Day 100/180 Constructor and Destructor

1: What is shallow copy and Deep Copy

In C++, shallow copy duplicates the object's data, including pointers (shared memory), while deep copy creates a new object with new memory for each data member, ensuring independence. Shallow copies may lead to unintended side effects, while deep copies maintain object isolation.

For more:
https://www.geeksforgeeks.org/difference-between-shallow-and-deep-copy-of-a-class/

2: Why the Destructor of Objects is executed in reverse order.

In C++, the order in which destructors are executed is determined by the reverse order of object construction. This is known as the "stack discipline" or "last in, first out" (LIFO) order. The reason for this lies in how objects are managed in C++.

When objects are created, they are typically placed on the call stack. The last object to be constructed is the first one to be destructed because the call stack follows a last-in, first-out order. This behavior ensures that when a function or scope is exited, the objects created within that scope are properly cleaned up in the reverse order of their creation.

```cpp
#include <iostream>

class MyClass {
public:
    MyClass(const std::string& name) : name(name) {
        std::cout << "Constructing " << name << std::endl;
    }

    ~MyClass() {
        std::cout << "Destructing " << name << std::endl;
    }

private:
    std::string name;
};

int main() {
    MyClass obj1("obj1");
    MyClass obj2("obj2");

    return 0; // obj2 is destructed first, followed by obj1
}
```

In this example, when the main function is about to return, obj2 is destructed first, followed by obj1. This follows the LIFO order of the call stack.