Week 2

CS 150 – C++ Programming I
In-Person Lecture 2

A Little Review

- We'll start with a little review of weeks 1
- To review the material, I'm going to ask you questions
  - You are going to confer with your "group" (those in your row)
  - You'll answer using a "clicker" program
- Log into the desktop computers
- Double-click the file in Q:\faculty5\sgilbert\cs150\TTAFT
- Log in with your Canvas ID (eg. sgilbert) and your student ID (eg. C01234567), just like the Homework Console

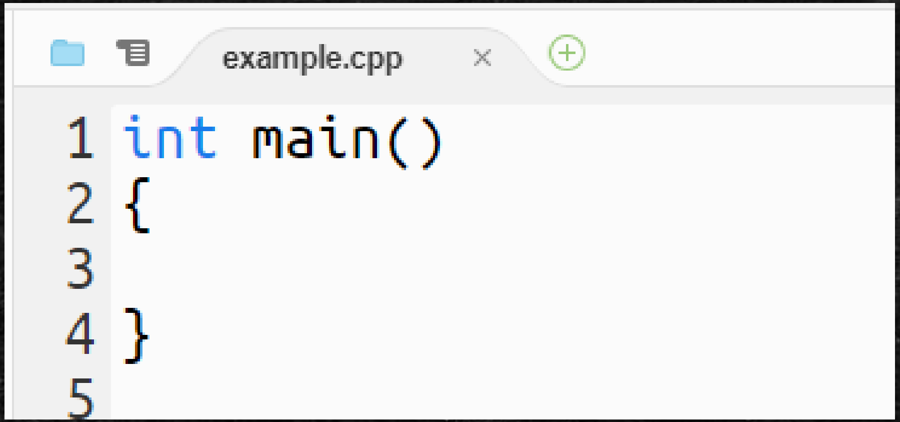
Review: Hello World

```
// Prints hello world on the screen
#include <iostream>
using namespace std;

int main()
{
    cout << "hello world" << endl;
    return 0;  // optional in main
}
```

Question

- What happens if this program is compiled & run?
  - A. Compiles, runs and returns `0` to the OS
  - B. Compiles but does not run
  - C. Does not compile: missing `return` statement
  - D. Does not compile: missing `#include`



```cpp
int main()
{

}

```

Question

- This code
  - A. Compiles, runs, and prints "Hello CS 150" on a line by itself
  - B. Compiles, runs, prints "Hello CS 150" w/o newline
  - C. Compiles, but crashes when it is run
  - D. Does not compile

```
#include <iostream>
int main() {
    cout << "Hello CS 150";
}
```

Question

- This code:
  - A. Compiles, runs, and prints "Hello CS 150" on a line by itself
  - B. Compiles, runs, prints "Hello CS 150" w/o newline
  - C. Compiles, but crashes when it is run
  - D. Does not compile

```
#include <iostream>
int main() {
   std::cout << "Hello CS 150";
}
```

Question

This Code

A. Compiles, runs, and prints "Hello CS 150" on a line by itself
B. Compiles, runs, prints "Hello CS 150" w/o newline
C. Compiles, but crashes when it is run
D. Does not compile

```
#include <iostream>
using std::cout;
int main() {
   cout << "Hello CS 150\n";
}
```

Question

This code:

A. Compiles, runs, and prints "Hello CS 150" on a line by itself
B. Compiles, runs, prints "Hello CS 150" w/o newline
C. Compiles, but crashes when it is run
D. Does not compile

```
#include <iostream>
using std::cout;
int main() {
   cout << "Hello CS 150" << endl;
}
```

Question

This code:

A. OK; Compiles, runs and prints 8 (all C++ versions)
B. Type error; must call `sqrt(64.0)`
C. May compile, but is actually undefined behavior
D. Syntax error; should use `math.sqrt(64)`
E. Declaration error for `endl`; Should be `std::endl`

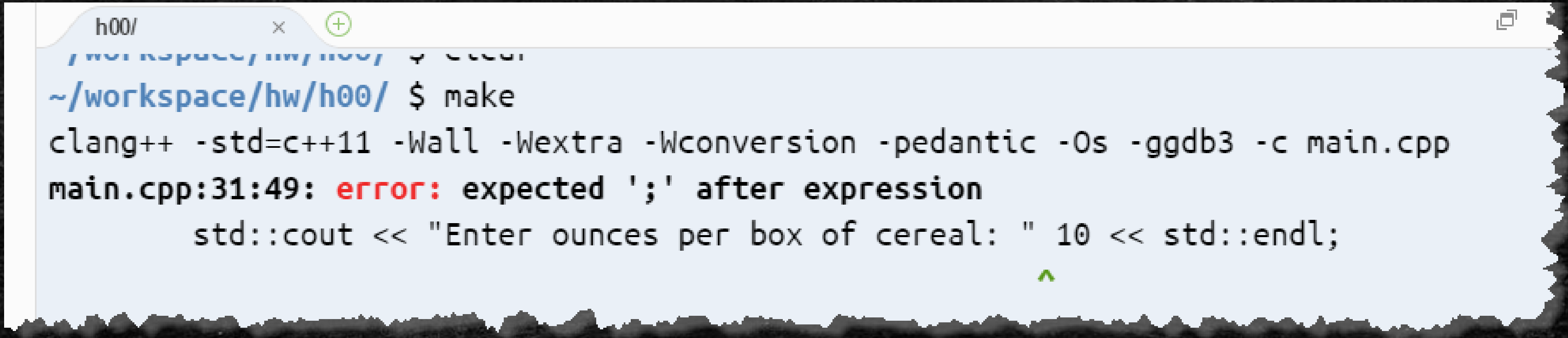
```
#include <iostream>
using namespace std;
int main() {
   cout << sqrt(64) << endl;
}
```

Review: C++ Mechanics

- Physical steps to build a C++ program
  1. Create the source code in your editor
  2. Use the compiler toolchain (gcc, clang, cl, etc.) to:
     A. Pre-process the source code (`#include`, etc)
     B. Parse and generate intermediate code from the translation unit
     C. Convert intermediate code to native object code
     D. Link object code with start-up and library code
  3. Run or debug the program
- Types of errors: compile-time, link-time, run-time
  – Syntax, type, declaration, logic, exception

Question

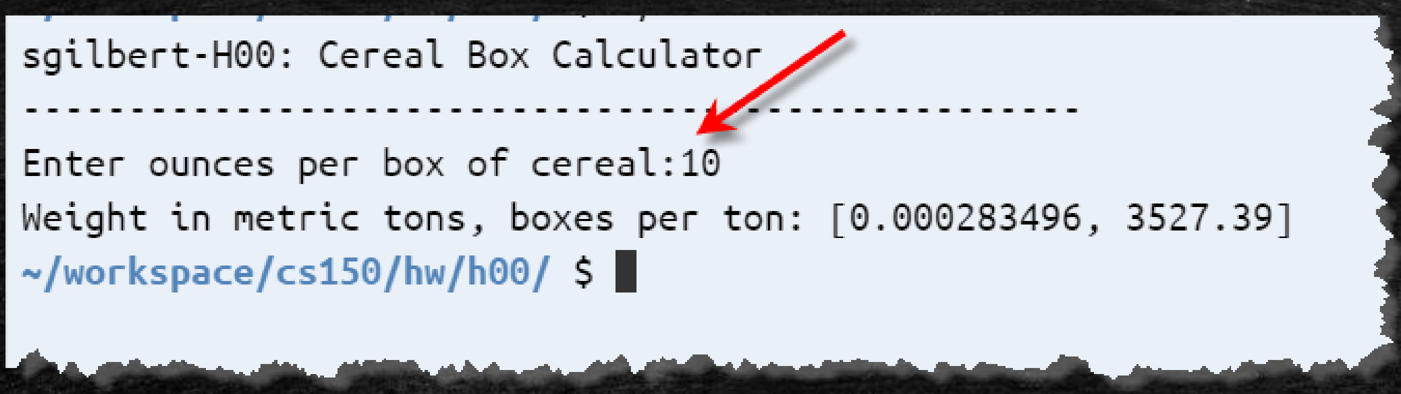
- What kind of error is this?
  - A. A logic error
  - B. A syntax error
  - C. A runtime error
  - D. A linker error



```
~/workspace/hw/h00/ $ make
clang++ -std=c++11 -Wall -Wextra -Wconversion -pedantic -Os -ggdb3 -c main.cpp
main.cpp:31:49: error: expected ';' after expression
        std::cout << "Enter ounces per box of cereal: " 10 << std::endl;
                                                        ^
```

Question

- Given the specifications for H00, what kind of error is this?
  - A. A logic error
  - B. A syntax error
  - C. A runtime error
  - D. This is not an error



```
sgilbert-H00: Cereal Box Calculator
---------------------------------------------------
Enter ounces per box of cereal:10
Weight in metric tons, boxes per ton: [0.000283496, 3527.39]
~/workspace/cs150/hw/h00/ $
```

Review: The CS 150 Toolchain

- In CS 150 we use the GCC toolchain to build the program
- Build instructions executed by the Unix program `make`
  - make – just builds the executable
    - Catch syntax, type and declaration errors at this point
  - make run – builds the executable and runs it interactively
  - make test – builds and runs instructor tests
  - make stest – builds and runs your student tests
  - make submit – builds, tests and submits homework
  - make lint – check for common programming errors

Review: Variables and Values

- Values are quantities of binary data: 0100-0001
  - May represent different types: 65, 'A', red
  - A set of bits interpreted according to its type
- Variable: a named storage location that holds a value
  - Declaration: associate name with a particular type
  - Definition: create an object in memory to store a value
  - Initialization: provide an initial value when defining
  - Assignment: place a new value in an existing variable
  - Input: read input from the user and store in a variable

Question

What prints out here (assuming all includes, etc.)

A. size=42, cost=9.99
B. Logic error; uninitialized variable
C. Type error; assign double value to int variable
D. Declaration error: undeclared variable

```
int size = 42;
cost = 9.99;
cout << "size=" << size
     << ", cost=" << cost << endl;
```

Question

What happens (assuming in main, all includes, 1 file)

A. size=42, cost=9
B. Type Error: assign double value to int variable
C. Declaration Error: undeclared variable
D. Linker error: object not found (undefined)

```
extern int cost;
int size = 42;
cost = 9.99;
cout << "size=" << size
     << ", cost=" << cost << endl;
```

Question

What prints out here (assuming all includes, etc.)

A. 0
B. 1
C. 2
D. Unpredictable result; logic error
E. Does not compile; syntax error

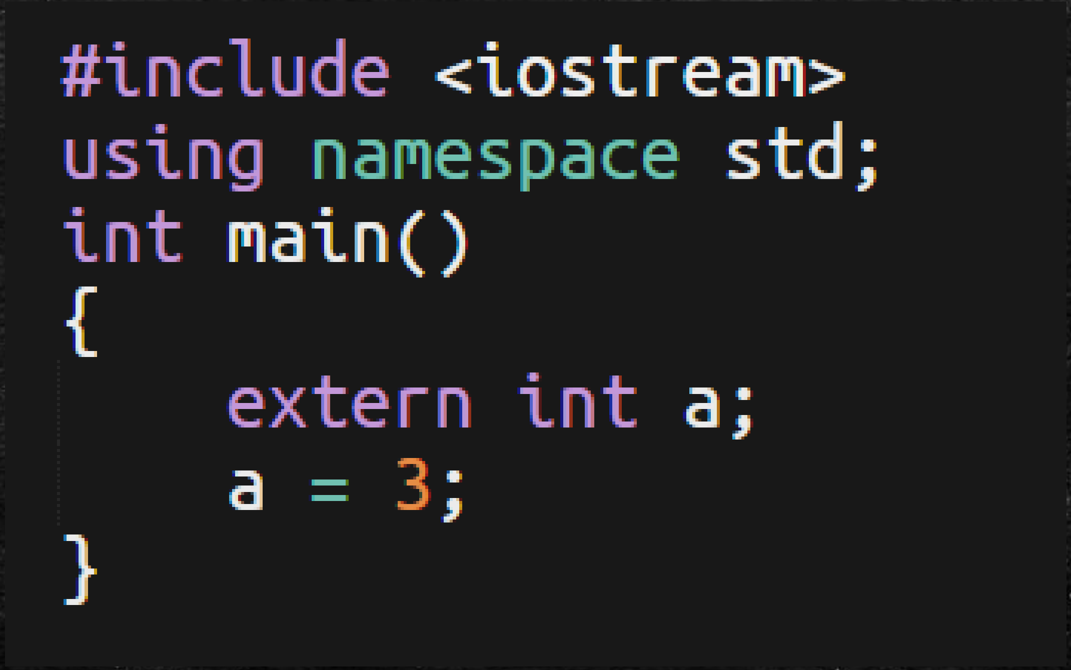
```
double bottles;
double bottleVolume = bottles * 2;
cout << bottleVolume << endl;
```

Question

- Which of these five concepts is illustrated here.

  I. Declaration , II. Definition, III. Initialization,
  IV. Assignment, V. input

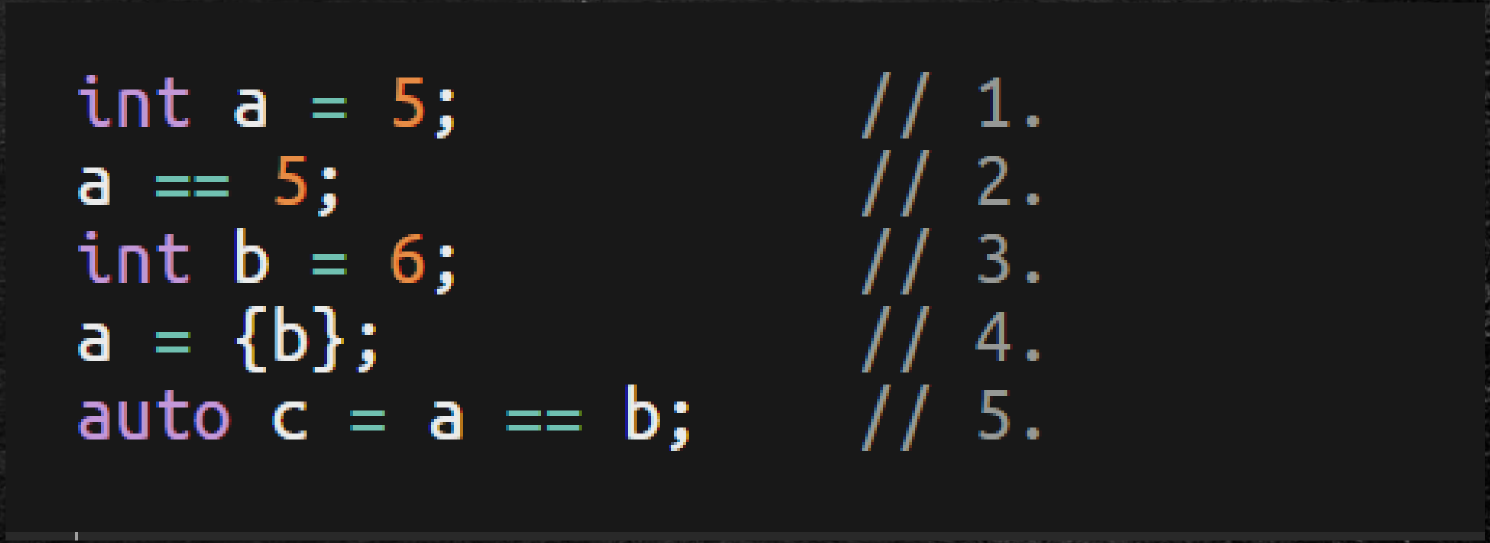
  - A. All of them
  - B. I only
  - C. I and II
  - D. I and III
  - E. I and IV



```
#include <iostream>
using namespace std;
int main()
{
    extern int a;
    a = 3;
}
```

Question

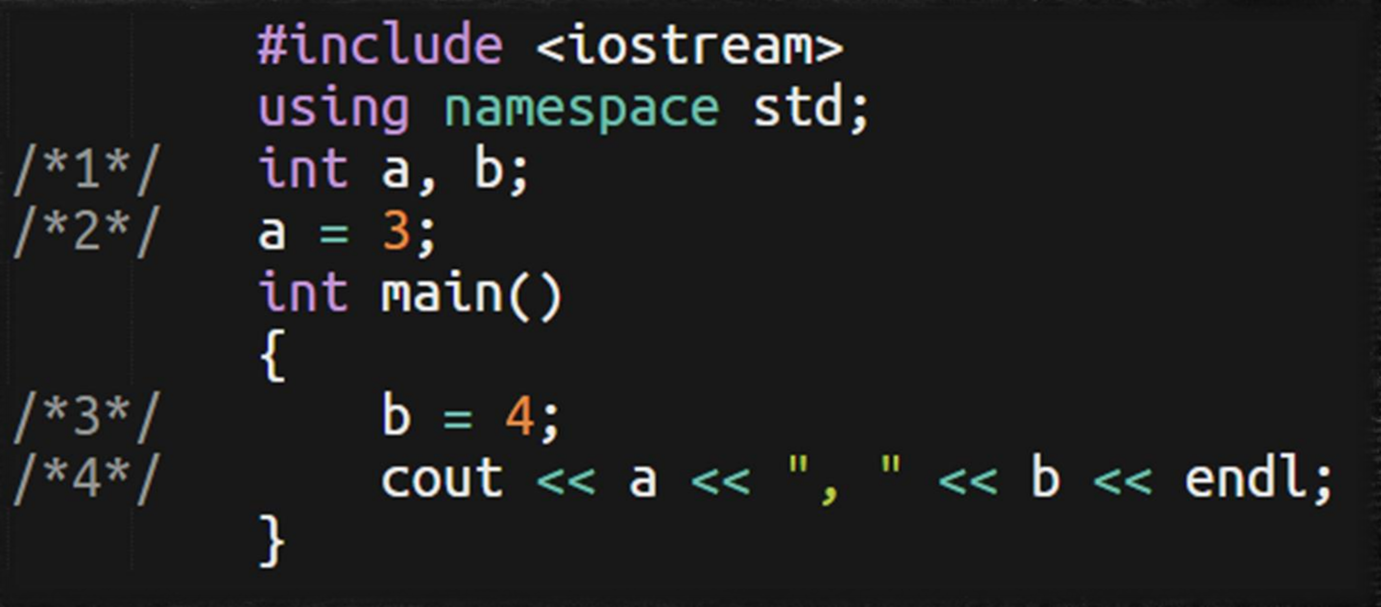
- This code is legal, compiles and is well defined. Which line or lines contains an assignment?
  - A. 1, 3, 5
  - B. 2
  - C. 4
  - D. All of these
  - E. All but line 2



```
int a = 5;              // 1.
a == 5;                 // 2.
int b = 6;              // 3.
a = {b};                // 4.
auto c = a == b;        // 5.
```

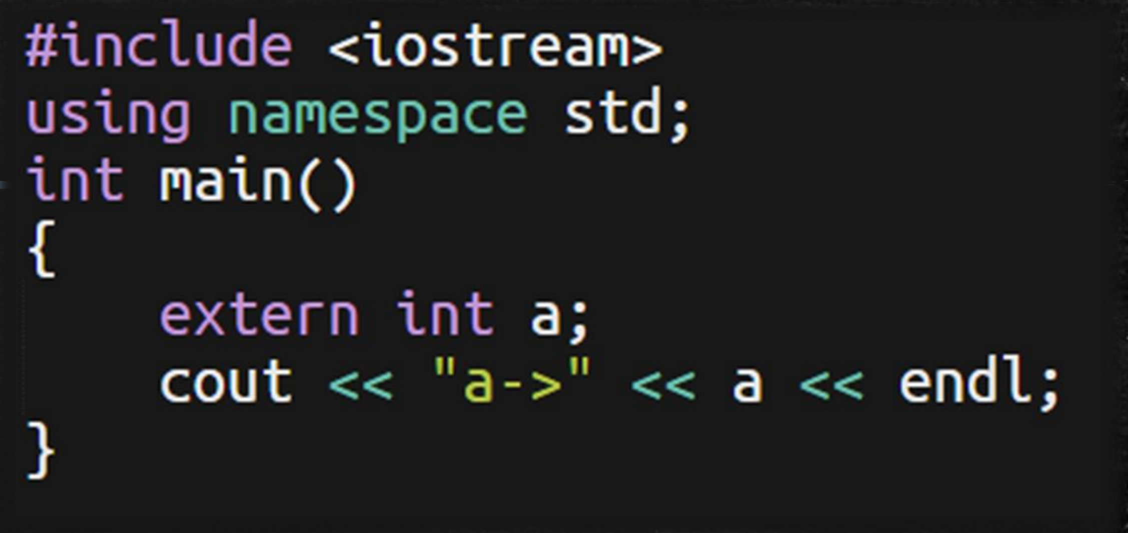
Question

- Which line is illegal?
  - A. 1
  - B. 2
  - C. 3
  - D. Several lines
  - E. None of these lines are illegal



```
       #include <iostream>
       using namespace std;
/*1*/  int a, b;
/*2*/  a = 3;
       int main()
       {
/*3*/      b = 4;
/*4*/      cout << a << ", " << b << endl;
       }
```

Question

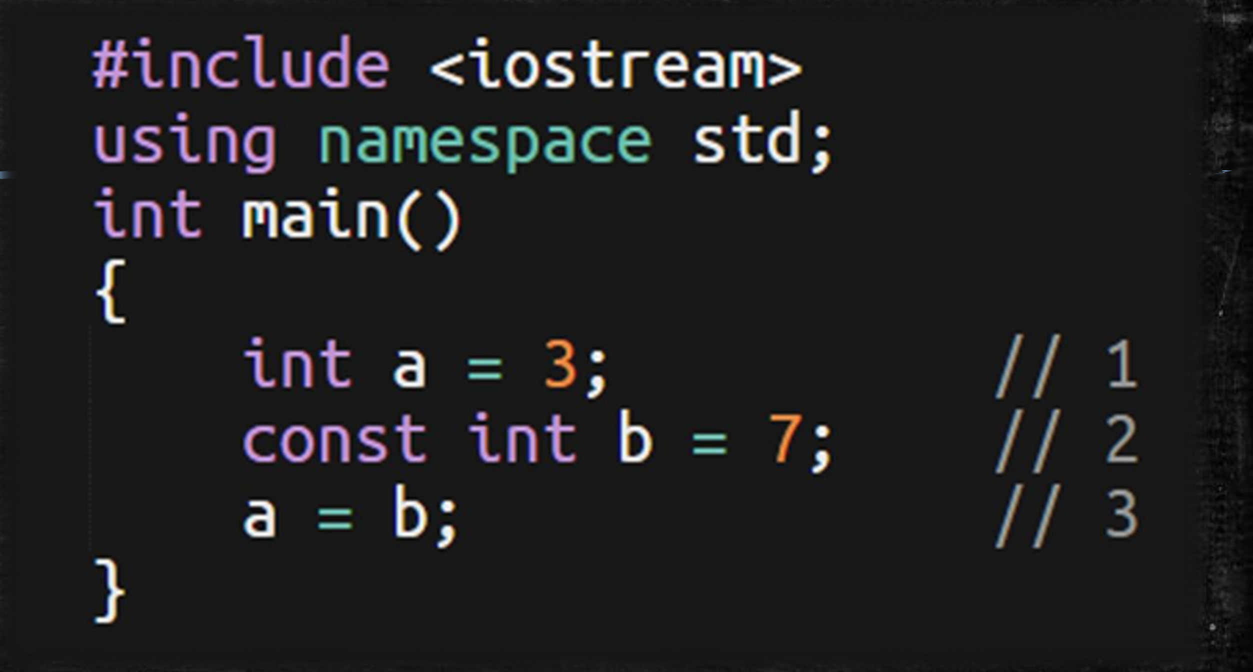


```
#include <iostream>
using namespace std;
int main()
{
    extern int a;
    cout << "a->" << a << endl;
}
```

- When built, this code creates an error.
  What is it?
  - A. Syntax (compiler) error: a is undeclared
  - B. Runtime error: a is uninitialized
  - C. Logic error: a is uninitialized
  - D. Linker error: a is undefined
  - E. I lied. There is no error at all.

Question

- On line 2, b is a:
  - A. Modifiable Lvalue
  - B. Non-modifiable Lvalue
  - C. Rvalue
  - D. Neither an LValue, nor an Rvalue



```cpp
#include <iostream>
using namespace std;
int main()
{
    int a = 3;              // 1
    const int b = 7;        // 2
    a = b;                  // 3
}
```

Input Review: Reading a number

- For numeric input use `cin` and extraction operator

```
    cin >> variable;
```

  - Read and discard leading whitespace characters
  - Match characters to the type of the variable
  - Stop reading when a mis-match occurs
  - Convert characters to binary data and store in variable
- What if input doesn't match the type?
  - `cin` goes into a fail state. No runtime exception, like Java
- What if there is no input? Program blocks (keyboard)

Question

- The user types: 23.57dogs
  What value is stored in a?

  – A. 23.57
  – B. .57
  – C. Nothing. A runtime error.
  – D. Nothing, but no runtime error (cin fails)
  – E. Still waiting for input

```
int n;
double a;
cin >> n >> a;
```

Question

- The user types: 23.57dogs
  What value is stored in n?
  - A. 23
  - B. 57
  - C. Nothing. A runtime error.
  - D. Nothing, but no runtime error (cin fails)
  - E. Still waiting (blocked) for input

```
int n;
double a;
cin >> a >> n;
```

Question

- The user types: `23.57dogs`
  What value is stored in s?
  - A. `23.57`dogs
  - B. dogs
  - C. Nothing. A runtime error.
  - D. Nothing, but no runtime error (`cin` fails)
  - E. Still waiting (blocked) for input

```
int n;
double a;
string s;
cin >> n >> a >> s;
```

Question

- The user types: `23.57dogs`
  What value is stored in a?
  - A. `23.57`
  - B. `.57`
  - C. Nothing. A runtime error.
  - D. Nothing, but no runtime error (`cin` fails)
  - E. Still waiting (blocked) for input

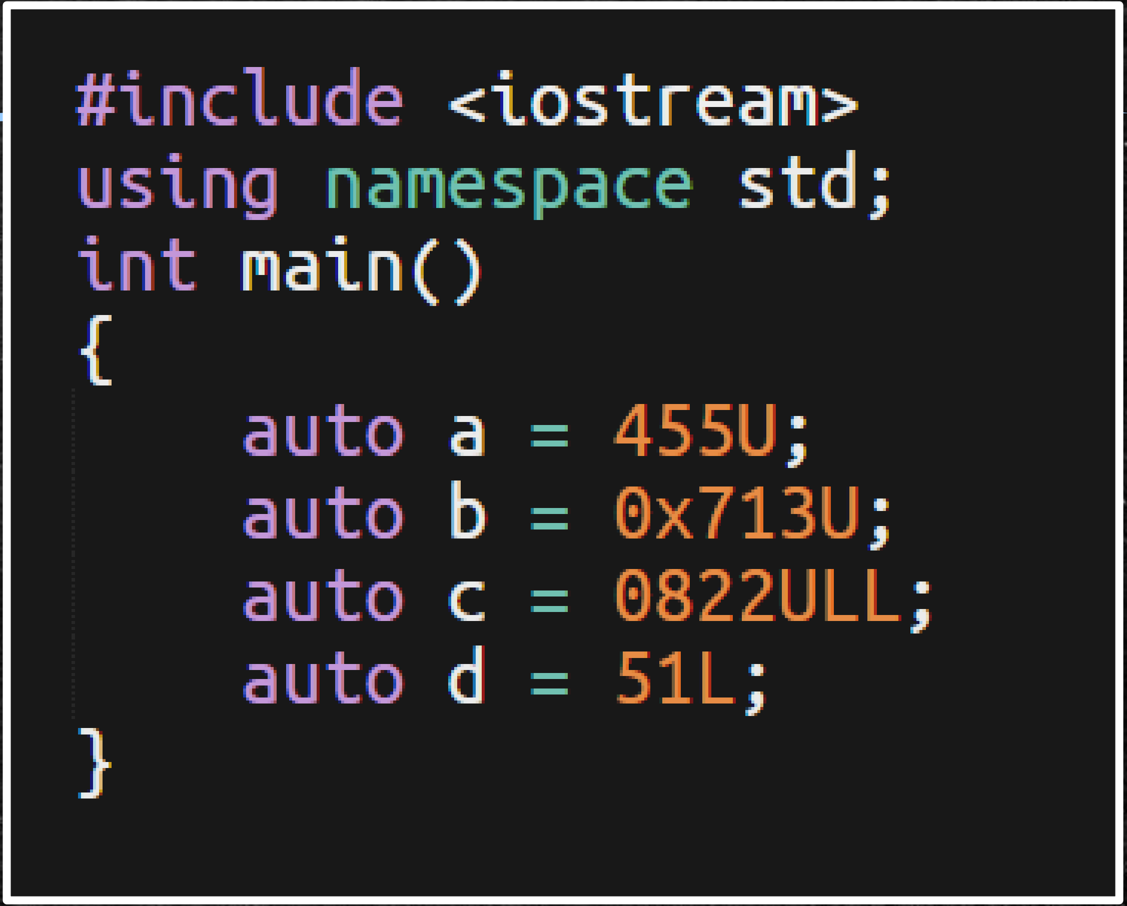
```
int n;
double a;
string s;
cin >> n >> s >> a;
```

Review: Numbers & Calculations

- Integers: `int`, `short`, `long`, `long long`
  - `signed`, `unsigned`, `signed` & `unsigned char`
  - Literals: `010`, `0x10`, `0b10`, `23L`, `23LL`, `23U`
- Floating-point: `float`, `double`, `long double`
  - Literals: 2.F, `.2`, `2e-3`
- Arithmetic operators: `+`, `-`, `*`, `/`, `%`
  - Arity, precedence & associativity
- Side-effect operators: `=`, `+=`, `++`, `--`
  - Prefix: (++a) (lvalue) & postfix (a++) (not lvalue)

Question

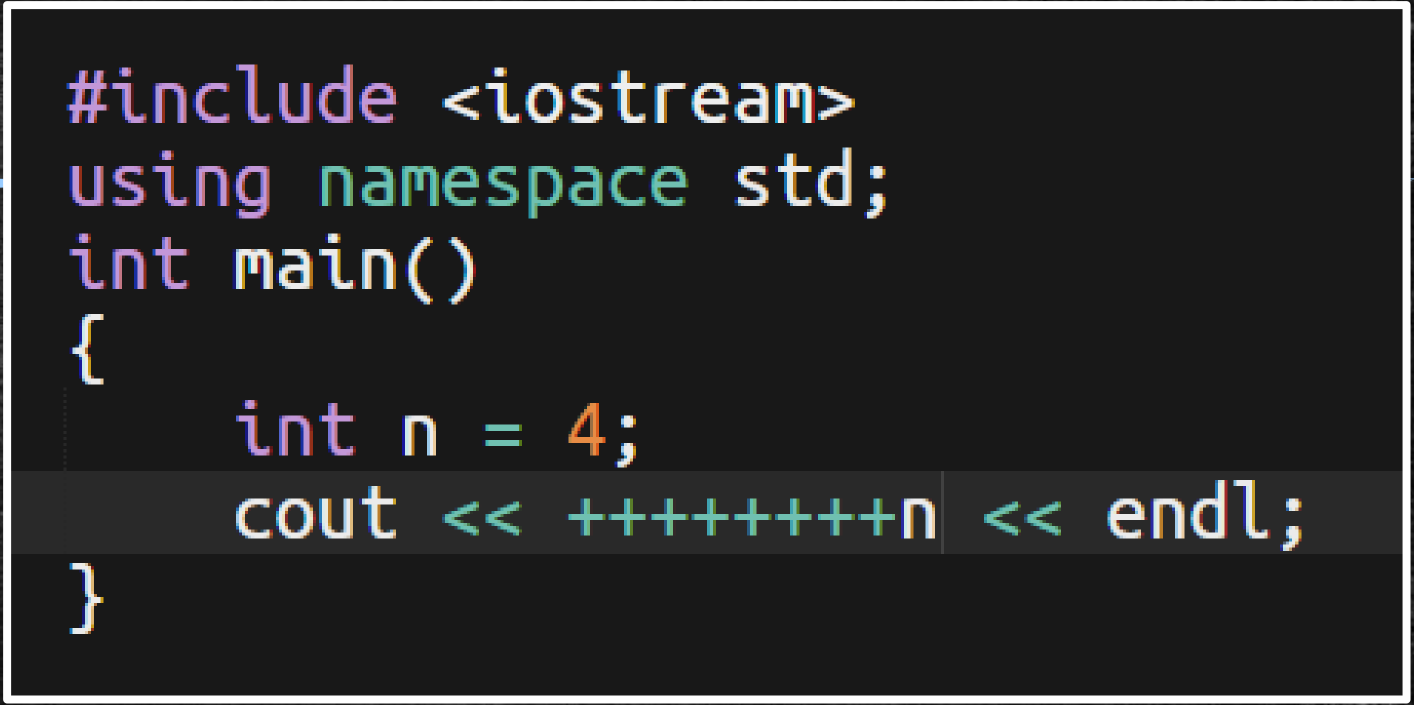
- Which initialization fails
  - A. a
  - B. b
  - C. c
  - D. d
  - E. None of these



```
#include <iostream>
using namespace std;
int main()
{
        auto a = 455U;
        auto b = 0x713U;
        auto c = 0822ULL;
        auto d = 51L;
}
```

Question

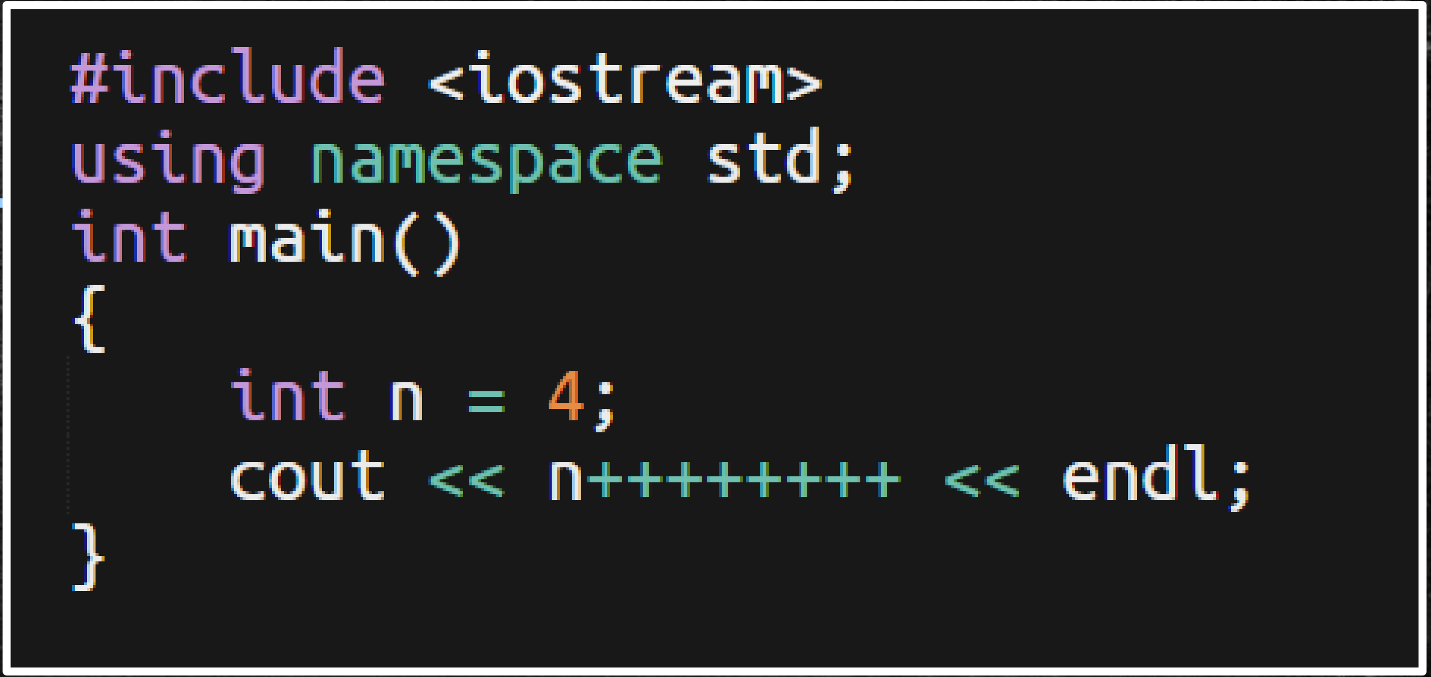
- What is printed when this runs?
  - A. Syntax error. Does not compile.
  - B. 7
  - C. 8
  - D. 4
  - E. Undefined output



```
#include <iostream>
using namespace std;
int main()
{
    int n = 4;
    cout << ++++++++n << endl;
}
```

Question

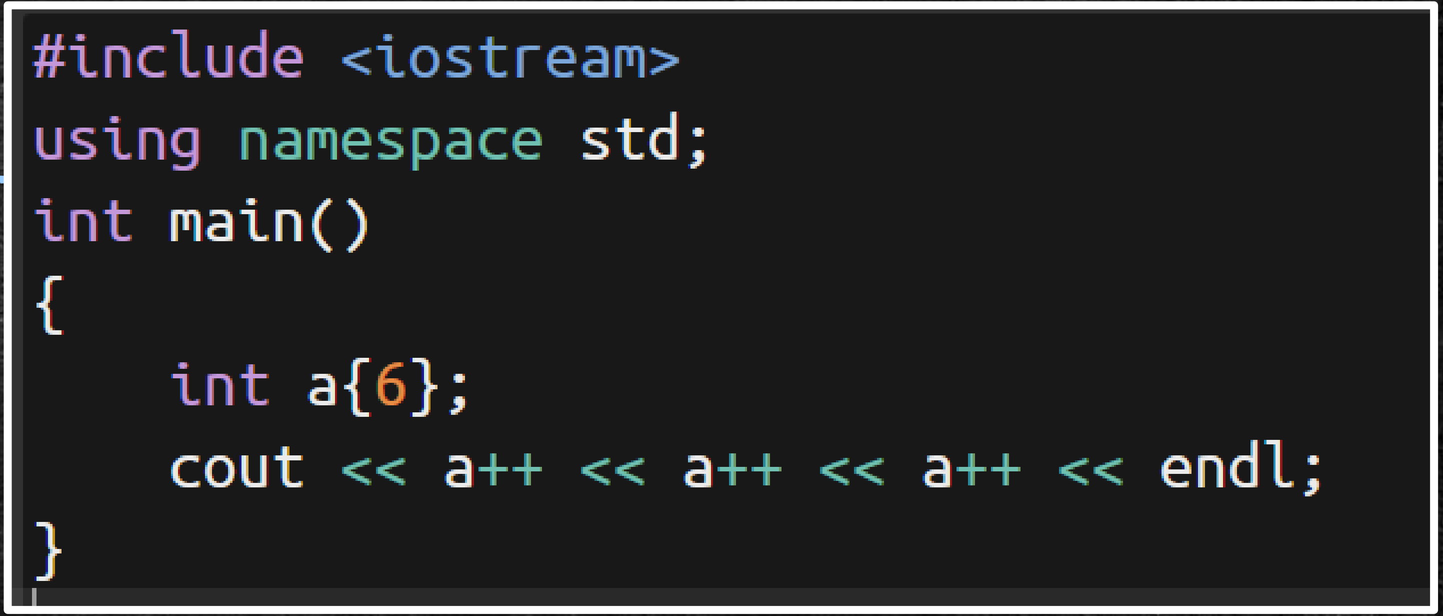
- What is printed when this runs?
  - A. Syntax error. Does not compile.
  - B. 7
  - C. 8
  - D. 4
  - E. Undefined output



```cpp
#include <iostream>
using namespace std;
int main()
{
    int n = 4;
    cout << n++++++++ << endl;
}
```

Question

- What is the output when running this program?
  - A. 678
  - B. 876
  - C. 666
  - D. All of these are possible; behavior is undefined
  - E. Does not compile



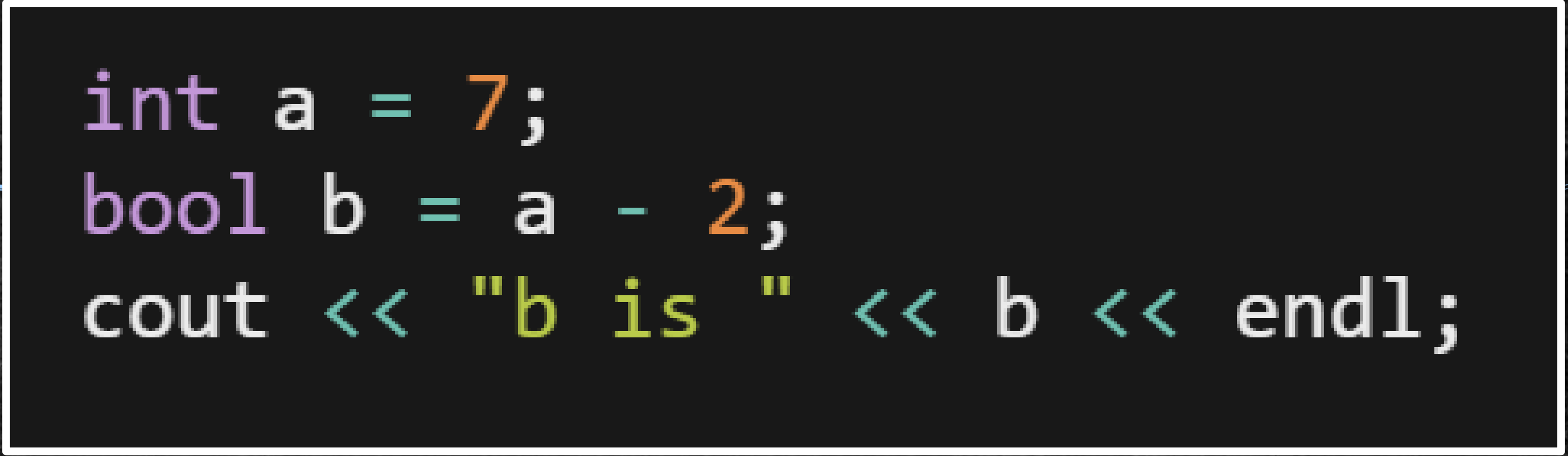
```
#include <iostream>
using namespace std;
int main()
{
    int a{6};
    cout << a++ << a++ << a++ << endl;
}
```

Selection Review

- Selection (branching) is conditional execution
  - Relational; compare values: `<`, `<=`, `>`, `>=`, `==`, `!=`
  - Boolean or Logical: `!` (**not**), `||` (**or**), `&&` (**and**)
  - Pitfalls: conversion, embedded assignment, impossible and unavoidable conditions, real comparisons
- Five types of selection statements
  - Independent, sequential, nested, numbered, conditional
- Functions: arguments, parameters, return types
  - Calling and defining

Question

- What prints?

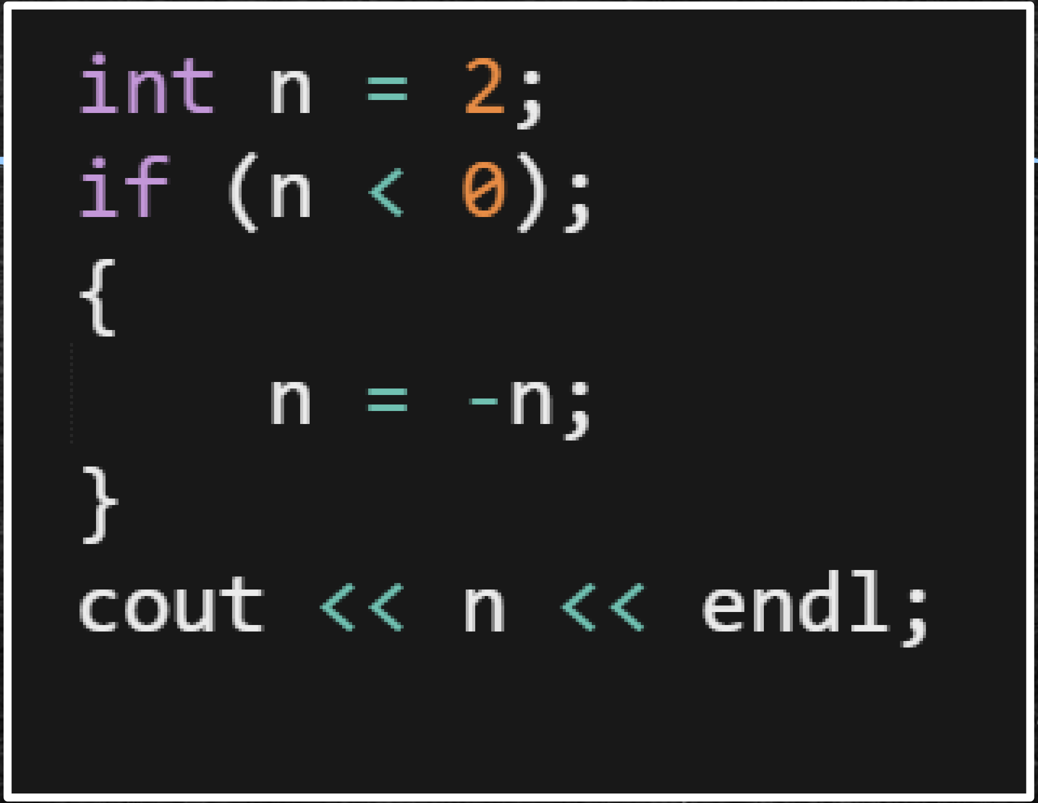


```
int a = 7;
bool b = a - 2;
cout << "b is " << b << endl;
```

  - A. Syntax error! Wrong type for b.
  - B. `b is true`
  - C. `b is false`
  - D. `b is 5`
  - E. `b is 1`

Question

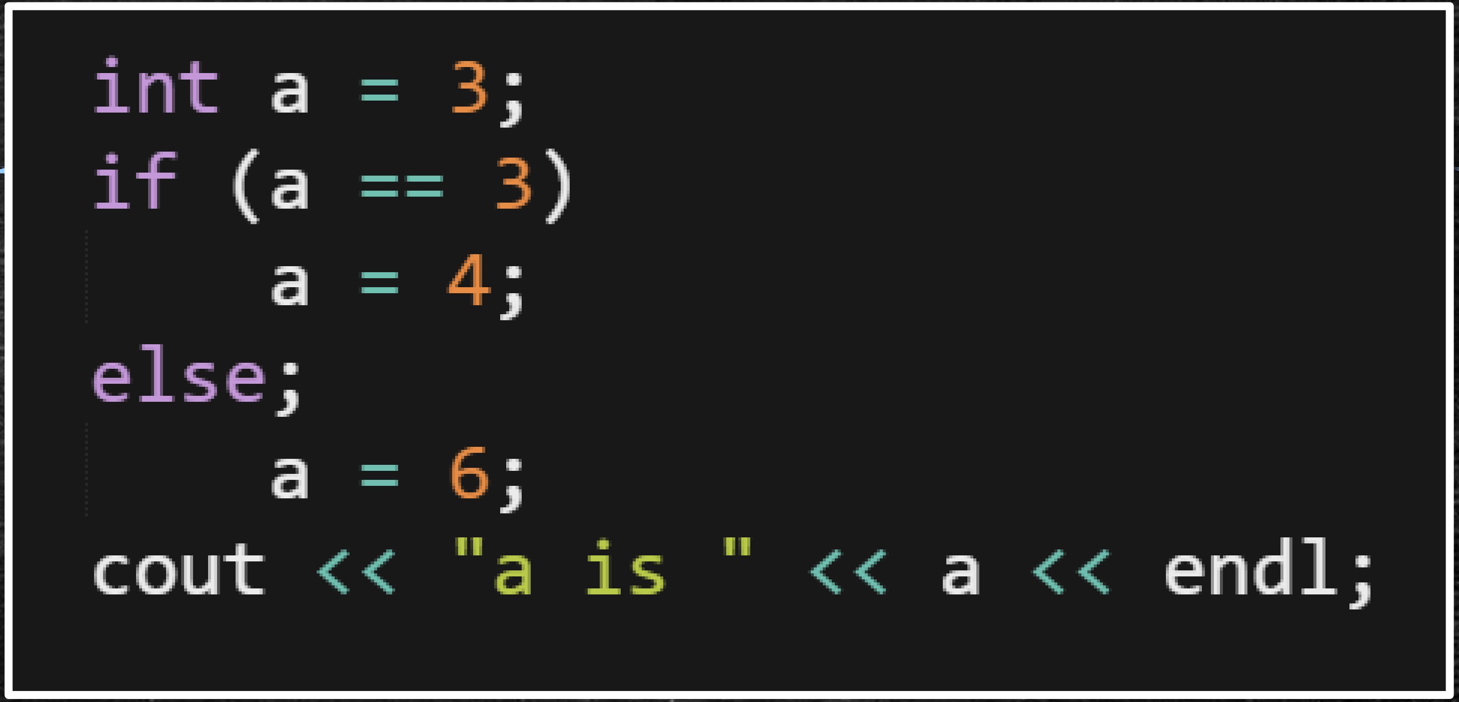
- What prints?
  - A. syntax error
  - B. runtime error
  - C. **2**
  - D. **-2**
  - E. None of these



```
int n = 2;
if (n < 0);
{
    n = -n;
}
cout << n << endl;
```

Question

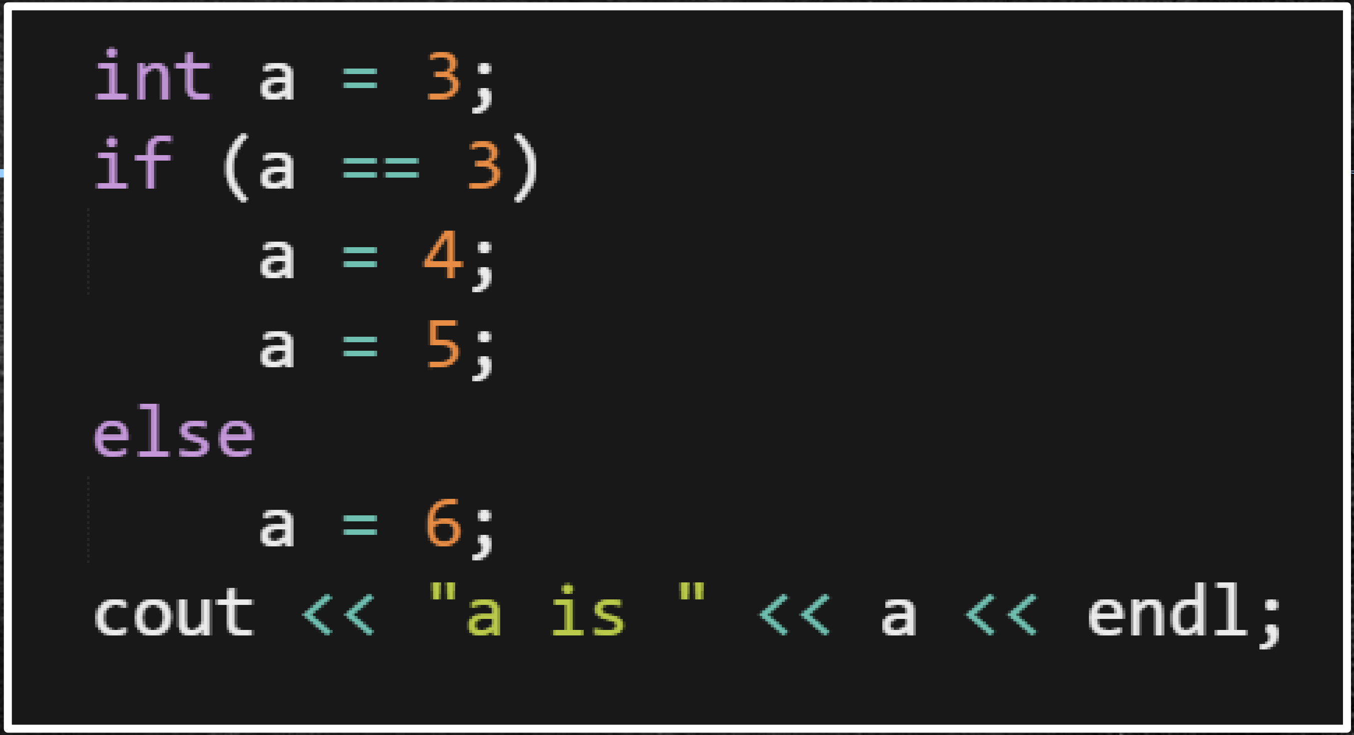
- What prints?
  - A. `a is 3`
  - B. `a is 4`
  - C. `a is 6`
  - D. Syntax error



```
int a = 3;
if (a == 3)
    a = 4;
else;
    a = 6;
cout << "a is " << a << endl;
```

Question

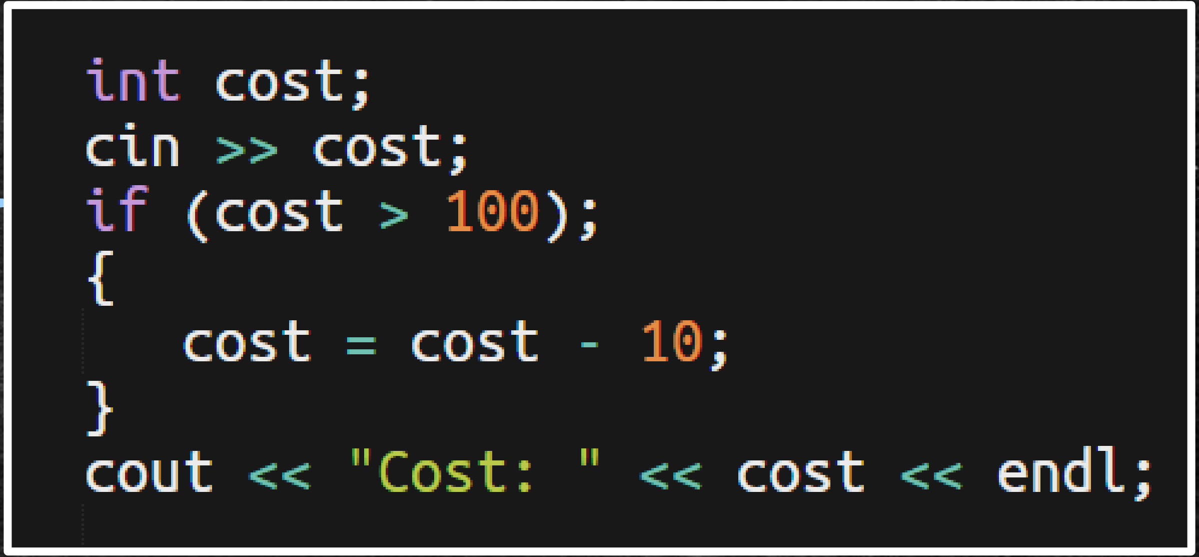
- What prints?
  - A. `a is 4`
  - B. `a is 5`
  - C. `a is 6`
  - D. Syntax error



```
int a = 3;
if (a == 3)
    a = 4;
    a = 5;
else
    a = 6;
cout << "a is " << a << endl;
```

Question

- What is the error in this code snippet?

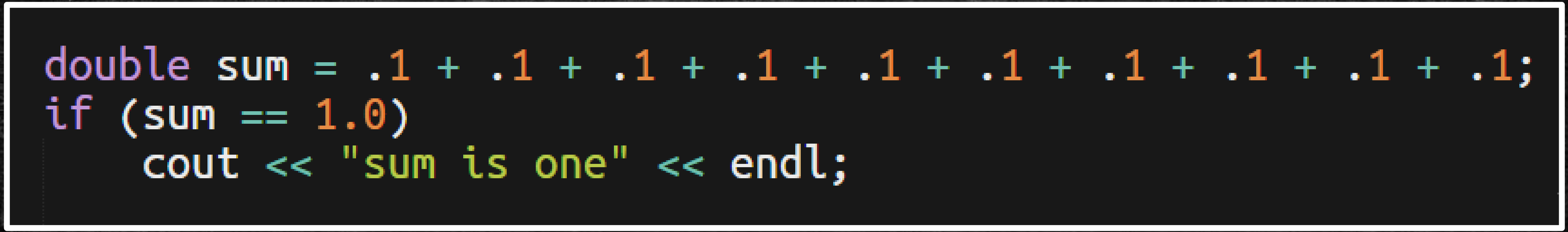


```cpp
int cost;
cin >> cost;
if (cost > 100);
{
    cost = cost - 10;
}
cout << "Cost: " << cost << endl;
```

  - A. Syntax error (won't compile)
  - B. Logic error: uninitialized variable
  - C. Logic error: body of if statement is empty
  - D. Logic error: assignment does not show equality
  - E. Runtime error: program crashes when run

Question

- What is the problem with this?



```cpp
double sum = .1 + .1 + .1 + .1 + .1 + .1 + .1 + .1 + .1 + .1;
if (sum == 1.0)
    cout << "sum is one" << endl;
```

  - A. Nothing: it prints "sum is one"
  - B. Syntax error: missing braces around the body.
  - C. Logic error: using assignment in an if condition
  - D. Logic error: do not use == with floating point
  - E. Logic error: never use literals in conditions

Question

```
cout << "Hi" && cout << "Bye" << endl;
```

- What prints here?
  - A. Nothing. Compilation error.
  - B. Nothing (but compiles and runs)
  - C. Hi
  - D. Bye + newline
  - E. HiBye + newline

Lesson 2A Preview - Strings

- C++ string objects: a library type similar to String in Java
  - Operations: concatenate & compare
  - Mutability and value assignment
  - Member functions: `size()` and `size_t`
  - Selection: `front()`, `back()`, `at()` and `[]`
  - Substrings and searching: `substr()`, `find()`, `rfind()`
- Reference types (opposed to Python and Java references)
  - `int a = 42`, `b = 75;`
  - `int& c = a; // just a second name for a`
  - Reference and `const` reference parameters

Lesson 2B Preview - Loops

- Loop concepts (how are loops and selection similar?)
- Types of loops:
  - Guarded and unguarded loops
  - Range-based, definite and indefinite loops
  - Types of indefinite loops: counter, sentinel, data, limit
- Using range-based loops to process strings
- Using conventional *for* loops to process strings
  - Character-by-character and substring by substring
- Using *for* loops to generate data

Lesson 2C Preview - More on Loops

- Six steps to writing loops effectively
  - Bounds, bounds precondition, advance, goal precondition, operation, postcondition
- Loop guards and intentional and necessary bounds
- Sentinel loop patterns
  - Writing a primed sentinel loop
  - Using the flag-controlled pattern
  - Using a loop-and-a-half
- Validating data

Lesson 2D Preview - Function Libraries

- Separate compilation and multi-file projects
  - Writing the client or test program
  - Putting the prototypes in the header or interface file
    - Adding header guards and why you need them
  - Implementing the code in the `.cpp` file
  - Writing a `makefile` to build your executable
- Documenting the interface using Doxygen
- Using *`while`* loops with limit bounds

Week 2 Homework Preview

- Homework is due by next Tuesday before class
- `H05` - Le, La, L', Les: writing string functions
  - Convert country names to their correct French gender
- `H06` - Strings, loops and numbers
  - A Codingbat problem, *`sumNumbers`*
- `H07` - Writing string libraries
  - Several more Codingbat problems that use strings
- `H08` - Going Postal - a bar-code library

Programming Exams 1 and 2

- Now – Programming Exam #1
  - I will collect your cellphones, watches & electronics
  - Place all books, backpacks, notes at front or back of the room
  - Move to your assigned seat; do not log in
  - I will start PE01 on your computer
  - Log in using your Homework Console credentials
  - When you are done, submit the exam and leave
- Come back by 4pm when PE02 will start