School Connect Online

SCO INTERNATIONAL OLYMPIAD

GRADE 9 CODING OLYMPIAD

Official question paper for SCO International Coding Olympiad

Designed from Class 9 coding syllabus pathways and aligned with global computational-thinking expectations

- 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 A - Official Rebranded 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.

Which of the following is a correct function definition in Python that returns the square of a number?

A) def square(x): return x * x
B) function square(x) { return x * x; }
C) def square(x) { return x ** 2 }
D) square(x) = x * x

Q2.

What is the time complexity of binary search on a sorted array of n elements?

A) O(1)
B) O(n)
C) O(log n)
D) $O(n^2)$

Q3.

In a RESTful web service, which HTTP method is typically used for retrieving data from the server?

A) GET
B) POST
C) PUT
D) DELETE

Q4.

Swift Programming Language - Constants:

In Swift, which keyword is used to declare a constant that cannot be changed once set?

A) var
B) let
C) const
D) static

Q5.

Which syntax correctly calls a method named calculate on an Objective-C object named calculator?

A) calculator.calculate()
B) [calculator calculate]
C) calculator->calculate()
D) calculate(calculator)

Q6.

In PHP, which operator is used to concatenate (join) two strings?

A) +
B) .
C) &
D) %

Q7.

Which SQL clause is used to filter rows based on a specified condition?

A) GROUP BY

B) ORDER BY
C) WHERE
D) HAVING

Q8.
What is the primary purpose of a decorator in Python?
A) To define a new class
B) To modify or extend the behavior of a function or method without changing its code
C) To convert a list into a dictionary
D) To handle exceptions automatically

Q9.
Which Python library is most commonly used for data manipulation and analysis?
A) NumPy
B) pandas
C) Matplotlib
D) SciPy

Q10.
Which function from Python's statistics module is used to calculate the median of a list of numbers?
A) statistics.mean()
B) statistics.median()
C) statistics.mode()
D) statistics.stdev()

Q11.
What will be the output of the following Python function?

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

A) 120
B) 25
C) 15
D) 5

Q12.
Which of the following sorting algorithms has an average time complexity of O(n log n) but worst-case complexity of $O(n^2)$?
A) Merge Sort
B) Quick Sort
C) Insertion Sort
D) Bubble Sort

Q13.
In web development, which HTTP status code indicates that the client must authenticate itself to get the requested response?
A) 200
B) 301
C) 401

D) 500

Q14.

What is the purpose of optionals in Swift?

A) To store multiple values in a single variable
B) To indicate that a variable can hold a value or be nil
C) To optimize memory allocation
D) To improve performance of loops

Q15.

Which method is used in Objective-C for manual memory management to release an allocated object?

A) dealloc
B) release
C) free
D) removeMemory

Q16.

Which function should be used in PHP to prevent SQL injection?

A) escape_sql()
B) mysql_real_escape_string()
C) strip_tags()
D) prepare()

Q17.

Which SQL keyword normally begins a subquery that returns data inside another SQL statement?

A) SELECT
B) HAVING
C) GROUP BY
D) ORDER BY

Q18.

What will be the output of the following Python code?

```
x = lambda a, b : a + b
print(x(5, 10))
```

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

Q19.

Which of the following is the correct way to select a single column named age from a Pandas DataFrame df?

A) df.age
B) df['age']
C) df.loc[:, 'age']
D) All of the above

Q20.

What does a high standard deviation in a dataset indicate?

A) The data points are close to the mean

B) The data points are spread out over a large range of values
C) There are outliers in the dataset
D) The dataset is skewed

Section 2: Error Analysis and Debugging

Q21.

A developer implements a caching decorator to optimize a recursive Fibonacci function. However, a similar function is later written without proper recursion decrement:

```
def factorial(n):
    if n == 0:
        return 1
    else:
        return n * factorial(n)  # Error: Should decrement n
print(factorial(5))
```

What error occurs in this code?

A) It prints 120 correctly.
B) It results in infinite recursion and eventually a RecursionError.
C) It returns 0 because n becomes 0.
D) It raises a TypeError.

Q22.

Consider the following code intended to compute squares for numbers greater than 2:

```
numbers = [1, 2, 3, 4, 5]
squares = [x**2 for x in numbers if y > 2]
print(squares)
```

What is the error in this code?

A) The exponent operator is used incorrectly.
B) The variable y in the condition is not defined.
C) The list comprehension syntax is completely invalid.
D) The condition should be written using a function.

Q23.

Examine the following C code:

```
#include <stdio.h>
int main() {
    int arr[4] = {10, 20, 30, 40};
    for (int i = 0; i <= 4; i++) {
        printf("%d ", arr[i]);
    }
    return 0;
}
```

What error does this code contain?

A) The array is declared incorrectly.
B) The loop condition i <= 4 causes an out-of-bounds access.
C) The printf format specifier is wrong.
D) There is no error; it correctly prints all elements.

Q24.

A PHP developer writes this code to retrieve user data from a database:

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

What is the primary security issue with this code?

A) It causes a syntax error.
B) It is vulnerable to SQL injection.
C) It is too slow to execute.
D) It does not return any data.

Q25.

Consider the Swift code below:

```
var greeting: String? = nil
print(greeting!)
```

What is the error, and why does it occur?

A) It prints "nil" correctly.
B) It crashes at runtime due to force unwrapping a nil optional.
C) It compiles but prints an empty string.
D) It gives a compile-time error due to incorrect syntax.

Q26.

A developer mistakenly writes the following Objective-C code:

```
#import <Foundation/Foundation.h>
@interface Person : NSObject
@property (nonatomic, strong) NSString *name;
@end
@implementation Person
@end
int main(int argc, const char * argv[]) {
    @autoreleasepool {
        Person *p = [Person init]; // Error: incorrect order of method calls
        p.name = @"Alice";
        NSLog(@"%@", p.name);
    }
    return 0;
}
```

What is the error in this code?

A) The property declaration is incorrect.
B) The object is not allocated before calling init.
C) The code uses NSLog incorrectly.
D) There is no error; it works fine.

Q27.

Consider this SQL query intended to retrieve the average salary per department:

```
SELECT department, AVG(salary)
FROM Employees
WHERE salary > 50000;
```

What is missing in the query that would cause an error or unintended behavior?

A) The query should use GROUP BY department.
B) The WHERE clause is incorrect.

C) The AVG function should be replaced with SUM.
D) The query needs an ORDER BY clause.

Q28.

Examine the following code:

```
def add_element(element, lst=[]):
    lst.append(element)
    return lst
print(add_element(1))
print(add_element(2))
```

What unexpected behavior occurs, and why?

A) It prints [1] then [2] because the default list is recreated every time.
B) It prints [1] then [1, 2] because the default list is shared across function calls.
C) It prints an error because mutable default arguments are not allowed.
D) It prints [1, 2] both times.

Q29.

A data scientist writes the following code using Pandas:

```
import pandas as pd
df = pd.DataFrame({'A': [1, 2, 3], 'B': [4, 5, 6]})
result = df[df['C'] > 3]
print(result)
```

What error occurs, and why?

A) The code runs correctly and returns an empty DataFrame.
B) A KeyError occurs because column 'C' does not exist.
C) A ValueError occurs due to mismatched data types.
D) The DataFrame is not printed because the syntax is incorrect.

Q30.

Consider the following C code:

```
#include <stdio.h>
int main() {
    int *p;
    *p = 10;
    printf("%d\n", *p);
    return 0;
}
```

What is the error in this code?

A) The pointer p is uninitialized, leading to undefined behavior.
B) The multiplication operator is used incorrectly.
C) The printf statement is missing a format specifier.
D) The code compiles and runs correctly.

Section 3: Advanced Runtime and Code Reasoning

Q31.

Consider the following code:

```
nums = [1, 2, 3, 4, 5]
doubled = [x * 2 for x in nums if y > 2]
print(doubled)
```

What is the error in this code?

A) The list comprehension syntax is invalid.
B) The variable y is undefined.
C) The multiplication operator is used incorrectly.
D) There is no error; it prints the doubled values.

Q32.

Examine the following function:

```
def add_item(item, items=[]):
    items.append(item)
    return items
print(add_item("apple"))
print(add_item("banana"))
```

What unexpected behavior will occur?

A) It prints ["apple"] then ["banana"].
B) It prints ["apple"] then ["apple", "banana"].
C) It prints an error due to mutable default arguments.
D) It always returns an empty list.

Q33.

Review this C code:

```
#include <stdio.h>
int main() {
    int arr[3] = {1, 2, 3};
    for (int i = 0; i <= 3; i++) {
        printf("%d ", arr[i]);
    }
    return 0;
}
```

What is the error in this code?

A) The array declaration is incorrect.
B) The loop condition i <= 3 accesses an out-of-bounds index.
C) The printf format specifier is wrong.
D) There is no error; the code prints all array elements correctly.

Q34.

Examine the following C code:

```
#include <stdio.h>
int main() {
    int *p;
    *p = 10;
    printf("%d\n", *p);
    return 0;
}
```

What is the error in this code?

A) The pointer p is uninitialized before dereferencing.

B) The multiplication operator is used incorrectly.

C) The printf statement is missing a newline character.

D) There is no error; the code runs correctly.

Q35.

Consider this SQL query:

```
SELECT department, COUNT(*)
FROM employees
WHERE salary > 50000;
```

What is the error or issue with this query?

A) The query will return an error due to missing GROUP BY clause for the non-aggregated column department.

B) The WHERE clause should be replaced with HAVING.

C) The COUNT(*) function is used incorrectly.

D) There is no error; the query executes correctly.

Q36.

Review the following PHP code:

```
<?php
$greeting = "Hello";
$name = "World";
echo $greeting + $name;
?>
```

What is the error in this code?

A) PHP does not support string variables.

B) The plus operator is incorrectly used; string concatenation requires the dot operator.

C) The echo statement is missing a semicolon.

D) The code is correct and prints "HelloWorld".

Q37.

Consider the following Swift code:

```
var value: Int? = nil
print(value!)
```

What error occurs when this code is executed?

A) It prints "nil".

B) It crashes at runtime due to force unwrapping a nil optional.

C) It prints an empty string.

D) It returns 0 without error.

Q38.

Consider the following Objective-C code snippet:

```
#import <Foundation/Foundation.h>
@interface Person : NSObject
@property (nonatomic, strong) NSString *name;
@end
@implementation Person
@end
int main(int argc, const char * argv[]) {
    @autoreleasepool {
```

```
        Person *p = [Person init];
        p.name = @"Alice";
        NSLog(@"%@", p.name);
    }
    return 0;
}
```

What is the error in this code?

A) The property declaration is incorrect.
B) The object is not allocated with alloc before calling init.
C) NSLog is used incorrectly.
D) There is no error; it prints "Alice".

Q39.

Consider the following JavaScript code intended to fetch data asynchronously:

```
javascript
function fetchData(callback) {
    setTimeout(() => {
        let data = "Hello, World!";
        // Missing callback invocation
    }, 1000);
}
fetchData((data) => {
    console.log(data);
});
```

What is the error in this code?

A) The function setTimeout is used incorrectly.
B) The callback function is never invoked, so nothing is logged.
C) The arrow function syntax is incorrect.
D) The code will throw a syntax error due to missing semicolons.

Q40.

Examine the following Python function intended to compute Fibonacci numbers:

```
def fib(n):
    if n == 1:
        return 1
    return fib(n-1) + fib(n-2)
print(fib(5))
```

What is the issue with this recursive function?

A) It correctly computes Fibonacci numbers.
B) The base case is incomplete; it should check for n == 0 as well to avoid incorrect results or infinite recursion.
C) It uses addition incorrectly.
D) The recursion depth is too high for n=5.

Section 4: Achievers Section - Applied Coding Scenarios

Q41.

Consider the following code snippet:

```
numbers = [10, 20, 30, 40]
result = [x for x in numbers if y > 15]
print(result)
```

What is the error in this code?

A) The list comprehension syntax is invalid.

B) The variable y is not defined.

C) The condition should be if x > 15.

D) Both B and C.

Q42.

Examine the following function:

```
def append_item(item, item_list=[]):
    item_list.append(item)
    return item_list
print(append_item(1))
print(append_item(2))
```

What unexpected behavior occurs, and why?

A) It prints [1] then [2] because a new list is created each time.

B) It prints [1] then [1, 2] because the same default list is reused.

C) It raises a TypeError due to mutable default arguments.

D) It returns an empty list every time.

Q43.

Consider this C code:

```
#include <stdio.h>
int main() {
    int arr[3] = {5, 10, 15};
    for (int i = 0; i <= 3; i++) {
        printf("%d ", arr[i]);
    }
    return 0;
}
```

What is the error in this code?

A) The array declaration is incorrect.

B) The loop condition i <= 3 causes out-of-bounds access.

C) The printf statement has the wrong format specifier.

D) There is no error; the code prints correctly.

Q44.

Examine the following code:

```
#include <stdio.h>
int main() {
    int *p;
    *p = 20;
    printf("%d\n", *p);
    return 0;
}
```

What is the error in this code?

A) The pointer p is uninitialized before being dereferenced.

B) The code uses the wrong data type.

C) The pointer arithmetic is incorrect.

D) There is no error; the code prints 20.

Q45.

Consider the following PHP code:

```
<?php
$first = "Hello";
$second = "World";
echo $first + $second;
?>
```

What is the error in this code?

A) PHP does not support string variables.
B) The plus operator is used instead of the concatenation operator.
C) The echo statement is missing a semicolon.
D) There is no error; it outputs "HelloWorld".

Q46.

Consider this SQL query:

```
SELECT department, SUM(salary)
FROM Employees
WHERE salary > 60000;
```

What is the error in this query?

A) The WHERE clause is invalid.
B) The query is missing a GROUP BY clause for the department column.
C) SUM() is not a valid SQL function.
D) There is no error; it returns the total salary for all employees with salary > 60000.

Q47.

Consider the following Swift code:

```
var username: String? = nil
print(username!)
```

What error will this code produce?

A) It prints "nil".
B) It causes a runtime crash due to force unwrapping a nil value.
C) It compiles, but prints an empty string.
D) It prints "Optional(nil)".

Q48.

Consider the following JavaScript function intended to fetch data asynchronously:

```
function fetchData(callback) {
  setTimeout(() => {
    let data = "Data Loaded";
    // Missing callback invocation here
}, 1000);
}
fetchData(function(result) {
console.log(result);
});
```

What is the error in this code?

A) The function setTimeout is not used correctly.
B) The callback is never invoked, so nothing is logged.
C) The arrow function syntax is incorrect.
D) The function fetchData should return data directly.

Q49.

Review the following Objective-C snippet:

```
#import <Foundation/Foundation.h>
@interface Student : NSObject
@property (nonatomic, strong) NSString *name;
@end
@implementation Student
@end
int main(int argc, const char * argv[]) {
    @autoreleasepool {
        Student *s = [Student init]; // Error: Incorrect initialization
        s.name = @"Emma";
        NSLog(@"%@", s.name);
    }
    return 0;
}
```

What is the error in this code?

A) The property declaration is incorrect.

B) The object is not properly allocated before calling init.

C) The NSLog statement is missing formatting.

D) There is no error; the code works fine.

Q50.

Consider the following C++ code:

```
#include <iostream>
#include <vector>
using namespace std;
int main() {
    vector<int> numbers = {1, 2, 3, 4, 5};
    cout << numbers.at(5) << endl;
    return 0;
}
```

What error occurs in this code?

A) It prints 6 because vectors are 1-indexed.

B) It throws an out_of_range exception because index 5 is out-of-bounds.

C) It prints nothing due to a runtime error.

D) It compiles but returns 0.

School Connect Online

Answer Key

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

Detailed Explanations

Q1. Answer: A - def square(x): return x * x

Option A is the proper Python syntax for defining a function. It uses the def keyword, takes a parameter x, and returns x * x which is the square of x.

Q2. Answer: C - O(log n)

Binary search repeatedly divides the search interval in half, resulting in a time complexity of O(log n) for a sorted array.

Q3. Answer: A - GET

GET is used to request data from a specified resource without modifying it. POST is for creating new resources, PUT is for updating, and DELETE is for removal.

Q4. Answer: B - let

Swift uses the keyword let to declare a constant. Once assigned, its value cannot be modified. The keyword var is used for variables that can change.

Q5. Answer: B - [calculator calculate]

Objective-C uses square bracket notation for message passing. The correct syntax to call the method is [calculator calculate].

Q6. Answer: B - .
PHP uses the dot operator (.) for string concatenation. The plus operator is for numeric addition, while & and % serve different purposes.

Q7. Answer: C - WHERE
The WHERE clause is used to filter records that meet a specified condition. GROUP BY aggregates data, ORDER BY sorts the result set, and HAVING is used to filter groups after aggregation.

Q8. Answer: B - To modify or extend the behavior of a function or method without changing its code
Decorators in Python are used to wrap a function or method with additional functionality. They allow modification of behavior without altering the original function's code.

Q9. Answer: B - pandas
The pandas library is widely used for data manipulation and analysis, offering powerful data structures like DataFrame. NumPy is used for numerical operations, Matplotlib for plotting, and SciPy for scientific computations.

Q10. Answer: B - statistics.median()
The statistics.median() function computes the median value of a dataset. The mean gives the average, mode returns the most frequent value, and stdev computes the standard deviation.

Q11. Answer: A - 120
This function implements recursion to compute the factorial of n.
mystery(5) = 5 * 4 * 3 * 2 * 1 = 120.

Q12. Answer: B - Quick Sort
Quick Sort has an average-case time complexity of O(n log n), but in the worst case (if the pivot selection is poor), it degrades to $O(n^2)$.

Q13. Answer: C - 401
The 401 Unauthorized status code means authentication is required.

Q14. Answer: B - To indicate that a variable can hold a value or be nil
Optionals allow variables to hold either a value or nil, preventing runtime errors when accessing empty variables.

Q15. Answer: B - release
In manual reference counting (MRC), release decreases the reference count of an object. When it reaches 0, the object is deallocated.

Q16. Answer: D - prepare()
Using prepare() with parameterized queries in PHP's PDO (or mysqli_prepare()) helps prevent SQL injection.

Q17. Answer: A - SELECT

A subquery is itself a query nested inside another statement. It commonly begins with SELECT, for example: SELECT name FROM students WHERE marks = (SELECT MAX(marks) FROM students). WHERE, HAVING, GROUP BY and ORDER BY may appear around or inside subqueries, but SELECT begins the subquery that returns the data.

Q18. Answer: C - 15
The lambda function takes two arguments and returns their sum: 5 + 10 = 15.

Q19. Answer: D - All of the above
All three methods correctly access the column age in a Pandas DataFrame.

Q20. Answer: B - The data points are spread out over a large range of values
A high standard deviation means that the data points are spread out, indicating high variability.

Q21. Answer: B - It results in infinite recursion and eventually a RecursionError.
The function calls factorial(n) recursively without decrementing n, causing infinite recursion. Eventually, Python's recursion limit is reached, resulting in a RecursionError.

Q22. Answer: B - The variable y in the condition is not defined.
The condition if y > 2 uses an undefined variable y. The intended condition was likely if x > 2.

Q23. Answer: B - The loop condition i <= 4 causes an out-of-bounds access.
An array of size 4 has valid indices 0 to 3. The loop condition i <= 4 iterates for i=4, causing an out-of-bounds access, which leads to undefined behavior.

Q24. Answer: B - It is vulnerable to SQL injection.
The code concatenates user input directly into the SQL query without sanitization, making it vulnerable to SQL injection attacks. Using prepared statements would mitigate this risk.

Q25. Answer: B - It crashes at runtime due to force unwrapping a nil optional.
Force unwrapping an optional that is nil (using !) results in a runtime crash. The code should safely unwrap the optional using optional binding (e.g., if let) to avoid this error.

Q26. Answer: B - The object is not allocated before calling init.
The proper way to initialize an object in Objective-C is to first allocate memory with alloc and then initialize with init, i.e., [[Person alloc] init]. Here, the developer calls [Person init] directly, which is incorrect and leads to undefined behavior.

Q27. Answer: A - The query should use GROUP BY department.
When using an aggregate function like AVG along with a non-aggregated column (department), a GROUP BY clause is required. Without it, the query will either throw an error or produce unintended results.

Q28. Answer: B - It prints [1] then [1, 2] because the default list is shared across function calls.
Using a mutable default argument (lst=[]) causes the same list to be reused across function calls. The first call appends 1, and the second call appends 2 to the same list, resulting in [1] then [1, 2].

Q29. Answer: B - A KeyError occurs because column 'C' does not exist.
The code attempts to filter rows based on column 'C', which is not present in the DataFrame. This leads to a KeyError. The correct column name should be used.

Q30. Answer: A - The pointer p is uninitialized, leading to undefined behavior.
The pointer p is declared but not initialized to point to valid memory. Dereferencing an uninitialized pointer results in undefined behavior, which often causes a runtime crash.

Q31. Answer: B - The variable y is undefined.
In the list comprehension, the condition uses y > 2 but the variable y is never defined. Likely, the intended variable was x. This mistake leads to a NameError when the code is executed.

Q32. Answer: B - It prints ["apple"] then ["apple", "banana"].
Using a mutable default argument (items=[]) means that the same list is used across multiple function calls. After the first call, "apple" is stored; in the second call, "banana" is appended to the same list, resulting in ["apple", "banana"].

Q33. Answer: B - The loop condition i <= 3 accesses an out-of-bounds index.
An array of size 3 has valid indices 0, 1, and 2. The loop condition i <= 3 causes the loop to iterate when i is 3, which results in an out-of-bounds access and undefined behavior.

Q34. Answer: A - The pointer p is uninitialized before dereferencing.
The pointer p is declared but never initialized to a valid memory address. Dereferencing an uninitialized pointer leads to undefined behavior, often causing a segmentation fault or crash at runtime.

Q35. Answer: A - The query will return an error due to missing GROUP BY clause for the non-aggregated column department.
When using an aggregate function (like COUNT(*)) along with a non-aggregated column (department), a GROUP BY clause is required. Without it, SQL standards require that every column in the SELECT list be either aggregated or included in the GROUP BY clause.

Q36. Answer: B - The plus operator is incorrectly used; string concatenation requires the dot operator.
In PHP, the plus operator (+) is used for numeric addition. To concatenate strings, the dot operator (.) must be used. The correct code should be echo $greeting . $name;.

Q37. Answer: B - It crashes at runtime due to force unwrapping a nil optional.
Force unwrapping (!) an optional that contains nil will cause a runtime crash in Swift. Safe unwrapping (using if let or guard let) should be used to avoid such crashes.

Q38. Answer: B - The object is not allocated with alloc before calling init.
In Objective-C, you must allocate an object before initializing it. The correct syntax is [[Person alloc] init]. Here, [Person init] is incorrect and results in undefined behavior.

Q39. Answer: B - The callback function is never invoked, so nothing is logged.
Inside the setTimeout callback, the developer forgot to call callback(data). As a result, the callback passed to fetchData is never executed, and no output is produced.

Q40. Answer: B - The base case is incomplete; it should check for n == 0 as well to avoid incorrect results or infinite recursion.

The Fibonacci sequence requires two base cases: typically, fib(0) = 0 and fib(1) = 1. This function only handles the case when n == 1. Without a base case for n == 0, the recursion may not work as intended, and in some cases, it could lead to infinite recursion. The function should include:

```
if n == 0:
   return 0
if n == 1:
   return 1
```

Q41. Answer: D - Both B and C.

The code incorrectly uses the variable y in the condition, which is not defined. The intended condition was likely if x > 15. Therefore, the error is due to an undefined variable and the wrong condition.

Q42. Answer: B - It prints [1] then [1, 2] because the same default list is reused.

The function uses a mutable default argument (item_list=[]), so the same list is shared across function calls. The first call appends 1 and returns [1], and the second call appends 2 to the same list, resulting in [1, 2].

Q43. Answer: B - The loop condition i <= 3 causes out-of-bounds access.

An array of size 3 has valid indices 0, 1, and 2. The loop condition i <= 3 iterates when i is 3, which is out of bounds. The loop should use i < 3.

Q44. Answer: A - The pointer p is uninitialized before being dereferenced.

The pointer p is declared but not initialized to point to a valid memory location. Dereferencing an uninitialized pointer leads to undefined behavior and often a crash.

Q45. Answer: B - The plus operator is used instead of the concatenation operator.

In PHP, the plus operator (+) is used for numeric addition. To concatenate strings, the dot operator (.) must be used. The correct code is echo $first . $second;.

Q46. Answer: B - The query is missing a GROUP BY clause for the department column.

When using aggregate functions (like SUM) alongside a non-aggregated column (department), a GROUP BY clause is necessary to group results by that column. Without it, the query will produce an error or unintended results.

Q47. Answer: B - It causes a runtime crash due to force unwrapping a nil value.

Force unwrapping an optional that is nil (using !) leads to a runtime crash. To avoid this, optional binding should be used to safely unwrap the value.

Q48. Answer: B - The callback is never invoked, so nothing is logged.

Inside the setTimeout callback, the developer failed to call callback(data). Therefore, the provided callback is never executed, and no output is logged.

Q49. Answer: B - The object is not properly allocated before calling init.

In Objective-C, objects must first be allocated using alloc before being initialized with init. The correct initialization is [[Student alloc] init].

Q50. Answer: B - It throws an out_of_range exception because index 5 is out-of-bounds.
C++ vectors are zero-indexed; valid indices are 0 to 4 for a vector of size 5. Using numbers.at(5) throws an out_of_range exception because index 5 is invalid.