PRACTICE MCQ

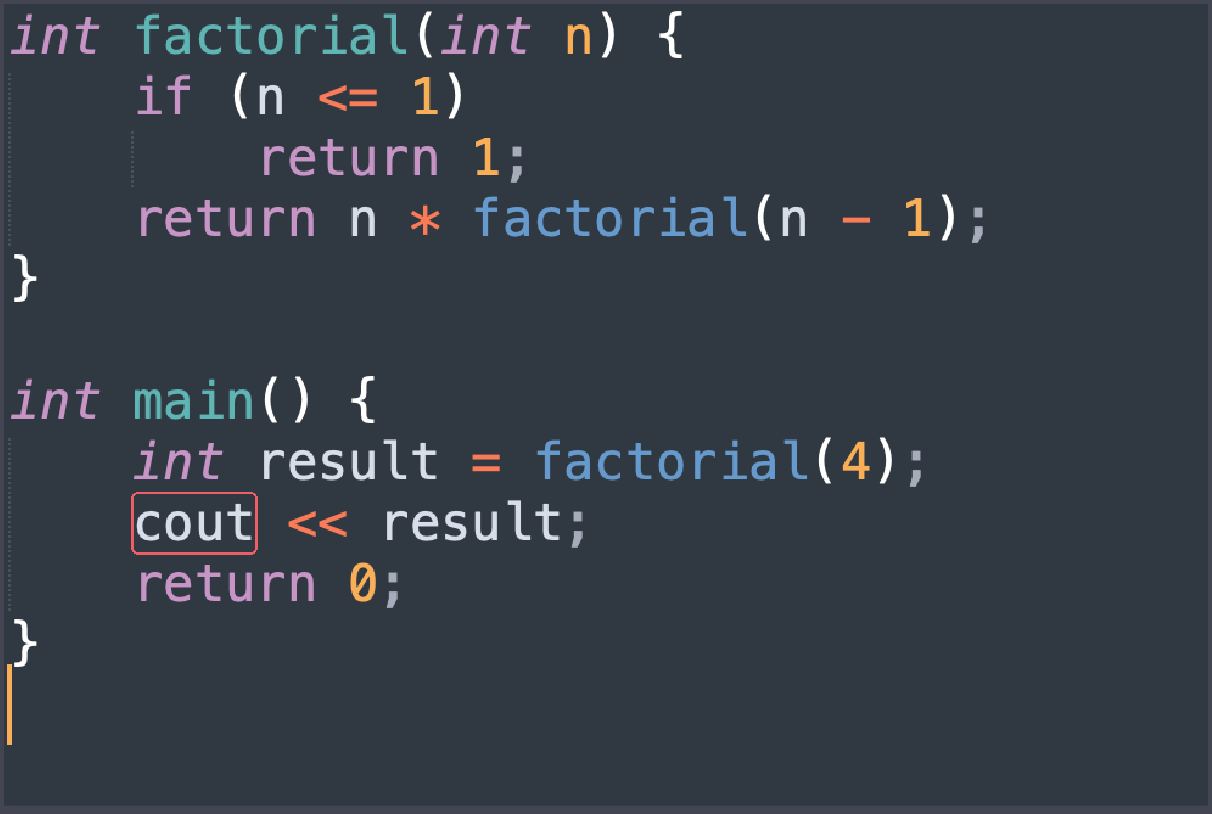
**Q1: What is the purpose of the new operator in C++?**

To create a new instance of a class.

To allocate memory for an object or an array of objects on the heap and returns the address of the memory created.

To initialise a variable with a new value. To allocate memory for a variable on the stack.

**Q2: Consider the following recursive C++ function:**



What will be the output of the main function?

4 8

16 24

**Q3: How is a 2D array stored in memory in C++?**

Elements are stored row-wise. Elements are stored column-wise.

Elements are stored randomly. It depends on the size of the array.

**Q4: How is memory allocated for a 2D array in C++ compared to a 1D array?**

1. 2D arrays are always allocated on the heap, while 1D arrays are allocated on the stack.
2. Both 1D and 2D arrays are always allocated on the stack.
3. 1D arrays are always allocated on the heap, while 2D arrays are allocated on the stack.
4. Both 1D and 2D arrays can be allocated on either the stack or the heap.

**Q5: What is the purpose of the delete operator in C++?**

To free up system resources. To delete a ﬁle from the ﬁle system.

To deallocate memory previously allocated using the new operator.

To delete a variable from the program.

**Q6: What is the difference between malloc and new in C++ for dynamic memory allocation?**

malloc is used for allocating memory of primitive data types, while new is used for objects.

malloc returns a pointer to uninitialized memory, while new returns a pointer to initialised memory.

malloc is used for static memory allocation, while new is used for dynamic memory allocation.

There is no difference; malloc and new can be used interchangeably.

**Q7: What is recursion in C++?**

A loop that repeats a block of code a speciﬁc number of times.

A function that calls itself directly or indirectly in order to solve a problem.

A feature that allows functions to return multiple values.

A type of variable that can hold multiple values.

**Q 8: What is the base case in a recursive function?**

The case where the function returns a value. The case where the function calls itself.

The case where the function stops calling itself and returns a result directly.

The case where the function contains a loop.

**Q9: What is the potential drawback of using recursion in C++ compared to iterative solutions?**

Recursion is always less eﬃcient in terms of time and space complexity.

Recursion can lead to stack overﬂow for large input sizes.

Recursion can only be applied to speciﬁc types of problems.

Recursion is not supported in C++.

**Q10: What is the term used to describe a situation where a function calls itself indirectly through a sequence of other functions?**

Circular calling Nested calling

Recursive calling Indirect recursion

**Q11:** **What is tail recursion in C++?**

A recursion where the base case is at the beginning of the function.

A recursion where the recursive call is the last operation in the function.

A recursion that involves calling multiple functions in a sequence.

A recursion that doesn't involve any base case.

**Q12: What is the term for a situation in which a function calls itself directly?**

Iteration Recursion

Looping Redundancy

**Q13: What is a class in C++?**

A data type that represents a single value. A collection of functions.

A user-deﬁned data type that contains data members and member functions.

A built-in data type.

**Q14: What is a potential beneﬁt of optimising tail-recursive functions in C++?**

Improved readability of the code.

Reduced risk of stack overﬂow for large input sizes.

Enhanced portability across different compilers.

Better handling of exceptions within the recursive function.

**Q15: What is the key difference between direct and indirect recursion in C++?**

Direct recursion involves a function calling itself, while indirect recursion involves multiple functions calling each other in a chain.

Direct recursion is not allowed in C++, while indirect recursion is a common practice.

Direct recursion always leads to a stack overﬂow, while indirect recursion does not.

Indirect recursion involves a function calling itself, while direct recursion involves multiple functions calling each other in a chain.

**Q16: What is the primary purpose of the call stack in C++?**

To store the program's source code. To manage dynamic memory allocation.

To keep track of function calls and local variables. To handle user input and output operations.

**Q17: What happens to a function's activation record on the call stack when the function completes its execution?**

It remains on the call stack indeﬁnitely. It is immediately removed from the call stack.

It becomes read-only. It is moved to the heap.

**Q18: What is a stack overﬂow in the context of C++ programming?**

A memory allocation error when using the new operator.

An exception thrown when an invalid type conversion occurs.

A runtime error that happens when the call stack exceeds its maximum size.

A compiler error due to incorrect syntax in the code.

**Q19: What is a common cause of a stack overﬂow in C++?**

Including too many header ﬁles.

Recursive function calls with insuﬃcient base cases.

Using the malloc function instead of new for dynamic memory allocation.

Declaring too many global variables.

**Q20: In a 32-bit system, what is the size of a pointer variable in C++ on a 32-bit system?**

2bytes 4 bytes

8 bytes It varies depending on the compiler.