique subjects where score > 80
B) Counts all rows where score > 80
C) Counts distinct scores greater than 80
D) Returns an error

Q5.

In PHP, which function is used to delete all session variables?

A) session_start()
B) session_unset()
C) session_destroy()
D) session_delete()

Q6.

Which statement is true about pointers in Objective-C?

A) int *p; stores the value of an integer.
B) int *p; stores the address of an integer variable.
C) int *p; creates a new integer.
D) int *p; removes an integer from memory.

Q7.

Which Python library is most commonly used for handling large datasets in Data Science?

A) matplotlib
B) seaborn
C) pandas
D) tensorflow

Q8.

Which PHP function escapes special characters in a string before it is used in a mysqli SQL query?

A) query_escape()
B) mysqli_real_escape_string()
C) strip_tags()
D) session_escape()

Q9.

What is the time complexity of the following function?

```
def fun(n):
    for i in range(n):
        for j in range(i):
            print(i, j)
```

A) $O(n^2)$
B) O(n log n)
C) O(n)
D) $O(n^3)$

Q10.

In Swift, what is a protocol?

A) A predefined function
B) A template that defines methods and properties for a class
C) A way to store integer values
D) A type of Swift variable

Q11.

Which Python library provides statistical functions like mean, median, and mode?

A) numpy
B) scipy
C) statistics
D) math

Q12.

What is the purpose of a FOREIGN KEY in SQL?

A) To store temporary data
B) To enforce referential integrity between two tables
C) To delete rows automatically
D) To speed up queries

Q13.

Which function opens a file in Objective-C?

A) fopen()
B) open_file()
C) create()

D) file.read()

Q14.

What is the correct way to find the length of a string in PHP?

A) count()
B) length()
C) strlen()
D) sizeof()

Q15.

What will be the output of this recursive Python function?

```
def mystery(n):
    if n == 0:
        return 0
    return n + mystery(n-1)
print(mystery(4))
```

A) 10
B) 15
C) 20
D) 5

Section 2: Application Development

Q16.

Which algorithm is most efficient for sorting a nearly sorted list of 1,000 elements?

A) QuickSort
B) Bubble Sort
C) Insertion Sort
D) Merge Sort

Q17.

What is the output of the following Swift code?

```
var numbers = [1, 2, 3]
numbers.map { $0 * 2 }.filter { $0 > 3 }
print(numbers)
```

A) [2, 4, 6]
B) [4, 6]
C) [1, 2, 3]
D) Compilation Error

Q18.

In Objective-C, what is the purpose of the @synthesize directive?

A) To declare a new class
B) To generate getter and setter methods for a property
C) To include a header file
D) To handle memory management

Q19.

Which PHP code snippet is most suitable for safely displaying a posted username on a web page?

A) $user = $_GET['username'];
B) $user = $_POST['username'];

C) $user = htmlspecialchars($_POST['username'], ENT_QUOTES, 'UTF-8');
D) $user = $_REQUEST['username'];

Q20.

A full-stack student management system has two tables: students(id, name, age) and scores(student_id, subject, marks). Which SQL query fetches student names with their subject and marks where marks are greater than 90?

A) SELECT name FROM students WHERE id IN (SELECT student_id FROM scores WHERE marks > 90);
B) SELECT students.name, scores.subject, scores.marks FROM students INNER JOIN scores ON students.id = scores.student_id WHERE scores.marks > 90;
C) SELECT DISTINCT name FROM scores WHERE marks > 90;
D) SELECT name FROM students WHERE marks > 90;

Q21.

What does the following Python decorator do?

```
def log(func):
    def wrapper(*args, **kwargs):
        print(f"Calling {func.__name__}")
        return func(*args, **kwargs)
    return wrapper
```

A) Measures execution time
B) Logs function calls
C) Validates input arguments
D) Caches results

Q22.

In Pandas, which code calculates the mean salary for each department in a DataFrame df with columns Department and Salary?

A) df.groupby('Department').sum()
B) df.groupby('Department').mean()
C) df.groupby('Salary').mean()
D) df.mean()

Q23.

Which Python library is used to calculate the median of a dataset?

A) NumPy
B) Pandas
C) SciPy
D) Matplotlib

Q24.

In Swift, what is the purpose of the guard statement?

A) To handle optional unwrapping with early exit
B) To declare a constant
C) To loop through an array
D) To define a protocol

Q25.

Which PHP function prevents SQL injection when executing a query?

A) mysql_real_escape_string()
B) mysqli_prepare()
C) htmlentities()
D) strip_tags()

Q26.

In Objective-C, what is the difference between retain and assign in a property declaration?

A) retain increases the reference count; assign does not.

B) retain is used for primitives; assign for objects.

C) assign increases the reference count; retain does not.

D) Both are identical.

Q27.

What is the output of this Python code?

```
numbers = [lambda x: x + i for i in range(3)]
print([n(5) for n in numbers])
```

A) [5, 6, 7]

B) [7, 7, 7]

C) [6, 7, 8]

D) [5, 5, 5]

Q28.

Which SQL clause is used to combine rows from two tables where there is no match?

A) INNER JOIN

B) LEFT JOIN

C) FULL OUTER JOIN

D) CROSS JOIN

Q29.

In Pandas, what does df.dropna() do?

A) Removes rows with missing values

B) Fills missing values with zero

C) Drops duplicate rows

D) Sorts the DataFrame

Q30.

Which data structure is best suited for implementing a LIFO (Last-In-First-Out) system?

A) Queue

B) Stack

C) Linked List

D) Tree

Section 3: Coding Projects

Q31.

Full-Stack Web Development Project Challenge

You are developing a web-based student management system. The frontend uses HTML, CSS and JavaScript. The backend uses PHP and SQL. The database has two tables:

```
CREATE TABLE students (id INT PRIMARY KEY, name VARCHAR(100), age INT);
CREATE TABLE scores (student_id INT, subject VARCHAR(50), marks INT, FOREIGN KEY (student_id) REFERENCES students(id));
```

Which query fetches student names along with the subject and marks for all scores greater than 90?

A) SELECT name FROM students WHERE id IN (SELECT student_id FROM scores WHERE marks > 90);

B) SELECT students.name, scores.subject, scores.marks FROM students INNER JOIN scores ON students.id = scores.student_id WHERE scores.marks > 90;

C) SELECT DISTINCT name FROM scores WHERE marks > 90;

D) SELECT name FROM students WHERE marks > 90;

Q32.

Mobile App Development Using Swift

You are developing a Swift-based iOS app for an e-commerce platform. You need to fetch product data from a remote API and display it dynamically.

Which Swift code correctly fetches JSON data using URLSession?

A)

```
let url = URL(string: "https://api.example.com/products")
let data = try Data(contentsOf: url)
print(data)
```

B)

```
let url = URL(string: "https://api.example.com/products")!
let task = URLSession.shared.dataTask(with: url) { data, response, error in
    if let data = data {
        print(String(data: data, encoding: .utf8)!)
    }
}
task.resume()
```

C)

```
let url = URL(string: "https://api.example.com/products")!
let response = URLSession.shared.synchronousData(with: url)
print(response.data)
```

D)

```
let url = "https://api.example.com/products"
let response = URLSession.dataFetch(url)
print(response)
```

Q33.

Data Science & Machine Learning Model in Python

You are building a Machine Learning model to predict house prices using Python. Your dataset (houses.csv) contains:

sqft_living: Size of the house in square feet

bedrooms: Number of bedrooms

bathrooms: Number of bathrooms

price: The house price

Which Python code correctly loads the dataset and prepares it for training using Scikit-learn?

A)

```
import pandas as pd
df = pd.read_csv("houses.csv")
X = df[['sqft_living', 'bedrooms', 'bathrooms']]
y = df['price']
```

B)

```
import numpy as np
data = np.load("houses.csv")
X = data[:, 0:3]
y = data[:, -1]
```

C)

```
import sklearn
data = sklearn.load_dataset("houses.csv")
X = data.features
y = data.target
```

D)

```
python
```

```
import tensorflow as tf
dataset = tf.data.Dataset("houses.csv")
X, y = dataset
```

Q34.

Advanced PHP Web Application Development

You are building a PHP-based user authentication system. The system stores hashed passwords in a database for security.

Which PHP code securely verifies a user's password?

A)

```
if ($_POST['password'] == $hashed_password) {
   echo "Login successful";
}
```

B)

```
if (password_verify($_POST['password'], $hashed_password)) {
   echo "Login successful";
}
```

C)

```
if (hash_equals($_POST['password'], $hashed_password)) {
   echo "Login successful";
}
```

D)

```
if (sha1($_POST['password']) == $hashed_password) {
   echo "Login successful";
}
```

Q35.

SQL Query Optimization in Large-Scale Applications

You are developing a high-performance SQL application for tracking online transactions. The table transactions has millions of rows and stores:

Which SQL query efficiently retrieves the total successful transaction amount per user?

A)

```
SELECT user_id, SUM(amount) FROM transactions WHERE status = 'success';
```

B)

```
SELECT user_id, SUM(amount) FROM transactions WHERE status = 'success'
GROUP BY user_id;
```

C)

```
SELECT SUM(amount) FROM transactions WHERE status = 'success';
```

D)

```
SELECT user_id, amount FROM transactions WHERE status = 'success';
```

Q36.

Scenario: You are building a SwiftUI app that displays a sorted list of students by GPA. The list may contain more than 10,000 records. Which approach is most appropriate when the array can be safely mutated and memory usage should be minimized?

A) Use students.sort { $0.gpa > $1.gpa } on a mutable array.

B) Use lazy.sorted() to avoid materializing the sorted result.

C) Use concurrentPerform for all sorting operations.

D) Convert every record to a String before sorting.

Q37.

Scenario:

You're building a user registration system in PHP with a MySQL database. The code below is vulnerable to SQL injection:

Code Snippet:

```
$username = $_POST['username'];
$query = "INSERT INTO users (username) VALUES ('$username')";
mysqli_query($conn, $query);
```

Which change will best fix the vulnerability?

A) Use mysqli_real_escape_string($conn, $username).
B) Replace $_POST with $_GET.
C) Use a prepared statement with mysqli_prepare().
D) Add htmlspecialchars($username).

Q38.

Scenario:

You're debugging an iOS app where a ViewController isn't deallocating, causing memory leaks. The code uses ARC (Automatic Reference Counting).

Code Snippet:

```
@property (strong, nonatomic) NSMutableArray *data;
- (void)loadData {
    [API fetchDataWithCompletion:^(NSArray *result) {
        self.data = [result mutableCopy]; // Leak occurs here
    }];
}
```

What is causing the memory leak?

A) strong property creates a retain cycle.
B) mutableCopy isn't released.
C) The completion block captures self strongly.
D) NSMutableArray isn't thread-safe.

Q39.

Scenario:

You're analyzing sales data in Pandas. The code merges two DataFrames but produces incorrect results:

Code Snippet:

```
sales = pd.read_csv("sales.csv")  # Columns: Date, Product, Revenue
products = pd.read_csv("products.csv")  # Columns: Product, Category
merged = pd.merge(sales, products, on="Product", how="outer")
total_revenue = merged.groupby("Category")["Revenue"].sum()
```

Why does total_revenue include NaN values?

A) how="outer" includes unmatched rows.
B) Revenue column has missing data.
C) sum() skips NaN by default.
D) The merge key "Product" has duplicates.

Q40.

Scenario:

You're building a multithreaded Python program to process tasks. The code uses a decorator to log execution time but causes race conditions:

Code Snippet:

```
from threading import Thread
import time
def log_time(func):
    def wrapper(*args):
        start = time.time()
        result = func(*args)
```

```
        print(f"{func.__name__} took {time.time() - start}s")
        return result
    return wrapper
@log_time
def process_task(task_id):
    time.sleep(1)
    return f"Task {task_id} done"
threads = [Thread(target=process_task, args=(i,)) for i in range(5)]
[t.start() for t in threads]
```

Why does this code produce unpredictable logs?

A) The decorator isn't thread-safe.
B) time.sleep(1) causes delays.
C) print() isn't synchronized.
D) wrapper lacks **kwargs support.

Section 4: Achievers Section - Case-Based Reasoning

Q41.

Case Study: AI-Based Movie Recommendation System

You are developing a Movie Recommendation System using Python and Data Science.

The system uses a dataset (movies.csv) with the following columns:

You need to suggest the top 5 most popular movies based on their views and ratings.

Which Python code correctly filters the top 5 movies using Pandas?

A)

```
import pandas as pd
df = pd.read_csv("movies.csv")
top_movies = df.sort_values(by=['rating', 'views'], ascending=False).head(5)
print(top_movies)
```

B)

```
import numpy as np
df = np.loadtxt("movies.csv")
top_movies = sorted(df, key=lambda x: (x[3], x[4]), reverse=True)[:5]
print(top_movies)
```

C)

```
import pandas as pd
df = pd.read_csv("movies.csv")
top_movies = df[df['rating'] > 8].sort_values(by='views').head(5)
print(top_movies)
```

D)

```
import pandas as pd
df = pd.read_csv("movies.csv")
top_movies = df[df['views'] > 1_000_000].head(5)
print(top_movies)
```

Q42.

Case Study: Secure User Authentication System (PHP & SQL)

You are creating a PHP-based authentication system for a banking application. The system hashes passwords before storing them in the database.

Which PHP + SQL combination correctly handles user login securely?

A)

```
$password = $_POST['password'];
$result = $db->query("SELECT * FROM users WHERE password = '$password'");
```

B)

```
$password = $_POST['password'];
$hash = password_hash($password, PASSWORD_BCRYPT);
$result = $db->query("SELECT * FROM users WHERE password = '$hash'");
```

C)

```
$password = $_POST['password'];
$stmt = $db->prepare("SELECT password FROM users WHERE username = ?");
$stmt->execute([$_POST['username']]);
$hash = $stmt->fetchColumn();
if (password_verify($password, $hash)) {
    echo "Login successful";
}
```

D)

```
$password = $_POST['password'];
$stmt = $db->prepare("SELECT password FROM users WHERE username = :username");
$stmt->bindParam(':username', $_POST['username']);
$stmt->execute();
if ($stmt->fetchColumn() == md5($password)) {
    echo "Login successful";
}
```

Q43.

Case Study: Building a Chat Application (Swift & Objective-C)

You are developing a real-time chat app for iOS using Swift and Objective-C.

The app uses WebSockets to enable real-time messaging.

Which Swift code correctly implements a WebSocket connection?

A)

```
import Foundation
let url = URL(string: "wss://chat.example.com/socket")!
let task = URLSession.shared.webSocketTask(with: url)
task.resume()
```

B)

```
import WebKit
let socket = WKWebView()
socket.load(URLRequest(url: URL(string: "wss://chat.example.com/socket")!))
```

C)

```
import Foundation
let socket = URLSession.shared.streamTask(withHostName: "chat.example.com", port: 443)
socket.resume()
```

D)

```
import UIKit
let url = URL(string: "https://chat.example.com/socket")!
let session = URLSession(configuration: .default)
let socket = session.dataTask(with: url)
socket.resume()
```

Q44.

Case Study: Data Science and Statistics with Python

You are analyzing e-commerce sales data using Python. Your dataset (sales.csv) contains:

You need to calculate the average sales amount per customer using Python.

Which code correctly calculates this?

A)

```
import pandas as pd
df = pd.read_csv("sales.csv")
avg_sales = df.groupby('customer_id')['amount'].mean()
```

B)

```
import numpy as np
df = np.load("sales.csv")
avg_sales = np.mean(df, axis=0)
```

C)

```
import pandas as pd
df = pd.read_csv("sales.csv")
avg_sales = df['amount'].sum() / df.shape[0]
```

D)

```
import matplotlib.pyplot as plt
df = pd.read_csv("sales.csv")
plt.plot(df['customer_id'], df['amount'])
```

Q45.

Case Study: AI-Based Healthcare Diagnosis System (Python & Machine Learning)

A dataset patients.csv contains patient_id, age, symptoms, diagnosis and probability. You want to train a simple text-based model that predicts a diagnosis category from symptom text. Which approach correctly converts symptoms into numeric features before classification?

A) Use DecisionTreeClassifier directly on the raw symptoms text.

B) Use GaussianNB only on age and ignore symptoms.

C) Use TfidfVectorizer on symptoms and train a classifier such as RandomForestClassifier.

D) Filter rows where symptoms exactly match the new patient's symptoms.

Q46.

Case Study: Banking Transaction Fraud Detection (SQL & Data Science)

A transaction should be flagged if it is failed, the amount is greater than 200000, it happened in the last 24 hours, and the same user has more than 3 failed transactions in that 24-hour window. Which SQL query best captures these conditions?

A) SELECT * FROM transactions WHERE amount > 200000 AND status = 'Failed';

B) SELECT user_id, COUNT(*) AS fail_count FROM transactions WHERE status = 'Failed' AND txn_time >= NOW() - INTERVAL 1 DAY GROUP BY user_id HAVING fail_count > 3;

C) SELECT * FROM transactions t WHERE t.status = 'Failed' AND t.amount > 200000 AND t.txn_time >= NOW() - INTERVAL 1 DAY AND (SELECT COUNT(*) FROM transactions x WHERE x.user_id = t.user_id AND x.status = 'Failed' AND x.txn_time >= NOW() - INTERVAL 1 DAY) > 3;

D) SELECT user_id FROM transactions WHERE txn_type = 'Transfer' AND amount > 200000;

Q47.

Case Study: Predicting Student Performance (Python & Data Science)

A school wants to predict student performance based on attendance and test scores.

The dataset students.csv has the following columns:

Which Python code correctly predicts a student's grade based on attendance and test scores?

A)

```
import pandas as pd
df = pd.read_csv("students.csv")
df.corr()
```

B)

```
import pandas as pd
from sklearn.linear_model import LogisticRegression
df = pd.read_csv("students.csv")
X = df[['attendance', 'test_score']]
y = df['grade']
```

```
model = LogisticRegression().fit(X, y)
print(model.predict([[80, 75]]))
```

C)

```
import pandas as pd
from sklearn.cluster import KMeans
df = pd.read_csv("students.csv")
model = KMeans(n_clusters=3).fit(df[['attendance', 'test_score']])
print(model.labels_)
```

D)

```
import pandas as pd
df = pd.read_csv("students.csv")
df['predicted_grade'] = df['test_score'] * 0.5 + df['attendance'] * 0.5
print(df[['student_id', 'predicted_grade']])
```

Q48.

Case Study: E-Commerce Recommendation System (Python & Machine Learning)

An e-commerce platform wants to suggest product categories using a user's previous purchase_count and avg_rating. Which introductory Python approach is most suitable from the options below?

A) Use the most frequent category for every user.

B) Train a KNeighborsClassifier using purchase_count and avg_rating to predict product_category.

C) Train RandomForestClassifier with an unencoded text column price_range passed directly as a numeric feature.

D) Group by category and return the same top category for all users.

Q49.

Case Study: Secure Login System (PHP & SQL Injection Prevention)

A company stores password_hash values in a database. Which PHP flow correctly prevents SQL injection and verifies the password securely?

A) Build a query by concatenating username and password directly into SQL.

B) Query the database using MD5(password) in the WHERE clause.

C) Use a prepared statement to fetch the stored password_hash by username, then use password_verify($password, $storedHash).

D) Use htmlspecialchars() before concatenating values into SQL.

Q50.

Case Study: Data Analytics in Swift (Swift & Data Science)

A company uses Swift to process sales data and perform analytics.

They store the sales data in an array of dictionaries:

```
let sales = [
   ["region": "North", "amount": 5000],
   ["region": "South", "amount": 7000],
   ["region": "West", "amount": 4500],
   ["region": "East", "amount": 8000]
]
```

Which Swift function correctly finds the total sales amount for all regions?

A)

```
var total = 0
for sale in sales {
   total += sale["amount"]
}
print(total)
```

B)

```
let totalSales = sales.reduce(0) { $0 + ($1["amount"] as! Int) }
```

```
print(totalSales)
```

C)

```
let totalSales = sales.map({ $0["amount"] }).sum()
print(totalSales)
```

D)

```
var total = sales.count
print(total)
```

School Connect Online

Answer Key

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

Detailed Explanations

Q1. Answer: B - [5] [5, 10]
The default mutable list y persists across function calls, so 10 gets added to the same list containing 5.

Q2. Answer: A - Merge Sort
Merge Sort has a worst-case time complexity of O(n log n), better than Bubble Sort ($O(n^2)$), Selection Sort ($O(n^2)$), and Quick Sort ($O(n^2)$ in the worst case).

Q3. Answer: B - It automatically deallocates objects when they are no longer needed.
ARC tracks strong references and automatically frees memory when no more references exist.

Q4. Answer: A - Counts only unique subjects where score > 80
The query counts the distinct values in the subject column only where score > 80.

Q5. Answer: B - session_unset()
session_unset() removes all session variables, whereas session_destroy() destroys the session but doesn't remove variables.

Q6. Answer: B - int *p; stores the address of an integer variable.
*p is a pointer variable that holds the memory address of an integer.

Q7. Answer: C - pandas
pandas provides powerful data manipulation tools, while tensorflow is for machine learning.

Q8. Answer: B - mysqli_real_escape_string()
mysqli_real_escape_string() escapes special characters for the current MySQL connection. However, for modern secure coding, parameterized prepared statements are preferred because they separate SQL logic from user data and reduce SQL-injection risk more reliably.

Q9. Answer: A - O(n^2)
The nested loop runs i times for each n, resulting in O(n^2) complexity.

Q10. Answer: B - A template that defines methods and properties for a class
Protocols define blueprints for properties and methods in Swift classes.

Q11. Answer: C - statistics
The statistics module provides built-in mean, median, mode functions.

Q12. Answer: B - To enforce referential integrity between two tables
A foreign key ensures that a value in one table must exist in another, maintaining data consistency.

Q13. Answer: A - fopen()
fopen() is used in C and Objective-C to open files for reading/writing.

Q14. Answer: C - strlen()
strlen() returns the length of a string.

Q15. Answer: A - 10
This recursively sums numbers from 4 to 0, giving 4 + 3 + 2 + 1 + 0 = 10.

Q16. Answer: C - Insertion Sort
Insertion Sort has a time complexity of O(n) for nearly sorted data, making it faster than QuickSort (O(n log n) average case) or Bubble Sort (O(n^2)). Merge Sort (O(n log n)) is less efficient here due to its fixed divide-and-conquer approach.

Q17. Answer: C - [1, 2, 3]
map and filter return new arrays and do not modify the original numbers array. Hence, numbers remains unchanged.

Q18. Answer: B - To generate getter and setter methods for a property
@synthesize automatically creates accessor methods (getters/setters) for properties, simplifying code in manual memory management (MRC).

Q19. Answer: C - htmlspecialchars($_POST['username'], ENT_QUOTES, 'UTF-8')
htmlspecialchars() encodes special HTML characters before output, helping to prevent cross-site scripting (XSS) when user input is displayed. This is different from SQL-injection prevention, which requires prepared statements.

Q20. Answer: B - SELECT students.name, scores.subject, scores.marks ... INNER JOIN ... WHERE scores.marks > 90
Option B is the most complete query because it joins students and scores, then returns the name, subject and marks for rows where marks are above 90. Option A can return names but not the subject or marks. Options C and D refer to wrong or incomplete table structures.

Q21. Answer: B - Logs function calls

The decorator prints the function name before executing it, enabling logging of function calls.

Q22. Answer: B - df.groupby('Department').mean()
groupby('Department') groups data by department, and .mean() calculates the average salary for each group.

Q23. Answer: A - NumPy
numpy.median() computes the median efficiently. Pandas also has a median method, but the question focuses on basic statistics libraries.

Q24. Answer: A - To handle optional unwrapping with early exit
guard checks conditions and exits the scope (e.g., function) if they fail, ensuring safe unwrapping of optionals.

Q25. Answer: B - mysqli_prepare()
Prepared statements (using mysqli_prepare) with parameterized queries are the most secure way to prevent SQL injection.

Q26. Answer: A - retain increases the reference count; assign does not.
retain is used for objects (manages memory in MRC), while assign is for primitive types (no memory management).

Q27. Answer: B - [7, 7, 7]
The lambda captures the final value of i (2) due to late binding. All lambdas compute 5 + 2 = 7.

Q28. Answer: C - FULL OUTER JOIN
FULL OUTER JOIN returns all rows from both tables, including unmatched ones. LEFT JOIN only includes unmatched rows from the left table.

Q29. Answer: A - Removes rows with missing values
dropna() removes rows or columns containing NaN (missing values).

Q30. Answer: B - Stack
Stacks follow LIFO (e.g., push and pop operations), while queues follow FIFO.

Q31. Answer: B - INNER JOIN with subject and marks
The JOIN is required to combine student details from students with mark details from scores. Option B correctly returns name, subject and marks.
Option A is valid only for names, but it omits subject and marks, so it does not fully satisfy this question.

Q32. Answer: B - let url = URL(string: "https://api.example.com/products")!
Option B correctly uses asynchronous networking in Swift.
Option A incorrectly tries to fetch data synchronously.
Option C does not exist in Swift.
Option D uses a non-existent method.

Q33. Answer: A - import pandas as pd
Option A correctly loads a CSV file using Pandas.
Option B does not work since np.load() is for NumPy files, not CSVs.
Option C uses a non-existent function in Scikit-learn.
Option D misuses TensorFlow's Dataset API.

Q34. Answer: B - if (password_verify($_POST['password'], $hashed_password)) {
Option B is correct because password_verify() compares a plain text password with a securely hashed password.
Option A is incorrect because passwords should never be stored in plain text.

Option C does not work because hash_equals() is for comparing two hashes, not a password against a hash.
Option D is insecure because sha1() is outdated and not recommended.

Q35. Answer: B - SELECT user_id, SUM(amount) FROM transactions WHERE status = 'success'

Option B is correct because it groups transactions per user and calculates their total success amount.
Option A is incorrect because GROUP BY is missing.
Option C only calculates the total amount but not per user.
Option D retrieves individual transactions instead of aggregation.

Q36. Answer: A - Use students.sort { $0.gpa > $1.gpa } on a mutable array.

Swift's sorted() returns a new sorted array, while sort() sorts a mutable array in place. For large collections, in-place sorting can reduce extra memory usage when mutating the original array is acceptable. Lazy evaluation does not avoid the need to materialize a fully sorted order.

Q37. Answer: C - Use a prepared statement with mysqli_prepare().

Prepared statements separate SQL logic from data, preventing injection. While mysqli_real_escape_string (A) helps, it's error-prone. htmlspecialchars (D) prevents XSS but not SQL injection.

Q38. Answer: C - The completion block captures self strongly.

The block retains self (the ViewController) strongly, creating a retain cycle. Fix: Use __weak typeof(self) weakSelf = self; inside the block.

Q39. Answer: A - how="outer" includes unmatched rows.

An outer join includes all rows from both DataFrames, leading to NaN in unmatched rows. Use how="inner" to exclude them. Option B is unrelated to the merge logic.

Q40. Answer: C - print() isn't synchronized.

Multiple threads calling print() simultaneously can interleave output. Use a lock to synchronize the print statement. The decorator itself is thread-safe (A is incorrect).

Q41. Answer: A - import pandas as pd

Option A is correct because it first sorts by rating and views and then selects the top 5.
Option B is incorrect because np.loadtxt() is not suitable for CSVs with text values.
Option C filters movies with a rating above 8 but does not sort properly.
Option D filters by views but ignores rating, which is also important.

Q42. Answer: C - $password = $_POST['password'];

Option C is correct because it fetches the stored hash and verifies it using password_verify().
Option A is insecure because it stores passwords in plaintext.
Option B is incorrect because it hashes the password incorrectly before checking.
Option D uses md5(), which is outdated and insecure.

Q43. Answer: A - import Foundation

Option A is correct because WebSockets use the wss:// protocol and webSocketTask(with:) is the correct API.
Option B incorrectly uses WKWebView, which is for web browsing, not WebSockets.
Option C incorrectly uses streamTask, which is not WebSocket-compatible.
Option D uses https:// instead of wss://, making it invalid for real-time communication.

Q44. Answer: A - import pandas as pd

Option A is correct because groupby('customer_id')['amount'].mean() calculates the average amount per customer.
Option B is incorrect because np.load() does not support CSVs.
Option C calculates the overall average but not per customer.
Option D is incorrect because it just plots data instead of analyzing it.

Q45. Answer: C - Use TfidfVectorizer on symptoms and train a classifier.

Textual symptom descriptions must be converted into numeric features before most machine-learning classifiers can use them. TF-IDF vectorization is a standard introductory method for representing text. The classifier can then learn patterns between symptoms and diagnosis labels.

Q46. Answer: C - Correlated subquery with amount, failed status, 24-hour window and failed-count condition

The rule has four conditions: high amount, failed status, last 24 hours, and more than 3 failed transactions by the same user in that window. Option C combines all conditions using a correlated subquery for the user-level failed-count check.

Q47. Answer: B - import pandas as pd

Option B is correct because Logistic Regression is best for categorical predictions (grades A, B, C, etc.).
Option A only calculates correlation but does not predict grades.
Option C incorrectly uses clustering (KMeans) instead of classification.
Option D incorrectly uses a linear formula, missing the categorical classification requirement.

Q48. Answer: B - Train a KNeighborsClassifier using purchase_count and avg_rating.

K-nearest neighbors is a reasonable introductory recommendation-style classifier when numeric user-behavior features are available. Option C is invalid as written because a text feature such as price_range must be encoded before many scikit-learn estimators can process it.

Q49. Answer: C - Prepared statement + password_verify()

A secure login flow separates SQL code from user input by using a prepared statement. It then checks the submitted password against the stored hash using password_verify(). MD5 and string-concatenated SQL are not suitable for secure authentication.

Q50. Answer: B - let totalSales = sales.reduce(0) { $0 + ($1["amount"] as! Int) }

Option B is correct because reduce(0) properly sums up the sales values.
Option A is incorrect because Swift dictionaries require type casting, making it invalid.
Option C is incorrect because map() returns an array, and .sum() is not a valid Swift function.
Option D is incorrect because it only returns the number of sales entries, not their total amount.