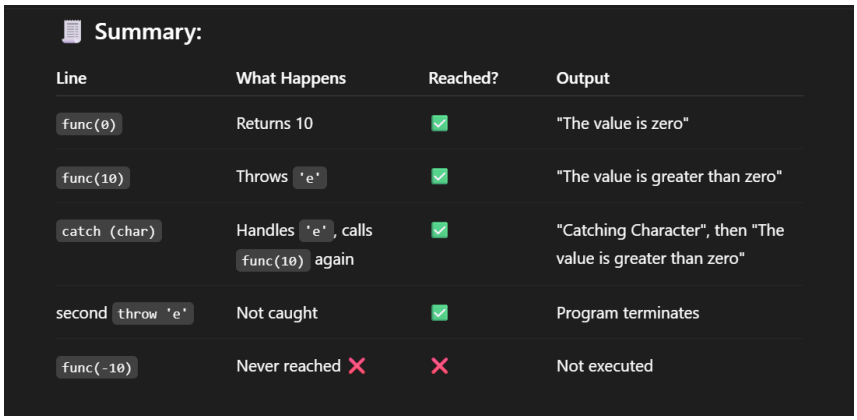
Question 1:

A- 10 15Normal Constructor called
15Copy Constructor called
15Copy Constructor called
1 1Normal Constructor called

B- The value is zero ← from func(0)
The value is greater than zero ← from func(10)
Catching Character ← char caught
The value is greater than zero ← func(10) again in catch block
terminate called after throwing an instance of 'char'
Aborted

Explanation:



📝 **Summary:**

Line	What Happens	Reached?	Output
`func(0)`	Returns 10	✅	"The value is zero"
`func(10)`	Throws `'e'`	✅	"The value is greater than zero"
`catch (char)`	Handles `'e'`, calls `func(10)` again	✅	"Catching Character", then "The value is greater than zero"
second `throw 'e'`	Not caught	✅	Program terminates
`func(-10)`	Never reached ❌	❌	Not executed

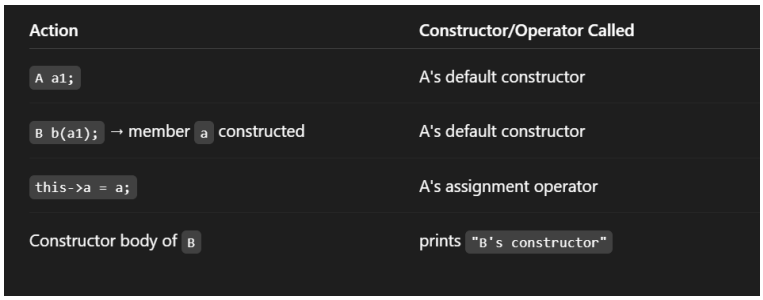
C- Count: 0
Count: 3
Count: 0

Explanation:

Step	Action	count
Start	`int Base::count = 0;`	0
`Base obj1`	`Base()`	1
`Derived obj2`	`Base()`, `Derived()`	3
Print	`Count: 3`	3
Destroy `obj2`	`~Derived()`, `~Base()`	1
Destroy `obj1`	`~Base()`	0
Final print	`Count: 0`	0

D- A's constructor
A's constructor
A's Assignment Operator
B's constructor

Explanation:



Action	Constructor/Operator Called
`A a1;`	A's default constructor
`B b(a1);` → member `a` constructed	A's default constructor
`this->a = a;`	A's assignment operator
Constructor body of `B`	prints `"B's constructor"`

Question 2:

A- Base operator+(const Base& other) {
return Base (data + other.data);
}

B- template <typename T>
T maximum (T a, T b) {
if (a > b) {
return a;
}
else {
return b;
};

C- Student(const std::string& name, int age, const std::string& university)
: Person(name, age), university(university) {}

```
void saveToFile(const std::string& filename) const override {
    std::ofstream file(filename);
    if (file.is_open()) {
        file << "Name: " << name << std::endl;
        file << "Age: " << age << std::endl;
        file << "University: " << university << std::endl;
        file.close();
        std::cout << "Data saved to file with details: "
                  << name << "\n" << age << "\n" << university << std::endl;
```

```
    } else {
        std::cerr << "Unable to open file: " << filename << std::endl;
    }
}
```

D-

```
    // Friend function declaration inside class
    template <typename U>
    friend U average(const Pair<U>& p);
};

// Friend function definition outside class
template <typename U>
U average(const Pair<U>& p) {
    return (p.first + p.second) / static_cast<U>(2);
}
```