🧠 1. Describe the characteristics of ANSI C++, C++ tokens, and step through the structure of a C++ program

Problem Statement:
List and explain 5 key characteristics of ANSI C++. Then, identify all tokens in a given C++ program and describe the structure of the program.

Solution:

Characteristics of ANSI C++:

1. **Object-Oriented:** Supports encapsulation, inheritance, and polymorphism.
2. **Strongly Typed:** Variables must be declared with a type.
3. **Rich Library Support:** Includes STL and standard libraries.
4. **Backward Compatibility:** Compatible with most C programs.
5. **Platform Independent (Source Code):** Can compile on various systems with the appropriate compiler.

C++ Tokens in this Code:

```
#include<iostream>
using namespace std;

int main() {
   int a = 10, b = 20;
   int sum = a + b;
   cout << "Sum = " << sum;
   return 0;
}
```

Tokens: `#include`, `<iostream>`, `using`, `namespace`, `std`, `int`, `main`, `a`, `=`, `10`, `b`, `20`, `sum`, `+`, `cout`, `<<`, `"Sum = "`, `return`, `0`, `;`, `{`, `}`

Structure:

- Preprocessor Directive
- Namespace Declaration

- `main()` function
- Variable Declarations and Initialization
- Output Statement
- Return Statement

🧠 2. Write a simple C++ program

Problem Statement:
Write a C++ program that takes two integers as input and prints their sum.

Solution:

```
#include<iostream>
using namespace std;

int main() {
    int num1, num2, sum;
    cout << "Enter two numbers: ";
    cin >> num1 >> num2;
    sum = num1 + num2;
    cout << "Sum = " << sum;
    return 0;
}
```

🧠 3. Compile and execute a C++ program and describe associated files

Problem Statement:
Explain the steps to compile and execute a C++ program. List the intermediate files generated during compilation (in GCC/Visual Studio).

Solution:

Steps:

- Write code in a `.cpp` file.
- Compile using `g++ filename.cpp -o outputname`
- Run using `./outputname`

Associated Files:

- `.cpp` – Source file
- `.o` or `.obj` – Object file (intermediate machine code)
- Executable (`.exe` or no extension in Linux)

🧠 4. Use fundamental data types and qualifiers

Problem Statement:
Declare and print a `short`, `int`, `long`, and `float` using appropriate qualifiers (`signed`, `unsigned`, `const`).

Solution:

```
#include<iostream>
using namespace std;

int main() {
    unsigned short int age = 25;
    const float pi = 3.1415;
    signed long salary = 150000;

    cout << "Age: " << age << "\nPI: " << pi << "\nSalary: " << salary << endl;
    return 0;
}
```

🧠 5. Use bool data type

Problem Statement:
Declare a boolean variable that determines if a number is even, and print true or false.

Solution:

```
#include<iostream>
using namespace std;

int main() {
    int num = 10;
    bool isEven = (num % 2 == 0);
    cout << boolalpha << "Is number even? " << isEven << endl;
    return 0;
}
```

🧠 6. Implicit and explicit type conversion

Problem Statement:
Demonstrate both implicit and explicit conversion from `float` to `int`.

Solution:

```
#include<iostream>
using namespace std;

int main() {
    float pi = 3.14;
    int x = pi;  // Implicit
    int y = (int)pi;  // Explicit

    cout << "Implicit: " << x << ", Explicit: " << y << endl;
    return 0;
}
```

🧠 7. Constants and numeric constants

Problem Statement:
Define constants using `#define` and `const`, and use them in arithmetic operations.

Solution:

```
#include<iostream>
#define TAX 0.05
using namespace std;

int main() {
    const int price = 1000;
    float total = price + price * TAX;
    cout << "Total price: " << total << endl;
    return 0;
}
```

🧠 8. Character and string constants

Problem Statement:
 Store character and string literals in variables and print their ASCII values.

Solution:

```
#include<iostream>
using namespace std;

int main() {
    char letter = 'A';
    string word = "Hello";

    cout << "Letter: " << letter << ", ASCII: " << (int)letter << endl;
    cout << "Word: " << word << endl;
    return 0;
}
```

🧠 9. Escape characters and symbolic constants

Problem Statement:
 Use escape sequences like `\n`, `\t`, and create symbolic constants using `const`.

Solution:

```
#include<iostream>
using namespace std;

int main() {
    const char TAB = '\t';
    const char NEWLINE = '\n';

    cout << "Item" << TAB << "Price" << NEWLINE;
    cout << "Book" << TAB << "$10" << NEWLINE;
    return 0;
}
```

🧠 10. Enumeration constants

Problem Statement:
 Use `enum` to define days of the week and print the numerical value of Wednesday.

Solution:

```
#include<iostream>
```

```
using namespace std;

enum Day { Sunday, Monday, Tuesday, Wednesday, Thursday, Friday, Saturday };

int main() {
    Day today = Wednesday;
    cout << "Numeric value of Wednesday: " << today << endl;
    return 0;
}
```

🧠 11. Using variables in C++

Problem Statement:
Declare multiple variables of different types, assign values, and print them.

Solution:

```
#include<iostream>
using namespace std;

int main() {
    int id = 101;
    float salary = 55000.5;
    char grade = 'A';

    cout << "ID: " << id << "\nSalary: " << salary << "\nGrade: " << grade << endl;
    return 0;
}
```

🧠 12. Variable scope

Problem Statement:
Demonstrate variable scope in nested blocks and functions.

Solution:

```
#include<iostream>
using namespace std;

int x = 5; // Global

void display() {
    int x = 10; // Local to function
    cout << "Inside display(): x = " << x << endl;
}

int main() {
    int x = 20; // Local to main
    {
        int x = 30; // Block scope
        cout << "Inside nested block: x = " << x << endl;
    }
    cout << "Inside main(): x = " << x << endl;
    display();
    cout << "Global x = " << ::x << endl;
    return 0;
}
```
