Question 1:

A- Compile time error occurs when a code is not executed or compiled. This occurs due to problems in the body of the code. They are mainly syntax-related errors. Examples are: syntax errors, type errors, undefined variables or methods. Runtime errors are encountered during the execution of the code which result in invalid output or crashing during execution, mainly occurring when the program is running. Examples are: opening a file that doesn't exist would result in a file not found exception. Runtime errors can be dealt with using exception handling while compile time errors cannot.

B- 1) Encapsulation 2) Polymorphism

C- One advantage of using function overloading instead of generics (templates) in C++ is that overloading enhances code readability and is simpler to implement when handling a small number of specific data types or parameter variations. Function overloading allows multiple functions with the same name but different parameter lists, making the interface more intuitive without requiring template syntax or logic. It is particularly useful when each version of the function performs slightly different operations, or when generalization using templates is unnecessary.

D- No, a friend function of a base class cannot access the private or protected members of a derived class, unless it is explicitly declared as a friend in the derived class as well. This is because friendship is granted, not inherited. Being a friend of the base class only provides access to its private and protected members. It does not imply any special access to members of derived classes. Friendship gives access, not membership.

E- This is an example of multilevel inheritance.
Yes, an object of class C can be upcasted to an object of either class B or class A.
This is because class C inherits from class B, and class B inherits from class A — forming an inheritance chain. In C++, upcasting (converting from derived to base) is always allowed implicitly, as the derived class is a type of its base class.

Question 2:

A-
Output: Hello from Test() Main Started
Explanation: In C++, global objects are constructed before the main() function begins execution. Since a is a global object of class Test, its constructor is automatically invoked first, printing "Hello from Test()". After that, main() runs and prints "Main Started", resulting in the combined output.

B-
Output: Constructing an object of Test
Destructing an object of Test
Caught 10
Explanation: In C++, when an exception is thrown inside a try block, all local objects in that block are destroyed before transferring control to the catch block. So, the destructor of object t1 is called after the throw but before entering the catch.
This ensures proper cleanup and results in the constructor, destructor, and catch messages appearing in order.

C-
Output: x = 1 count = 0
x = 1 count = 1
x = 1.1 count = 0
Explanation: In C++, each instantiation of a function template has its own copy of static variables.
When fun<int>(1) is called twice, it shares the same static count, printing 0 and 1 respectively.
When fun<double>(1.1) is called, it uses a separate static count, so it again starts from 0.

D- **Output:** Throwing Block
Catching Default
Explanation: C++ matches exceptions to the first suitable catch block in order.
Since catch(...) appears before catch(char), it handles the thrown char, and the specific catch is skipped. As a result, "Catching Default" is printed instead of "Catching Char".

E- **Output:** c
c
t
o
Explanation: The loop calls foo(i), which returns 't' until i is divisible by 3, at which point it throws 'o'.
The last successfully returned value 't' is stored in c before the exception.
The catch block prints c and the caught exception, resulting in output: c c t o.

Question 3:

A- // Specialization of class template for <char, 10>

```
template <>
class QuestionTemplate<char, 10> {
    char arr1[10];
    char arr2[10];

public:
    QuestionTemplate() {
        for (int i = 0; i < 10; i++) {
            arr1[i] = 'a' + i;
            arr2[9 - i] = 'A' + i;
        }
    }

    string add() {
        string result;
        for (int i = 0; i < 10; i++) result += arr1[i];
        for (int i = 0; i < 10; i++) result += arr2[i];
        return result;
    }
};

int main() {
    QuestionTemplate<int, 10> qt;
    int* res = qt.add();
    for (int i = 0; i < 10; i++) {
        cout << res[i] << " ";
    }
    cout << endl;
    delete[] res;

    QuestionTemplate<char, 10> ct;
    cout << ct.add() << endl;

    return 0;
};
```

B-

```
class Test {
    int ID;
    string name;
    static int genID;

public:
    Test() {
        ID = ++genID;
        name = generateName(ID);
    }

    static string generateName(int id) {
        string idStr = to_string(id);
        while (idStr.length() < 4) {
            idStr = "0" + idStr;
        }
        return idStr;
    }

    void operator+(const string& filename) {
        ofstream file(filename, ios::app);  // Open in append mode
        if (file.is_open()) {
            file << "ID = " << ID << ", Name = " << name << endl;
            file.close();
        } else {
            cerr << "Error opening file: " << filename << endl;
        }
    }
};

int Test::genID = 0;

int main() {
    Test t1, t2;
    t1 + "outfile.txt";
    t2 + "outfile.txt";
    return 0;
}
```

Questions: 4, 5, & 6 uploaded on GITHUB