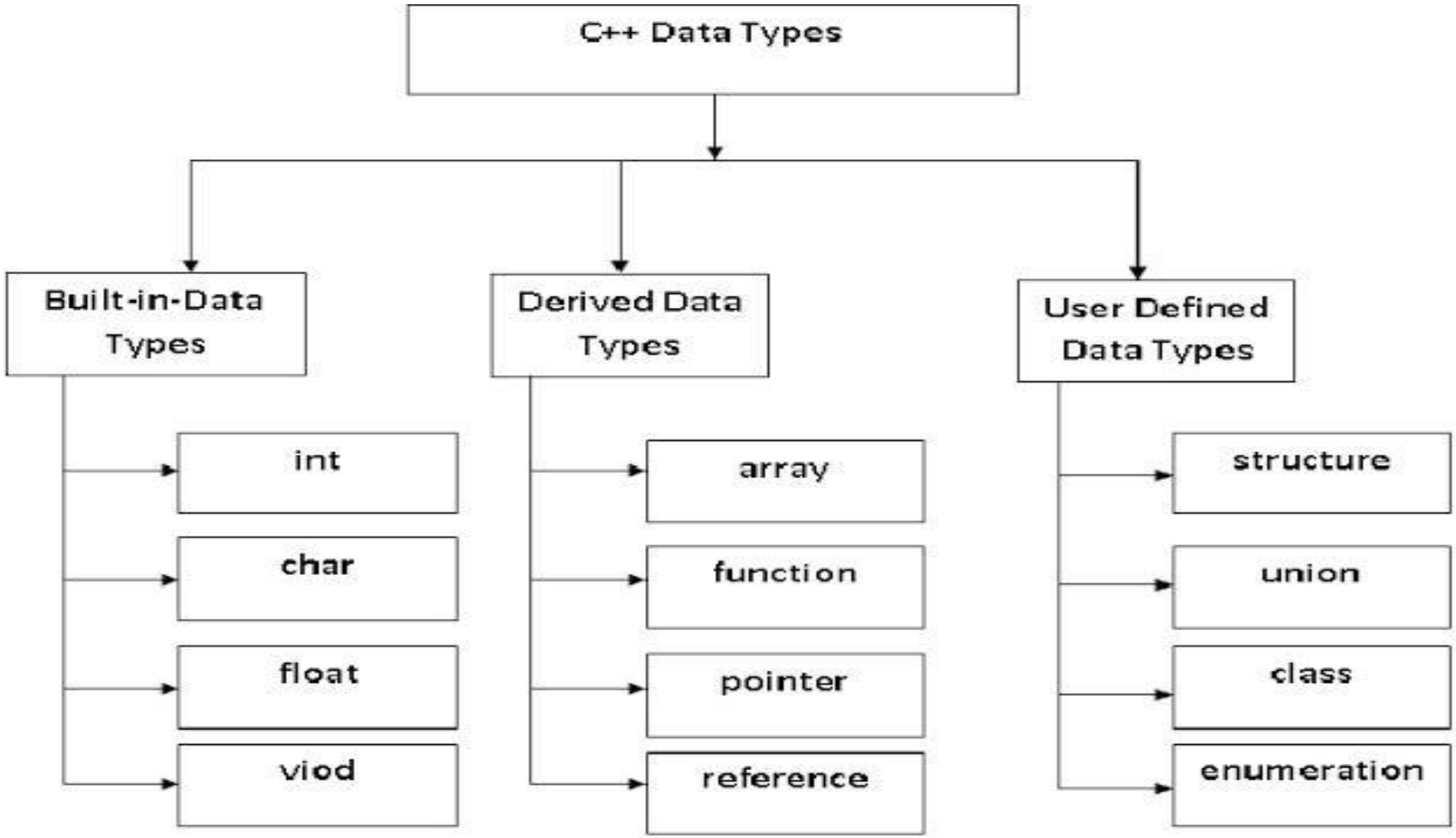
Object oriented programming with C++

C++ Datatypes & Operators


C++ Data Types
Built-in-Data Types
Derived Data Types
User Defined Data Types
int
char
float
viod
array
function
pointer
reference
structure
union
class
enumeration

C++

- char:: 1 byte, Stores a single character/letter/number, or ASCII values
- int:: 2 or 4 bytes , Stores whole numbers, without decimals
- float:: 4 bytes, Stores fractional numbers, containing one or more decimals. Sufficient for storing 6-7 decimal digits
- double:: 8 bytes, Stores fractional numbers, containing one or more decimals. Sufficient for storing 15 decimal digits
- boolean:: 1 byte, Stores true or false values

- The precision of a floating point value indicates how many digits the value can have after the decimal point. The precision of float is only six or seven decimal digits, while double variables have a precision of about 15 digits.
- A boolean data type is declared with the bool keyword and can only take the values true or false.
- When the value is returned, true = 1 and false = 0.

char

- The char data type is used to store a single character. The character must be surrounded by single quotes, like ‘A’ or ‘c’.

```
char a = 65, b = 66, c = 67;
cout << a;
cout << b;
cout << c;
```

Strings

- The string type is used to store a sequence of characters (text). This is not a built-in type, but it behaves like one in its most basic usage. String values must be surrounded by double quotes.

```
// Include the string library
#include <string>
// Create a string variable
string greeting = “Hello”;
// Output string value
cout << greeting;
```

signed and unsigned

These are used only with integer types (char, int, short, long, long long) to define whether they can hold negative values.

Example:

signed int x = -10; // Can hold negative values

unsigned int y = 10; // Cannot hold negative values

Modifier	Meaning	Range (Typical 4-byte int)
signed	Can store **negative & positive** values	-2,147,483,648 to 2,147,483,647
unsigned	Can store **only positive** values	0 to 4,294,967,295

short and long

Used to change the storage size of integer types.

Example:

short a = 32767;

long b = 100000L;

long long c = 900000000000LL;

Modifier	Typical Size	Range (Depends on System)
`short`	2 bytes	-32,768 to 32,767
`long`	4 or 8 bytes	Larger range than `int`
`long long`	8 bytes	Much larger range

const

Makes a variable read-only (cannot be changed after initialization).

Used to protect data from modification.

Example:

```
const int pi = 3.14;
pi = 3.15;  // Error: assignment of read-only variable
```

volatile

Tells the compiler not to optimize the variable because it may change unexpectedly (e.g., hardware registers, multithreading).

Example:

```
volatile int sensorValue;  // Always read fresh value


Short a;        // Same as short int a;
long b;         // Same as long int b;
long long c;    // Same as long long int c;
unsigned d;     // Same as unsigned int d;
```

Category	Operator	Associativity
Postfix	() [] -> . ++ - -	Left to right
Unary	+ - ! ~ ++ - - (type) * & sizeof	Right to left
Multiplicative	* / %	Left to right
Additive	+ -	Left to right
Shift	<< >>	Left to right
Relational	< <= > >=	Left to right
Equality	== !=	Left to right
Bitwise AND	&	Left to right
Bitwise XOR	^	Left to right
Bitwise OR	|	Left to right
Logical AND	&&	Left to right
Logical OR	||	Left to right
Conditional	?:	Right to left
Assignment	= += -= *= /= %= >>= <<= &= ^= |=	Right to left
Comma	,	Left to right

Conditional operator

- Variable = (condition) ? expressionTrue : expressionFalse;
- Int time = 20;
- string result = (time < 18) ? “Good day.” : “Good evening.”;

cout << result;

- int absValue = (num < 0) ? –num : num;

```
#include <iostream>
int main() {
    int a = 3, b = 5, c = 7;
    a = b == c ? b++ : c--;
    std::cout << “a = “ << a << “, b = “ << b << “, c = “ << c << std::endl;
    return 0;
}
```

```
#include <iostream>
int main() {
    unsigned int x = 0;
    x = -1;
    std::cout << “x = “ << x << std::endl;
    return 0;
}
```

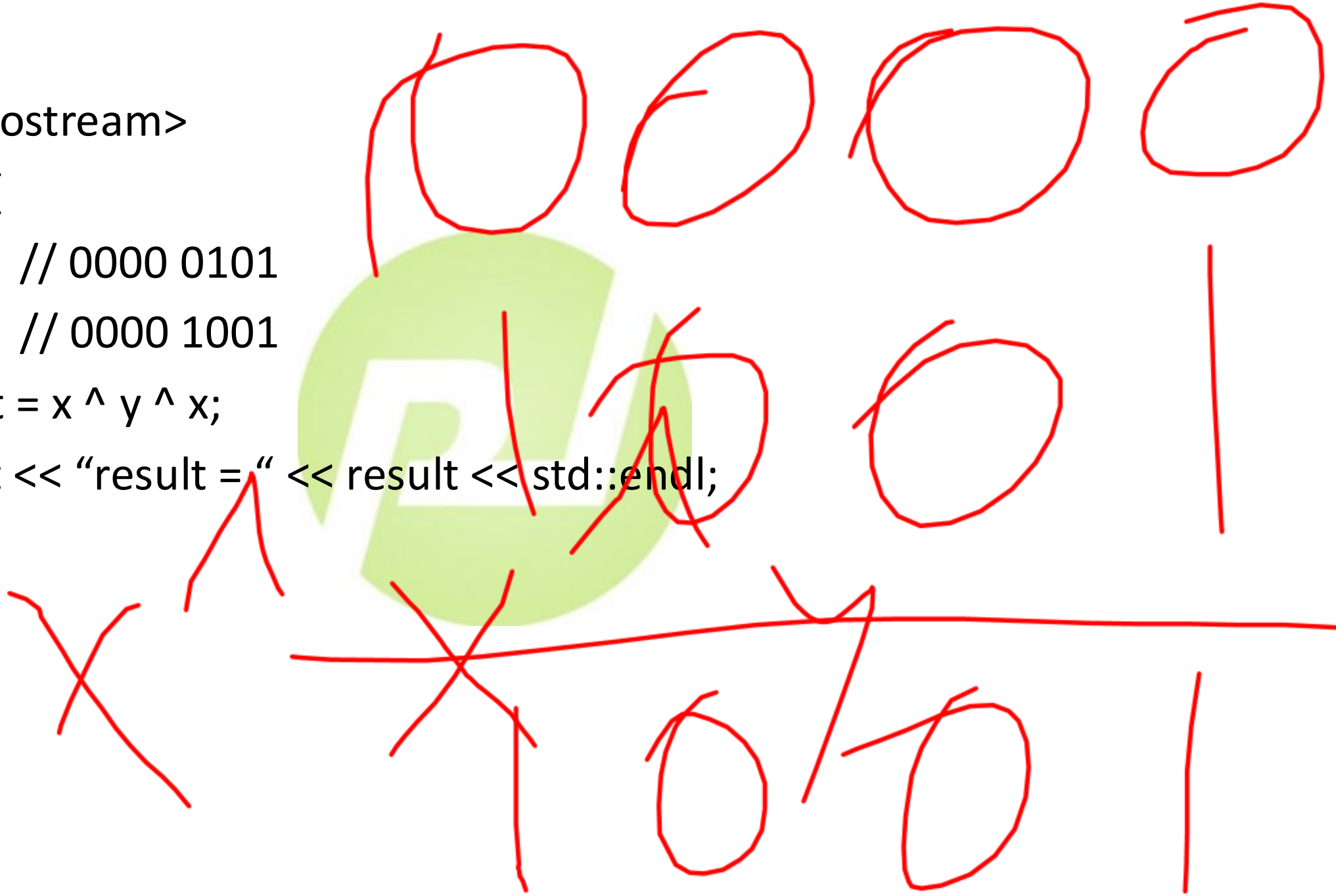
A) X = -1

B) X = 0

C) X = 4294967295 (assuming 32-bit unsigned int)

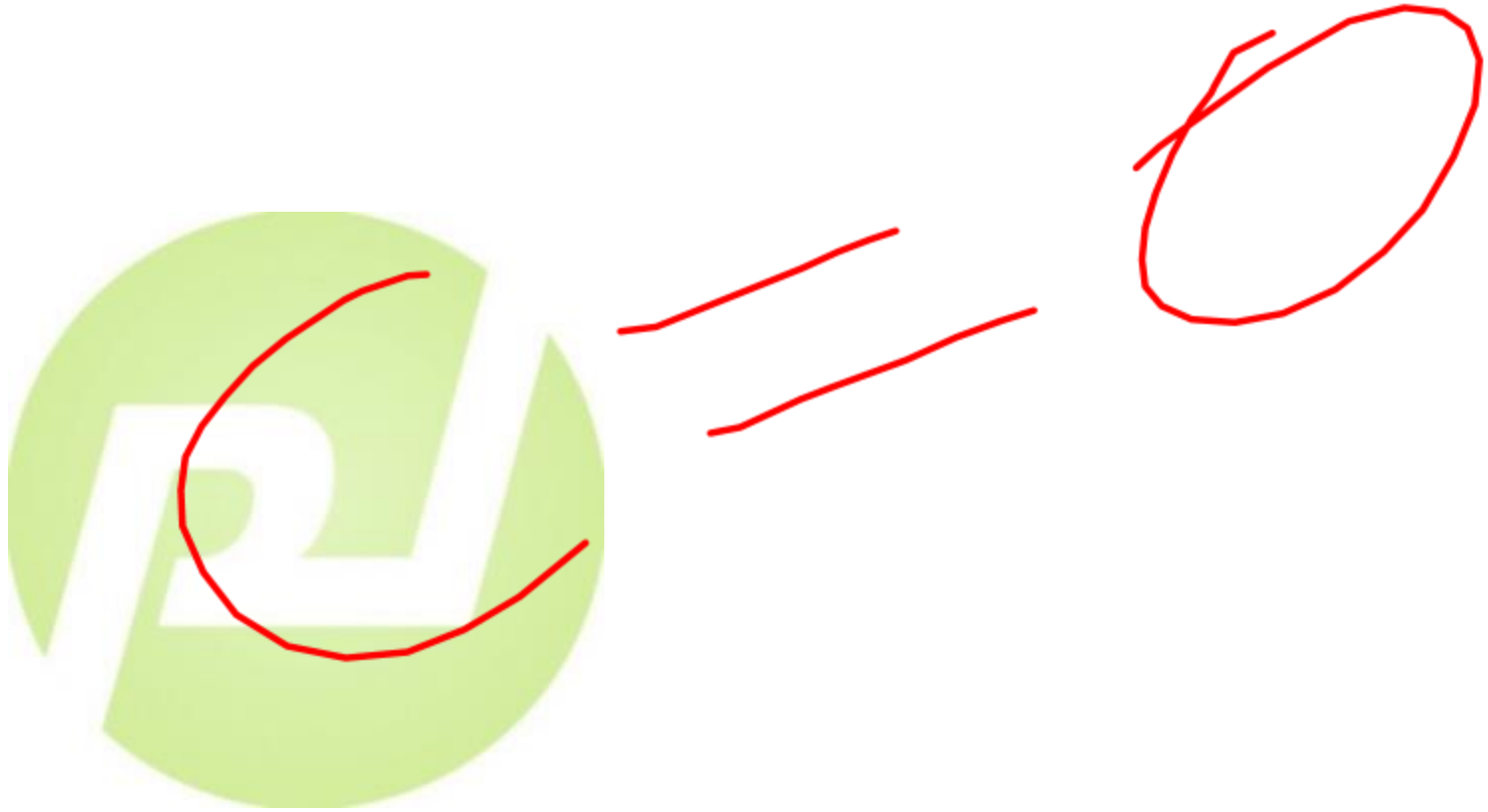
D) Compilation Error

```
#include <iostream>
int main() {
    int x = 5;  // 0000 0101
    int y = 9;  // 0000 1001
    int result = x ^ y ^ x;
    std::cout << “result = “ << result << std::endl;
    return 0;
}
```



```
#include <iostream>
int main() {
    int a = 3;
    int b = 10;
    int c = a & b == 2;

    std::cout << “c = “ << c << std::endl;
    return 0;
}
```



```
#include <iostream>
int main() {
    int x = 5, y = 0, z = 3;

    if (x & y && z)
        std::cout << “Condition is true\n”;
    else
        std::cout << “Condition is false\n”;

    return 0;
}
```

Thank you