PRACTICE MCQ:

Question 1: What is the purpose of the `new` operator in C++?

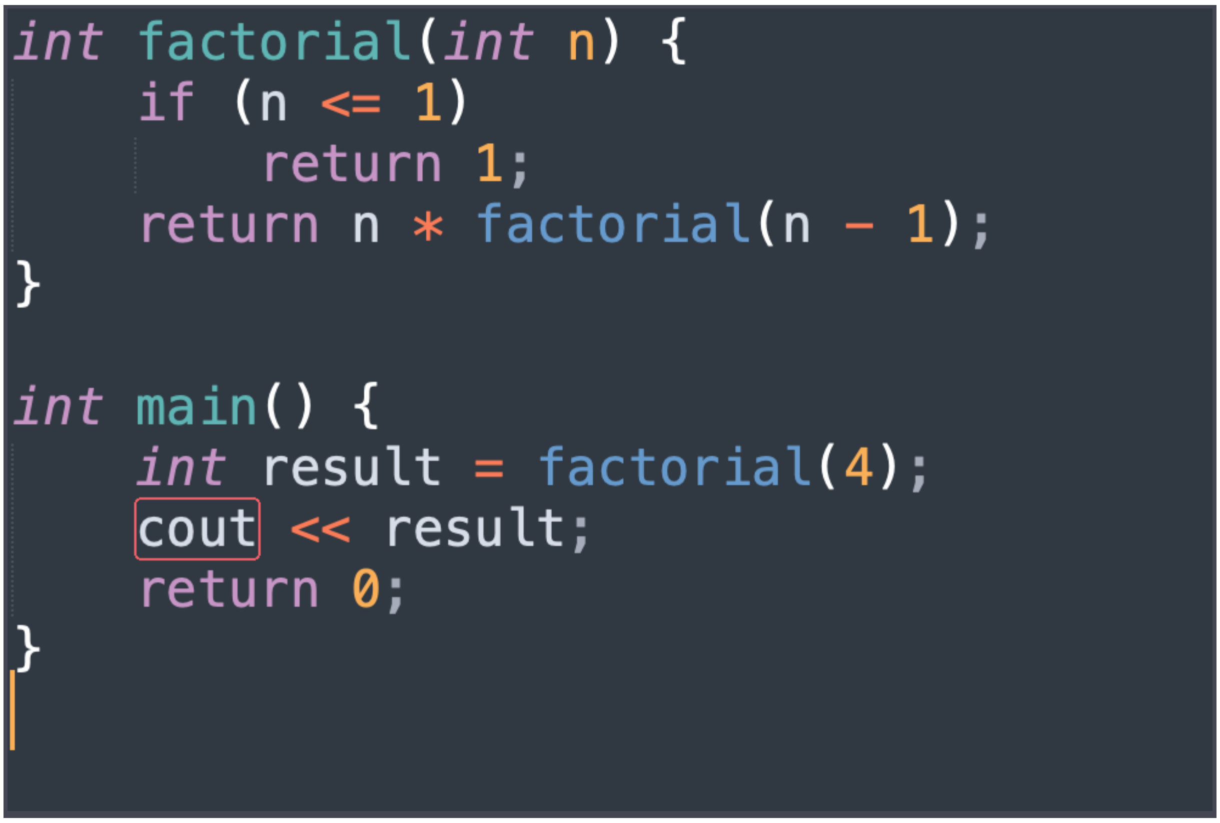
A. To create a new instance of a class.

B. To allocate memory for an object or an array of objects on the heap.

C. To initialise a variable with a new value.

D. To allocate memory for a variable on the stack.

Question 2: Consider the following recursive C++ function:



```
int factorial(int n) {
    if (n <= 1)
        return 1;
    return n * factorial(n - 1);
}

int main() {
    int result = factorial(4);
    cout << result;
    return 0;
}
```

What will be the output of the `main` function?

A. 4

B. 8

C. 16

D. 24

E. 120

Question 3: How is a 2D array stored in memory in C++?

A. Elements are stored row-wise.

B. Elements are stored column-wise.

C. Elements are stored randomly.

D. It depends on the size of the array.

Question 4: How is memory allocated for a 2D array in C++ compared to a 1D array?

A. 2D arrays are always allocated on the heap, while 1D arrays are allocated on the stack.

B. Both 1D and 2D arrays are always allocated on the stack.

C. 1D arrays are always allocated on the heap, while 2D arrays are allocated on the stack.

D. Both 1D and 2D arrays can be allocated on either the stack or the heap.

Question 5: What is the purpose of the `delete` operator in C++?

A. To free up system resources.

B. To delete a file from the file system.

C. To deallocate memory previously allocated using the `new` operator.

D. To delete a variable from the program.

Question 6: What is the difference between `malloc` and `new` in C++ for dynamic memory allocation?

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

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

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

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

Question 7: What is recursion in C++?

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

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

C. A feature that allows functions to return multiple values.

D. A type of variable that can hold multiple values.

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

A. The case where the function returns a value.

B. The case where the function calls itself.

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

D. The case where the function contains a loop.

Question 9: What is the potential drawback of using recursion in C++ compared to iterative solutions?

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

B. Recursion can lead to stack overflow for large input sizes.

C. Recursion can only be applied to specific types of problems.

D. Recursion is not supported in C++.

Question 10: What is the term used to describe a situation where a function calls itself indirectly through a sequence of other functions?

A. Circular calling

B. Nested calling

C. Recursive calling

D. Indirect recursion

Question 11: What is tail recursion in C++?

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

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

C. A recursion that involves calling multiple functions in a sequence.

D. A recursion that doesn't involve any base case.

Question 12: What is the term for a situation in which a function calls itself directly?

A. Iteration

B. Recursion

C. Looping

D. Redundancy

Question 13: What is a class in C++?

A. A data type that represents a single value.

B. A collection of functions.

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

D. A built-in data type.

Question 14: What is a potential benefit of optimising tail-recursive functions in C++?

A. Improved readability of the code.

B. Reduced risk of stack overflow for large input sizes.

C. Enhanced portability across different compilers.

D. Better handling of exceptions within the recursive function.

Question 15: What is the key difference between direct and indirect recursion in C++?

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

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

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

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

Question 16: What is the primary purpose of the call stack in C++?

A. To store the program's source code.

B. To manage dynamic memory allocation.

C. To keep track of function calls and local variables.

D. To handle user input and output operations.

Question 17: What happens to a function's activation record on the call stack when the function completes its execution?

A. It remains on the call stack indefinitely.

B. It is immediately removed from the call stack.

C. It becomes read-only.

D. It is moved to the heap.

Question 18: What is a stack overflow in the context of C++ programming?

A. A memory allocation error when using the `new` operator.

B. An exception thrown when an invalid type conversion occurs.

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

D. A compiler error due to incorrect syntax in the code.

Question 19: What is a common cause of a stack overflow in C++?

A. Including too many header files.

B. Recursive function calls with insufficient base cases.

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

D. Declaring too many global variables.

Question 20: In a 32-bit system, what is the size of a pointer variable in C++ on a 32-bit system?

A. 2 bytes

B. 4 bytes

C. 8 bytes

D. It varies depending on the compiler.