School Connect Online

SCO INTERNATIONAL CODING OLYMPIAD

CLASS 8 QUESTION PAPER

Official Question Paper Set A | Rebranded 2026-27

Official SCO cover format with academic, PDF-ready question layout.

- Designed for Class 8 learners developing programming, debugging, web, data and game logic skills.
- Compact question numbering, clean question blocks, answer key and explanations for website download.
- Aligned to middle-school computational thinking, C++/Kotlin/Python/XML, web basics, game logic and responsible coding practice.

Programming	C++	Kotlin	Python	Turtle	
Scratch	JavaScript	SQL	Game Logic	SCO	

SCO International Coding Olympiad | Duration: 60 minutes | Type: Objective MCQ | Total Questions: 50

Candidate Field	To be completed by student
Name	.....
Registration ID	.....
School / Organization	.....

School Connect Online

Section 1 - Programming Basics

Q1. Consider the following Python code snippet:

```
x = 5
y = x + 3
print(y)
```

What will be printed, and what is the role of variables in this code?

A. `8; Variables store values that can be used in expressions.`
B. `53; Variables concatenate numbers.`
C. `5; The variable y is not updated.`
D. An error occurs because variables are not declared.

Q2. What is the output of the following Python code?

```
total = 0
for i in range(1, 5):
total += i
print(total)
```

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

Q3. What does the following code output?

```
num = 7
if num % 2 == 0:
print("Even")
else:
print("Odd")
```

A. Even
B. Odd
C. 7
D. Error

Q4. Consider the code:

```
arr = [10, 20, 30, 40]
print(arr[2])
```

What is printed, and what does this illustrate about arrays (lists) in Python?

A. `20; Arrays are 1-indexed.`
B. `30; Arrays in Python are 0-indexed.`
C. `40; It accesses the last element.`
D. `Error; Array indexing is invalid.`

Q5. Which of the following is the correct way to declare and initialize an integer variable in C++?

A. `int x = 10;`
B. `var x = 10;`
C. `int x: 10;`
D. `x = 10;`

Q6. What is the output of the following C++ code snippet?

```
#include <iostream>
using namespace std;
int main() {
for (int i = 0; i < 3; i++) {
cout << i << " ";
}
return 0;
}
```

A. 0 1 2
B. 1 2 3
C. 0 1 2 3
D. 1 2

Q7. Consider the following C++ code:

```
#include <iostream>
using namespace std;
int main() {
int arr[3] = {5, 10, 15};
cout << arr[3] << endl;
return 0;
}
```

What is the error in this code?

A. The array declaration is invalid.
B. The index 3 is out-of-bounds because the valid indices are 0 to 2.
C. The code will print 15.
D. There is no error; it prints correctly.

Q8. Which of the following is a correct way to declare a variable in Kotlin using type inference?

A. `var name: String = "Alice"`
B. `val age = 15`
C. `var isStudent: Boolean = true`
D. All of the above

Q9. What is the output of the following Kotlin code?

```
for (i in 1..3) {
print(i * 2)
}
```

A. 246
B. 2 4 6
C. 8
D. 2 4 6 with spaces

Q10. In Scratch programming, what is a sprite?

A. A background image
B. A graphical object that can be programmed to move, interact, and perform actions
C. A type of sound effect
D. A text block for instructions

Q11. Consider the following Kotlin code:

```
var name: String? = null
println(name!!.length)
```

What error will occur, and why?

A. It prints 0 because the length of null is 0.

B. It prints null.

C. It throws a NullPointerException due to force unwrapping a null value.

D. It runs successfully with output "0".

Q12. Given the following C++ code, what is the output?

```
#include <iostream>
using namespace std;
int add(int a, int b) { return a + b; }
double add(double a, double b) { return a + b; }
int main() {
cout << add(3, 4) << " " << add(3.5, 4.5);
return 0;
}
```

A. 7 8.0

B. 7 7.0

C. 34 78

D. Error due to ambiguity

Q13. What is wrong with the following PHP code?

```
<?php
echo "Hello World;
?>
```

A. Missing closing double quote for the string.

B. Missing semicolon after the echo statement.

C. PHP does not support echo statements.

D. There is no error.

Q14. Given two tables, Orders and Customers, both containing a column id, what is the issue with this SQL query?

```
SELECT id, order_date
FROM Orders
JOIN Customers ON Orders.customer_id = Customers.id;
```

A. There is no issue; it works correctly.

B. The column id is ambiguous; it must be qualified with the table name.

C. The JOIN syntax is incorrect.

D. The query should use WHERE instead of JOIN.

Q15. Consider the following code:

```
dict1 = {'a': 1, 'b': 2}
dict2 = {'b': 3, 'c': 4}
merged = {key: dict1[key] + dict2[key] for key in dict1}
print(merged)
```

What is the error, and how can it be corrected to merge only common keys?

A. KeyError for 'c'; correct by iterating over common keys using if key in dict2.

B. Syntax error in dictionary comprehension.

C. It prints the wrong result; correct by using update().

D. There is no error; it merges dictionaries correctly.

Q16. What will be the output of the following C++ code after sorting the array in ascending order?

```
#include <iostream>
using namespace std;
int main() {
int arr[] = {64, 25, 12, 22, 11};
int n = sizeof(arr)/sizeof(arr[0]);
for (int i = 0; i < n-1; i++) {
int min_idx = i;
for (int j = i+1; j < n; j++) {
if (arr[j] < arr[min_idx])
min_idx = j;
}
int temp = arr[i];
arr[i] = arr[min_idx];
arr[min_idx] = temp;
}
for (int i = 0; i < n; i++)
cout << arr[i] << " ";
return 0;
}
```

A. 11 12 22 25 64

B. 64 25 22 12 11

C. 11 12 25 22 64

D. 11 22 12 25 64

Q17. Examine the following Kotlin code:

```
var number: Int? = null
println(number!! + 5)
```

What error occurs, and why?

A. It prints 5, because null is treated as 0.

B. It causes a runtime exception because force unwrapping a null value using !! throws an exception.

C. It prints "null5".

D. It results in a compile-time error.

Q18. Consider this JavaScript code intended to change the text of an HTML element:

```
document.getElementById("header").innerHTML = "Welcome!";
```

If the element with id "header" does not exist in the HTML, what happens?

A. It changes the text of the first element automatically.

B. A runtime error can occur because getElementById returns null and innerHTML is accessed on null.

C. It creates a new element with id "header".

D. It silently changes the page title.

Q19. What is the error in the following Python code intended to square only even numbers?

numbers = [1, 2, 3, 4, 5]

```
result = list(map(lambda x: x**2 if x % 2 == 0, numbers))
print(result)
```

A. The lambda syntax is incorrect; it should use a full if-else expression.

B. The map function cannot be used with lambda functions.

C. There is no error; it correctly squares even numbers.

D. The code will output None for odd numbers.

Q20. Consider the following SQL query intended to calculate the average salary per department:

```
SELECT department, AVG(salary) AS avg_salary
FROM employees
WHERE salary > 50000
GROUP BY department;
```

Which statement is correct about this query under standard SQL?

A. AVG(salary) is invalid because salary is not in GROUP BY.

B. The alias avg_salary causes an error.

C. The query is valid because department is grouped and salary is used inside an aggregate function.

D. ORDER BY is mandatory whenever GROUP BY is used.

Q21. Scenario: A developer writes a recursive function to compute the nth Fibonacci number but omits a necessary base case:

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

What is the error in this code?

A. It correctly computes fib(5).

B. It may cause infinite recursion because the base case for n == 0 is missing.

C. The recursion is too deep for n=5.

D. The addition operator is used incorrectly.

Q22. Scenario: Consider the following code intended to square numbers that are even:

nums = [1, 2, 3, 4, 5]

```
result = list(map(lambda x: x**2 if x % 2 == 0, nums))
print(result)
```

What is the error in this code?

A. The lambda syntax is incorrect because it lacks an else clause.

B. The map function cannot be used with conditionals.

C. The code will output all numbers squared.

D. There is no error; it works correctly.

Q23. A programmer tries to merge two dictionaries by summing values for common keys:

```
dict1 = {'a': 1, 'b': 2, 'c': 3}
dict2 = {'b': 4, 'c': 5, 'd': 6}
merged = {key: dict1[key] + dict2[key] for key in dict1}
print(merged)
```

What error can occur, and how should it be corrected?

A. A KeyError occurs for key "a" because it is not present in dict2; iterate only over common keys.

B. A syntax error occurs because dictionary comprehensions are not allowed.

C. It always merges dictionaries correctly without any condition.

D. The error occurs because dictionaries cannot store numeric values.

Q24. A student intends to write a generator that yields numbers from 0 to n - 1:

```
def generate_numbers(n):
for i in range(n):
return i
print(list(generate_numbers(5)))
```

What is the error?

A. It outputs [0, 1, 2, 3, 4]; the code is correct.

B. It raises a TypeError because the function returns an integer instead of an iterable generator.

C. It outputs an empty list because range(5) is empty.

D. It causes a syntax error because return cannot be used inside a loop.

Q25. A developer writes code to compute the sum of numbers from 1 to n:

```
def sum_to_n(n):
```

total = 0

```
for i in range(1, n):
```

total += i

```
return total
print(sum_to_n(5))
What is the error, and what is the correct sum for n=5?
```

A. The loop should iterate to n+1; correct sum is 15.

B. The loop should iterate to n; correct sum is 15.

C. The loop iterates too far; correct sum is 10.

D. There is no error; the sum is 10.

Q26. A C program attempts to print the value of a dynamically allocated integer:

```
#include <stdio.h>
#include <stdlib.h>
int main() {
int *ptr;
*ptr = 100;
printf("%d\n", *ptr);
return 0;
}
```

What is the error in this code?

A. The pointer is not allocated memory before dereferencing.

B. The code has a syntax error in printf.

C. The value 100 cannot be assigned to a pointer.

D. There is no error; the code prints 100.

Q27. Consider the following C++ code:

```
#include <iostream>
using namespace std;
void display(int &x) { cout << "Reference: " << x << endl; }
void display(const int &x) { cout << "Const Reference: " << x << endl; }
int main() {
int a = 5;
display(a);
return 0;
}
```

What happens when display(a) is called?

A. Compilation fails due to ambiguity.

B. The const reference overload is always chosen.

C. There is no error; the non-const reference overload is selected for the non-const lvalue a.

D. The program prints nothing.

Q28. A PHP script is intended to check if a user is an admin:

```
<?php
$user_role = "editor";
if ($user_role = "admin") {
echo "Access granted";
} else {
echo "Access denied";
}
?>
```

What is the error, and what does the code output?

A. It outputs "Access denied" because the condition is false.

B. It outputs "Access granted" because the assignment operator is used instead of comparison.

C. It outputs an error due to syntax mistakes.

D. It outputs nothing because the script fails silently.

Q29. A query is written to get the total sales by region and product:

```
SELECT region, product, SUM(sales)
FROM Sales
GROUP BY region;
```

What is the error in this query?

A. The query is correct.

B. All selected columns not aggregated must be included in the GROUP BY clause.

C. The SUM function is used incorrectly.

D. The GROUP BY clause should come after the ORDER BY clause.

Q30. Consider the following SQL query intended to find employees earning above the average salary:

```
SELECT name, salary
FROM Employees
WHERE salary > (SELECT AVG(salary) FROM Employees)
```

What is the potential issue with this query?

A. The subquery does not require AVG.

B. There is no error; the query is correct.

C. The subquery should be in the HAVING clause.

D. The query will always return zero results.

Q31. A Swift developer writes the following code:

```
class Downloader {
var completion: (() -> Void)?
func startDownload() {
DispatchQueue.global().async {
sleep(2)
self.completion?()
}
}
}
```

What is the safest improvement when capturing self in asynchronous closures?

A. No improvement is ever needed.

B. Use a capture list such as [weak self] when the closure may keep the object alive longer than intended.

C. Replace sleep(2) with a for loop.

D. Remove the completion closure completely.

Q32. An Objective-C developer writes the following code:

```
NSDictionary *info = @{@"name": @"Alice", @"age": nil};
NSLog(@"%@", info);
```

What error will this produce?

A. It prints the dictionary correctly with nil values.

B. It produces a runtime error because nil cannot be a value in an NSDictionary.

C. It causes a compile-time error.

D. It prints an empty dictionary.

Q33. A Kotlin developer writes a loop to print numbers, but the syntax is incorrect:

```
for (i in 1...5) {
println(i)
}
```

What is the error?

A. The range operator is incorrect; it should be 1..5 instead of 1...5.
B. The in keyword is used incorrectly.
C. Kotlin does not support loops.
D. There is no error; the code prints numbers 1 to 5.

Q34. Consider the following Python code:

```
a = 256
b = 256
if a is b:
print("Equal")
else:
print("Not Equal")
```

What potential issue might arise with the use of is instead of ==?

A. It always prints "Equal" because integers are interned.
B. It may print "Not Equal" for larger integers because is checks identity, not equality.
C. It results in a syntax error.
D. There is no difference between is and == for integers.

Q35. A PHP developer writes a loop to print numbers from 1 to 5 but makes an error:

```
<?php
for ($i = 1; $i <= 5; $i++);
{
echo $i . " ";
}
?>
```

What is the error, and what is the output?

A. The semicolon after the for loop causes the loop body to be empty; it prints "6 ".
B. The loop is missing curly braces; it prints nothing.
C. The variable $i is not defined; it prints an error.
D. The loop is correct; it prints "1 2 3 4 5".

Q36. Consider the query intended to fetch employee names with salaries above the company average:

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

If the query returns an error about an ambiguous column, what might be the cause?

A. The subquery is missing a GROUP BY clause.
B. The column salary is ambiguous because it exists in both the outer query and subquery.
C. There is no error; the query is correct.
D. The query should use HAVING instead of WHERE.

Q37. A Python developer writes the following code but receives an IndentationError:

```
def greet(name):
print("Hello, " + name)
```

return
greet("Bob")
What is the error?

A. The function lacks a proper return value.
B. The print statement is not indented correctly inside the function.
C. The function name is reserved.
D. Python does not allow string concatenation.

Q38. Consider the following Swift code:

```
let value: Any = "123"
let number = value as! Int
print(number)
```

What error occurs, and why?

A. It prints 123 correctly.
B. It crashes at runtime because a string cannot be force-cast to an Int.
C. It results in a compile-time error due to incorrect syntax.
D. It prints "123" as a string.

Q39. A C++ developer writes overloaded functions:

```
#include <iostream>
using namespace std;
void func(int x) { cout << "Int: " << x << endl; }
void func(double x) { cout << "Double: " << x << endl; }
int main() {
func(5.0f);
return 0;
}
```

What is the output, and what is the reason behind it?

A. "Int: 5" because the float is converted to int.
B. "Double: 5" because the float is promoted to double.
C. Compilation error due to ambiguity.
D. "Int: 5.0" because of implicit conversion.

Q40. A developer writes the following SQL update query:

```
UPDATE employees
SET salary = salary * 1.1
WHERE id IN (SELECT id FROM employees WHERE performance = 'excellent');
Which improvement is most appropriate if the SQL dialect complains about referencing the
same table in the subquery?
```

A. Remove the WHERE clause and update every row.
B. Use string concatenation to build the query dynamically.
C. Rewrite using a supported JOIN or derived-table form with clear aliases.
D. Replace UPDATE with SELECT because salary cannot be modified.

School Connect Online

Achievers Section - Applied Programming and Game Logic

Q41. Which C++ statement correctly checks whether a score is at least 80 and prints "Pass"?

A. `if score >= 80 cout << "Pass";`
B. `if (score >= 80) { cout << "Pass"; }`
C. `if score => 80: print("Pass")`
D. `when score >= 80 print Pass`

Q42. A Kotlin variable is declared as val points = 10. What does val mean?

A. The variable can be changed many times.
B. The value is read-only after assignment.
C. The variable is always null.
D. The variable stores only decimal numbers.

Q43. Which Scratch event block is commonly used to start a program when the user clicks the green flag?

A. when green flag clicked
B. forever
C. move 10 steps
D. change score by 1

Q44. In JavaScript, which check should be done before changing innerHTML of an element returned by getElementById?

A. Check whether the returned element is not null.
B. Check whether the browser is in dark mode.
C. Check whether the page has a footer.
D. Check whether the file extension is .xml.

Q45. Which SQL query correctly counts the number of orders for each customer?

A. `SELECT customer_id, COUNT(*) FROM Orders GROUP BY customer_id;`
B. `SELECT customer_id, COUNT(*) FROM Orders;`
C. `SELECT COUNT(customer_id) WHERE Orders;`
D. `DELETE FROM Orders GROUP BY customer_id;`

Q46. Which Python expression correctly gives the common keys of two dictionaries d1 and d2?

A. d1 + d2
B. `d1.keys() & d2.keys()`
C. d1.common(d2)
D. d1 == d2

Q47. In a 2D game, why is a game loop important?

A. It repeatedly processes input, updates the game state and renders frames.
B. It stores only the final score.
C. It replaces all variables in a program.
D. It prevents all errors automatically.

Q48. Which method is generally efficient for detecting collision between simple rectangles in a 2D game?

A. AABB overlap checking
B. Sorting the score table
C. Changing the background color
D. Printing every pixel value

Q49. Which format is more compact and commonly used than XML for lightweight web data exchange?

A. JSON
B. BMP
C. EXE
D. WAV

Q50. Which pathfinding algorithm is widely used in games because it combines path cost with a heuristic estimate?

A. Bubble Sort
B. A* Algorithm
C. Linear Search
D. Selection Sort

School Connect Online

Answer Key

Q	Ans	Correct Option
1	A	8; Variables store values that can be used in expressions.
2	A	10
3	B	Odd
4	B	30; Arrays in Python are 0-indexed.
5	A	int x = 10;
6	A	0 1 2
7	B	The index 3 is out-of-bounds because the valid indices are 0 to 2.
8	D	All of the above
9	A	246
10	B	A graphical object that can be programmed to move, interact, and perform actions
11	C	It throws a NullPointerException due to force unwrapping a null value.
12	A	7 8.0
13	A	Missing closing double quote for the string.
14	B	The column id is ambiguous; it must be qualified with the table name.
15	A	KeyError for 'c'; correct by iterating over common keys using if key in dict2.
16	A	11 12 22 25 64
17	B	It causes a runtime exception because force unwrapping a null value using !! throws an exception.
18	B	A runtime error can occur because getElementById returns null and innerHTML is accessed on null.
19	A	The lambda syntax is incorrect; it should use a full if-else expression.
20	C	The query is valid because department is grouped and salary is used inside an aggregate function.
21	B	It may cause infinite recursion because the base case for n == 0 is missing.
22	A	The lambda syntax is incorrect because it lacks an else clause.
23	A	A KeyError occurs for key "a" because it is not present in dict2; iterate only over common keys.
24	B	It raises a TypeError because the function returns an integer instead of an iterable generator.
25	B	The loop should iterate to n; correct sum is 15.
26	A	The pointer is not allocated memory before dereferencing.
27	C	There is no error; the non-const reference overload is selected for the non-const lvalue a.
28	B	It outputs "Access granted" because the assignment operator is used instead of comparison.
29	B	All selected columns not aggregated must be included in the GROUP BY clause.
30	B	There is no error; the query is correct.
31	B	Use a capture list such as [weak self] when the closure may keep the object alive longer than intended.
32	B	It produces a runtime error because nil cannot be a value in an NSDictionary.
33	A	The range operator is incorrect; it should be 1..5 instead of 1...5.
34	B	It may print "Not Equal" for larger integers because is checks identity, not equality.
35	A	The semicolon after the for loop causes the loop body to be empty; it prints "6 ".
36	C	There is no error; the query is correct.
37	B	The print statement is not indented correctly inside the function.
38	B	It crashes at runtime because a string cannot be force-cast to an Int.
39	B	"Double: 5" because the float is promoted to double.
40	C	Rewrite using a supported JOIN or derived-table form with clear aliases.
41	B	if (score >= 80) { cout << "Pass"; }
42	B	The value is read-only after assignment.

Q	Ans	Correct Option
43	A	when green flag clicked
44	A	Check whether the returned element is not null.
45	A	SELECT customer_id, COUNT(*) FROM Orders GROUP BY customer_id;
46	B	d1.keys() & d2.keys()
47	A	It repeatedly processes input, updates the game state and renders frames.
48	A	AABB overlap checking
49	A	JSON
50	B	A* Algorithm

Detailed Explanations

Q1. Answer A: 8; Variables store values that can be used in expressions.

The variable x is assigned 5, and then y is assigned the result of x + 3 (which equals 8). Variables store values that can be used in expressions. Python does not require explicit declaration types.

Q2. Answer A: 10

The loop iterates for i = 1, 2, 3, 4. Their sum is 1+2+3+4 = 10.

Q3. Answer B: Odd

Since 7 % 2 equals 1, which is not 0, the condition is false, and the code prints "Odd".

Q4. Answer B: 30; Arrays in Python are 0-indexed.

Python lists are 0-indexed. arr[2] accesses the third element, which is 30.

Q5. Answer A: int x = 10;

In C++, variables are declared with their type. Option A correctly declares an integer variable x with an initial value of 10.

Q6. Answer A: 0 1 2

The loop runs for i = 0, 1, 2, printing these values with a space. The output is “0 1 2”.

Q7. Answer B: The index 3 is out-of-bounds because the valid indices are 0 to 2.

Arrays in C++ are 0-indexed, so an array of size 3 has valid indices 0, 1, and 2. Accessing arr[3] causes out-of-bounds behavior.

Q8. Answer D: All of the above

All options are correct Kotlin variable declarations. Kotlin supports explicit type declarations as in A and C, and type inference as in B. Both var (mutable) and val (immutable) can be used.

Q9. Answer A: 246

The loop iterates through 1, 2 and 3. The print function does not add spaces, so it prints 2, then 4, then 6 as 246.

Q10. Answer B: A graphical object that can be programmed to move, interact, and perform actions

A sprite in Scratch is a visual object that can be programmed to perform actions (e.g., move, respond to events, interact with other sprites). It is central to creating interactive stories and games in Scratch.

Q11. Answer C: It throws a NullPointerException due to force unwrapping a null value.

Using !! on a null value forces a NullPointerException. The correct approach is to safely unwrap the variable using a safe call (name?.length) or optional binding.

Q12. Answer A: 7 8.0

Function overloading allows multiple functions with the same name but different parameter types. The integer version returns 3+4 = 7 and the double version returns 3.5+4.5 = 8.0.

Q13. Answer A: Missing closing double quote for the string.

The string "Hello World is missing a closing double quote. The correct syntax is echo "Hello World";.

Q14. Answer B: The column id is ambiguous; it must be qualified with the table name.

Since both tables have a column named id, using SELECT id is ambiguous. The query should specify which table’s id is desired, e.g., SELECT Orders.id, order_date ....

Q15. Answer A: KeyError for 'c'; correct by iterating over common keys using if key in dict2.

The comprehension iterates over keys in dict1 and tries to add dict2[key] without checking if key exists in dict2. Adding a condition if key in dict2 fixes the issue, merging only common keys.

Q16. Answer A: 11 12 22 25 64

This code implements selection sort. After sorting, the array is [11, 12, 22, 25, 64]. The inner loop finds the minimum element from the unsorted portion, and then swaps it with the element at the current index.

Q17. Answer B: It causes a runtime exception because force unwrapping a null value using !! throws an exception.

The !! operator forcefully unwraps the nullable variable. Since number is null, it results in a runtime exception (NullPointerException). The correct approach is to use safe calls or provide a default value.

Q18. Answer B: A runtime error can occur because getElementById returns null and innerHTML is accessed on null.

When no matching element exists, document.getElementById("header") returns null. Accessing innerHTML on null can throw a runtime error unless the value is checked first.

Q19. Answer A: The lambda syntax is incorrect; it should use a full if-else expression.

A conditional expression in a lambda must include both the "if" and "else" parts. The correct syntax is:

lambda x: x**2 if x % 2 == 0 else x
Without the "else" part, the lambda expression is invalid.

Q20. Answer C: The query is valid because department is grouped and salary is used inside an aggregate function.

In standard SQL, non-aggregated columns in the SELECT list must appear in GROUP BY. Here department is grouped, while salary is aggregated with AVG, so the query is valid.

Q21. Answer B: It may cause infinite recursion because the base case for n == 0 is missing.

A proper Fibonacci function needs two base cases: fib(0)=0 and fib(1)=1. Without the base case for n == 0, the function will call fib(-1) eventually, leading to infinite recursion or incorrect results.

Q22. Answer A: The lambda syntax is incorrect because it lacks an else clause.

In Python, a conditional expression in a lambda must include both "if" and "else". The correct lambda should be:

lambda x: x**2 if x % 2 == 0 else x
Without the else part, the syntax is invalid.

Q23. Answer A: A KeyError occurs for key "a" because it is not present in dict2; iterate only over common keys.

The comprehension iterates over all keys in dict1. When it reaches key "a", dict2["a"] does not exist, so a KeyError occurs. Use common keys such as dict1.keys() & dict2.keys().

Q24. Answer B: It raises a TypeError because the function returns an integer instead of an iterable generator.

The function returns 0 during the first loop iteration. Because it returns an integer, list(generate_numbers(5)) tries to make a list from an integer and raises a TypeError. A generator should use yield i instead of return i.

Q25. Answer B: The loop should iterate to n; correct sum is 15.

The range(1, n) iterates from 1 to n-1. For n=5, it sums 1+2+3+4=10, which is incorrect if we intend to sum numbers from 1 to 5. The correct range is range(1, n+1) to include 5. The correct sum should be 1+2+3+4+5 = 15.

Q26. Answer A: The pointer is not allocated memory before dereferencing.

The pointer ptr is declared but not allocated memory (e.g., using malloc). Dereferencing an uninitialized pointer results in undefined behavior, likely causing a crash.

Q27. Answer C: There is no error; the non-const reference overload is selected for the non-const lvalue a.

A non-const lvalue int can bind directly to int&, so the non-const reference overload is the better match and is selected.

Q28. Answer B: It outputs "Access granted" because the assignment operator is used instead of comparison.

The code mistakenly uses the assignment operator (=) instead of the equality operator (== or ===). As a result, $user_role is set to "admin", making the condition always true and printing "Access granted".

Q29. Answer B: All selected columns not aggregated must be included in the GROUP BY clause.

When using GROUP BY, every column in the SELECT list that is not aggregated must appear in the GROUP BY clause. In this query, product is missing from the GROUP BY clause, leading to an error.

Q30. Answer B: There is no error; the query is correct.

This query correctly computes the average salary with the subquery and filters employees with a salary greater than that average. There is no error with this query.

Q31. Answer B: Use a capture list such as [weak self] when the closure may keep the object alive longer than intended.

Closures can capture self strongly. In asynchronous or stored closures, using [weak self] can prevent unintended object retention and helps avoid retain-cycle patterns in larger designs.

Q32. Answer B: It produces a runtime error because nil cannot be a value in an NSDictionary.

In Objective-C, nil cannot be stored as a value in an NSDictionary. Attempting to do so will result in a runtime error.

Q33. Answer A: The range operator is incorrect; it should be 1..5 instead of 1...5.

In Kotlin, the range operator is .., not .... The correct loop should be:

```
for (i in 1..5) { println(i) }
```

Q34. Answer B: It may print "Not Equal" for larger integers because is checks identity, not equality.

The is operator checks for object identity, not equality. Although small integers may be interned in Python, using is for equality comparisons can produce unexpected results for larger numbers. The proper operator for equality is ==.

Q35. Answer A: The semicolon after the for loop causes the loop body to be empty; it prints "6 ".

The semicolon at the end of the for loop terminates the loop, so the block that follows executes only once after the loop, with $i having a value of 6. Thus, it prints "6 ".

Q36. Answer C: There is no error; the query is correct.

In this case, the query is standard and should work correctly as the subquery computes the average salary independently. Ambiguity is not an issue here because the subquery does not use a column from the outer query.

Q37. Answer B: The print statement is not indented correctly inside the function.

In Python, all code within a function must be indented consistently. The print statement should be indented inside the function. The correct version is:

```
def greet(name):
  print("Hello, " + name)
  return
```

Q38. Answer B: It crashes at runtime because a string cannot be force-cast to an Int.

The code force-casts a value of type Any (which is a String "123") to an Int, which is invalid and causes a runtime crash. The correct approach is to convert the string using Int("123").

Q39. Answer B: "Double: 5" because the float is promoted to double.

The literal 5.0f is a float. When passed to func, float is promoted to double (more precise conversion than to int). Thus, the func(double) overload is chosen, printing "Double: 5".

Q40. Answer C: Rewrite using a supported JOIN or derived-table form with clear aliases.

Some SQL dialects restrict certain self-referencing UPDATE subqueries. A clear JOIN or derived-table update with aliases is usually the safer and more portable correction.

Q41. Answer B: if (score >= 80) { cout << "Pass"; }

C++ conditions require parentheses around the condition and a valid output statement. Option B uses correct C++ syntax.

Q42. Answer B: The value is read-only after assignment.

In Kotlin, val declares a read-only value after initialization, while var declares a mutable variable.

Q43. Answer A: when green flag clicked

The "when green flag clicked" block is an event trigger that starts scripts in Scratch projects.

Q44. Answer A: Check whether the returned element is not null.

getElementById returns null if no element matches the id. A null check prevents runtime errors.

Q45. Answer A: SELECT customer_id, COUNT(*) FROM Orders GROUP BY customer_id;

COUNT(*) with GROUP BY customer_id calculates a separate order count for each customer.

Q46. Answer B: d1.keys() & d2.keys()

Dictionary key views support set-like operations. The & operator returns keys present in both dictionaries.

Q47. Answer A: It repeatedly processes input, updates the game state and renders frames.

A game loop keeps the game responsive by repeatedly handling input, updating objects and drawing the screen.

Q48. Answer A: AABB overlap checking

Axis-Aligned Bounding Box collision checks whether two rectangular bounds overlap and is fast for many 2D games.

Q49. Answer A: JSON

JSON is a lightweight data-interchange format widely used in web APIs and configuration files.

Q50. Answer B: A* Algorithm

A* combines the known cost so far with a heuristic estimate to guide the search efficiently toward the goal.