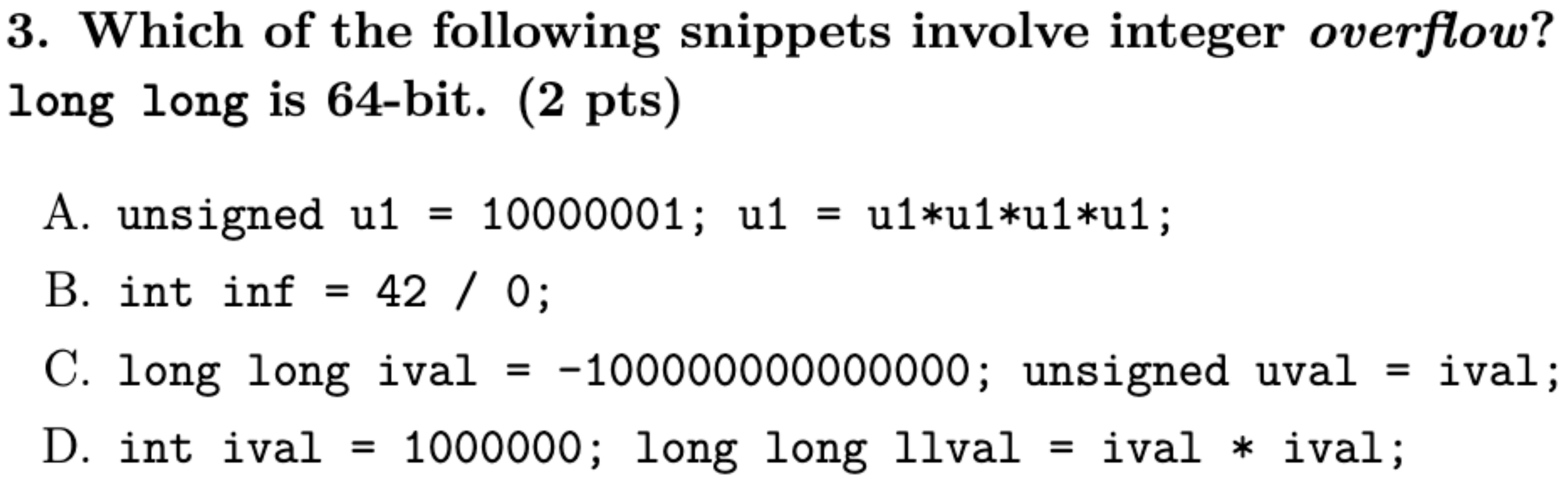
Recitation Class

Week 2

Homeworks



3. Which of the following snippets involve integer *overflow*? Assume `int` is 32-bit, `long long` is 64-bit. (2 pts)

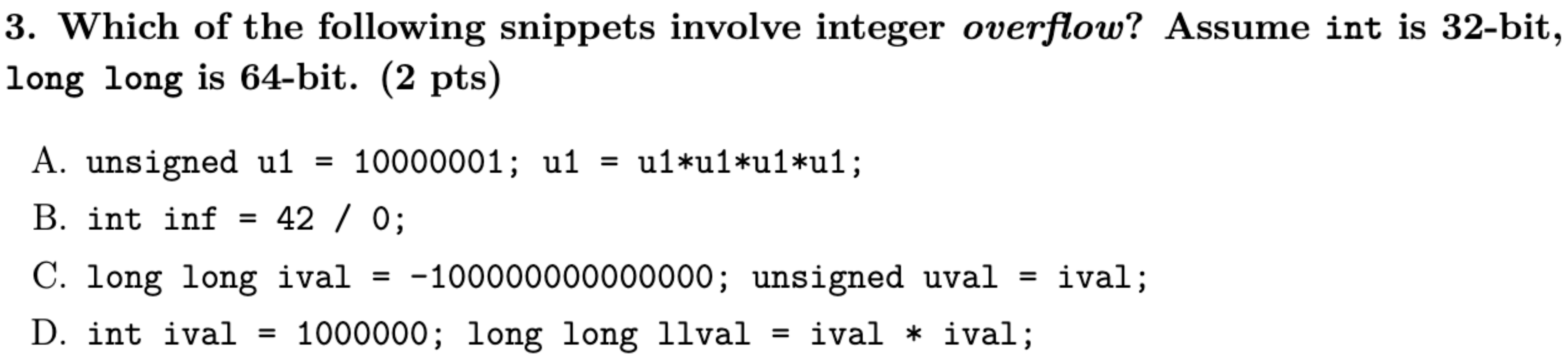
A. `unsigned u1 = 10000001; u1 = u1*u1*u1*u1;`

B. `int inf = 42 / 0;`

C. `long long ival = -100000000000000; unsigned uval = ival;`

D. `int ival = 1000000; long long llval = ival * ival;`

Homeworks



3. Which of the following snippets involve integer *overflow*? Assume `int` is 32-bit, `long long` is 64-bit. (2 pts)

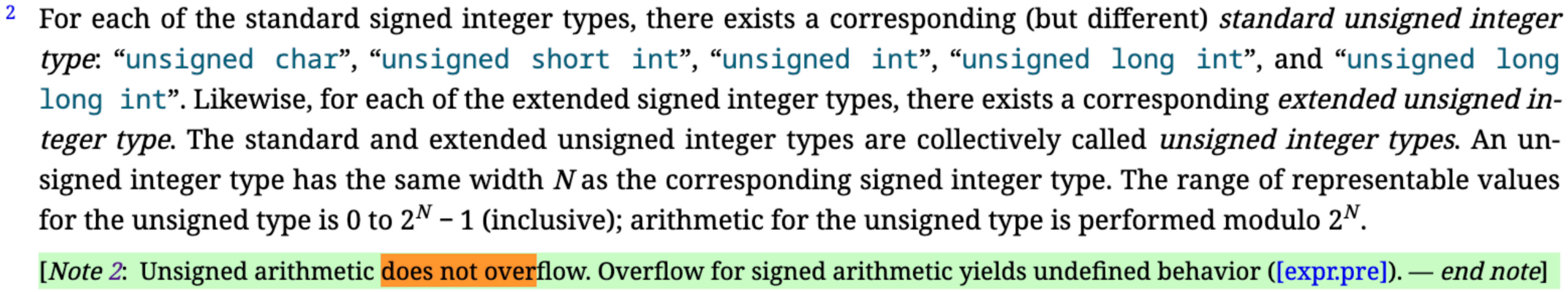
A. `unsigned u1 = 10000001; u1 = u1*u1*u1*u1;`

B. `int inf = 42 / 0;`

C. `long long ival = -100000000000000; unsigned uval = ival;`

D. `int ival = 1000000; long long llval = ival * ival;`

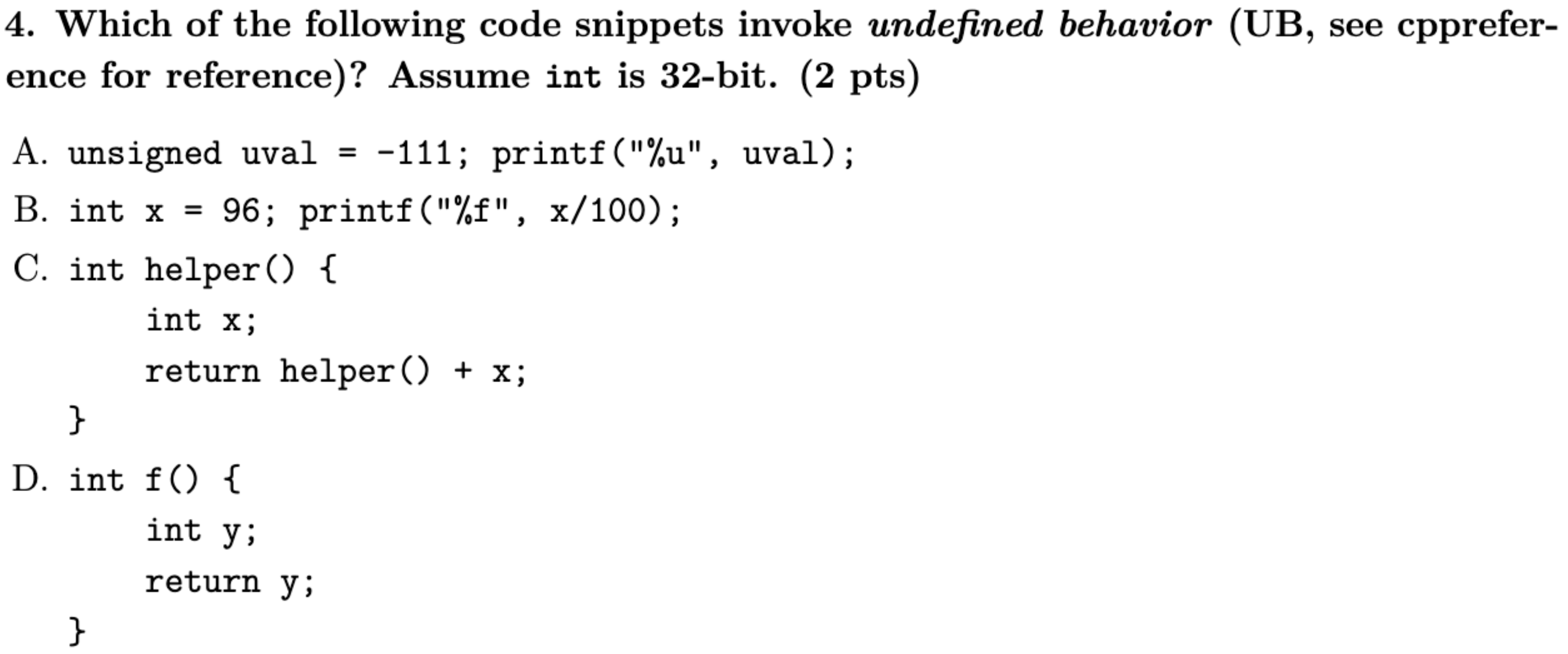
- Signed integer overflow is **undefined behavior**.
- Unsigned arithmetic **never overflows**: It is performed modulo 2^N, where N is the number of bits of that type.



[2] For each of the standard signed integer types, there exists a corresponding (but different) *standard unsigned integer type*: "`unsigned char`", "`unsigned short int`", "`unsigned int`", "`unsigned long int`", and "`unsigned long long int`". Likewise, for each of the extended signed integer types, there exists a corresponding *extended unsigned integer type*. The standard and extended unsigned integer types are collectively called *unsigned integer types*. An unsigned integer type has the same width N as the corresponding signed integer type. The range of representable values for the unsigned type is 0 to $2^N - 1$ (inclusive); arithmetic for the unsigned type is performed modulo 2^N.

[*Note 2*: Unsigned arithmetic does not overflow. Overflow for signed arithmetic yields undefined behavior ([expr.pre]). — *end note*]

Homeworks



4. Which of the following code snippets invoke *undefined behavior* (UB, see cppreference for reference)? Assume `int` is 32-bit. (2 pts)

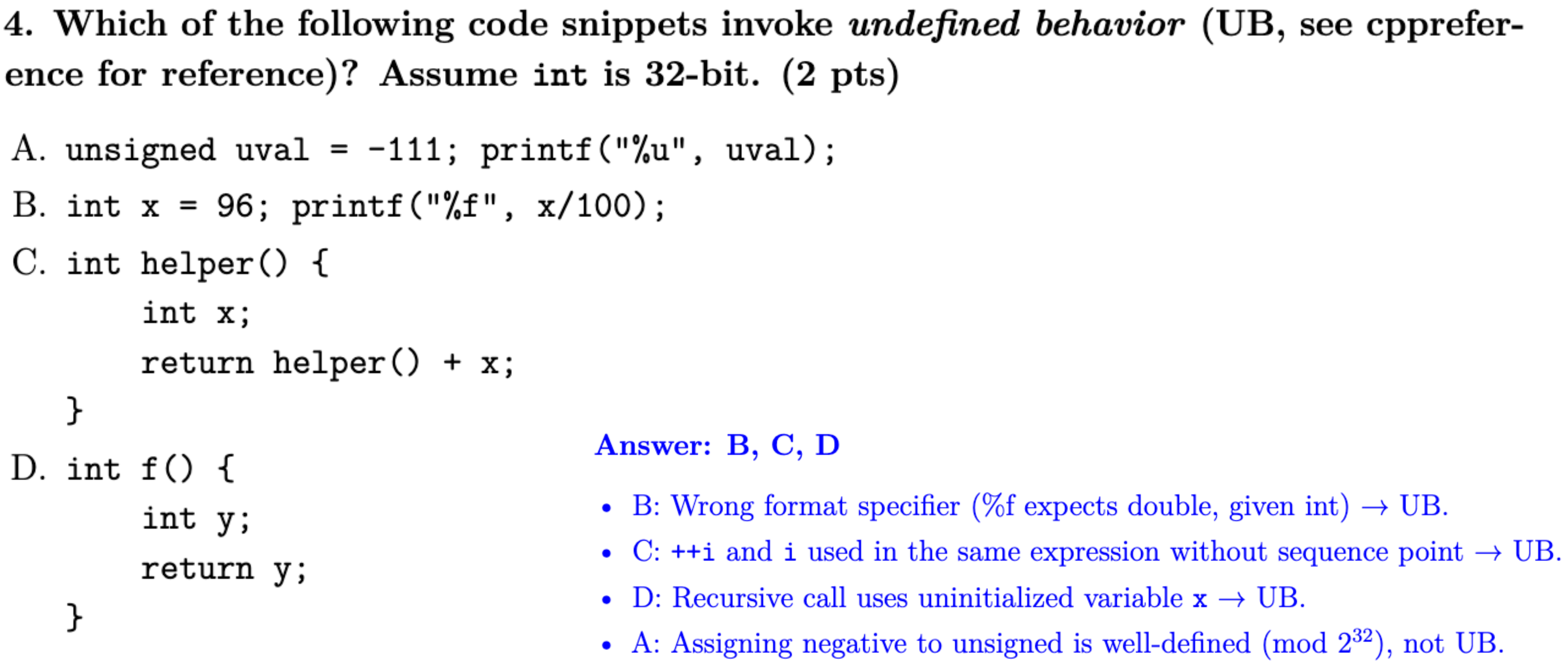
A. `unsigned uval = -111; printf("%u", uval);`

B. `int x = 96; printf("%f", x/100);`

C.
```
int helper() {
    int x;
    return helper() + x;
}
```

D.
```
int f() {
    int y;
    return y;
}
```

Homeworks



4. Which of the following code snippets invoke *undefined behavior* (UB, see cppreference for reference)? Assume `int` is 32-bit. (2 pts)

A. `unsigned uval = -111; printf("%u", uval);`

B. `int x = 96; printf("%f", x/100);`

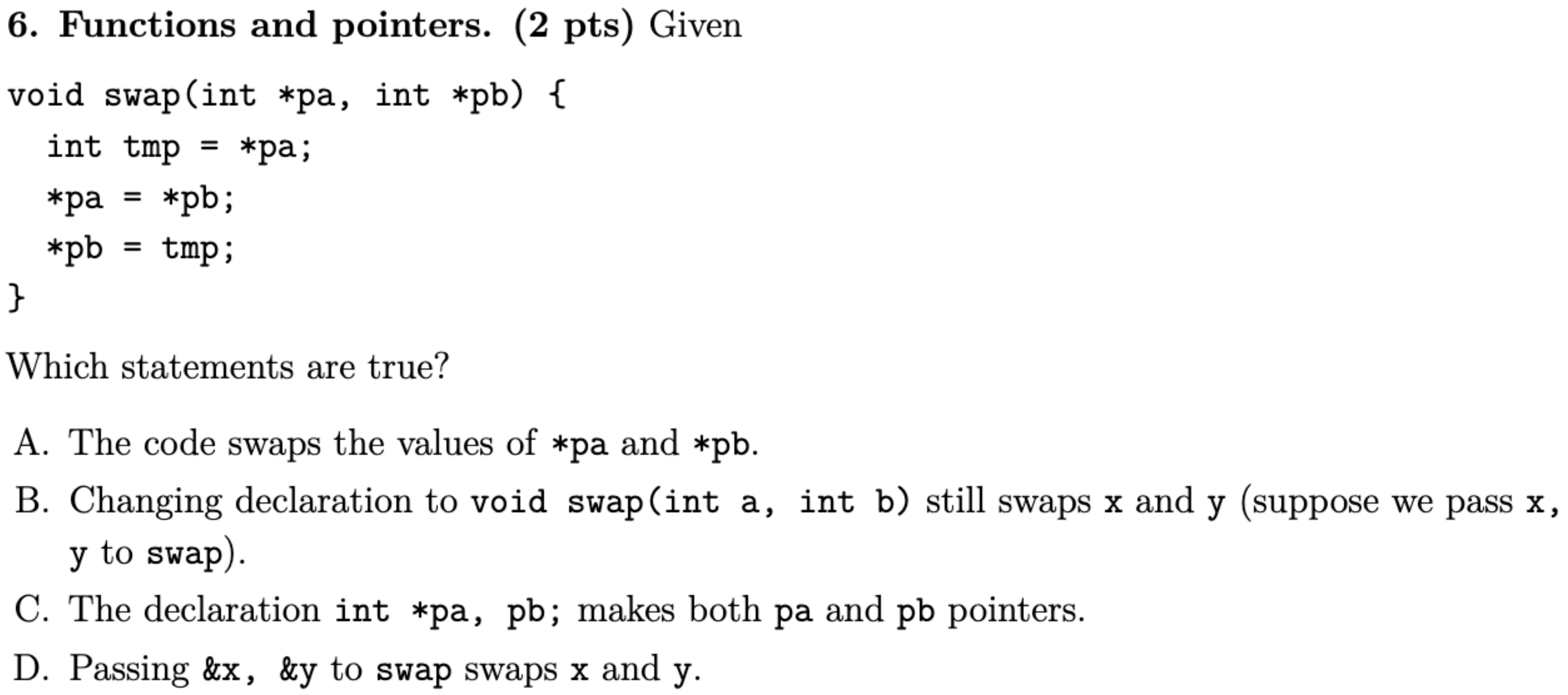
C.
```
int helper() {
    int x;
    return helper() + x;
}
```

D.
```
int f() {
    int y;
    return y;
}
```

Answer: B, C, D

- B: Wrong format specifier (%f expects double, given int) → UB.
- C: `++i` and `i` used in the same expression without sequence point → UB.
- D: Recursive call uses uninitialized variable **`x`** → UB.
- A: Assigning negative to unsigned is well-defined (mod 2^{32}), not UB.

Homework



6. Functions and pointers. (2 pts) Given

```
void swap(int *pa, int *pb) {
  int tmp = *pa;
  *pa = *pb;
  *pb = tmp;
}
```

Which statements are true?

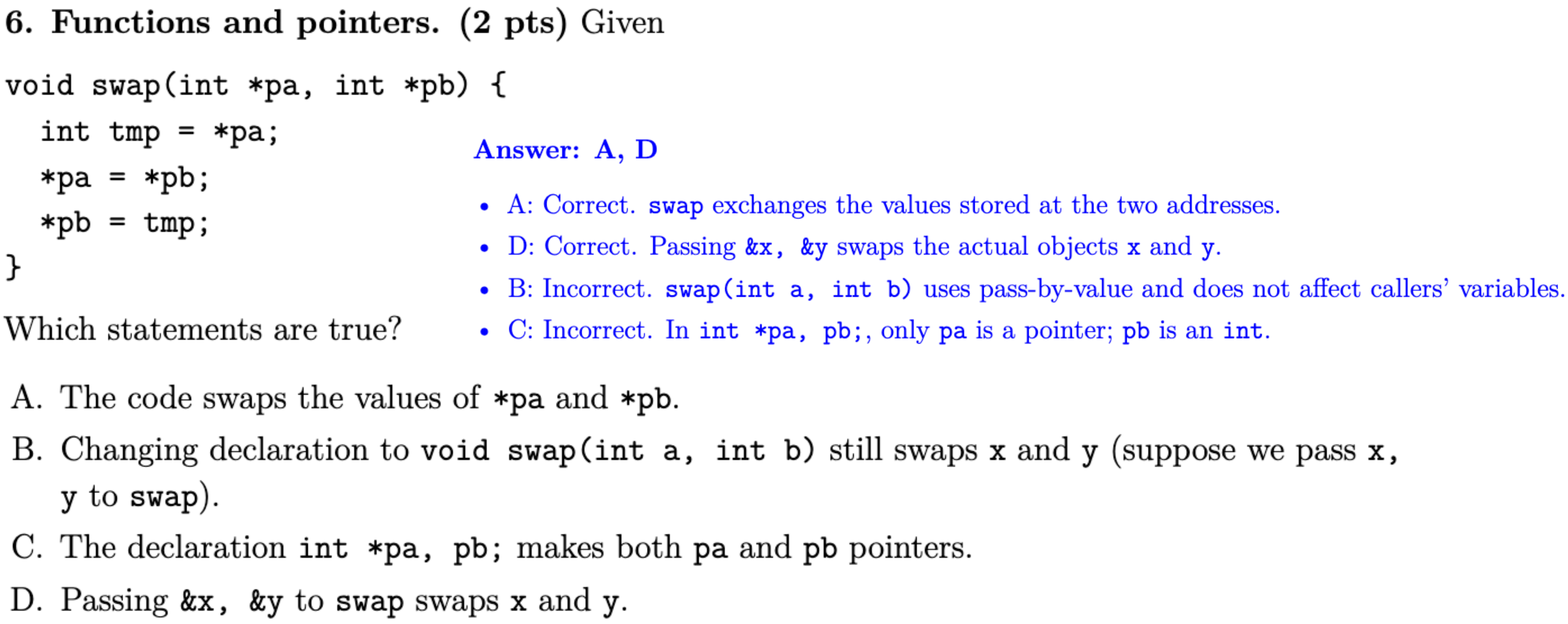
A. The code swaps the values of `*pa` and `*pb`.

B. Changing declaration to `void swap(int a, int b)` still swaps `x` and `y` (suppose we pass `x, y` to `swap`).

C. The declaration `int *pa, pb;` makes both `pa` and `pb` pointers.

D. Passing `&x, &y` to `swap` swaps `x` and `y`.

Homework



6. Functions and pointers. (2 pts) Given

```
void swap(int *pa, int *pb) {
  int tmp = *pa;
  *pa = *pb;
  *pb = tmp;
}
```

Which statements are true?

A. The code swaps the values of `*pa` and `*pb`.

B. Changing declaration to `void swap(int a, int b)` still swaps `x` and `y` (suppose we pass `x, y` to `swap`).

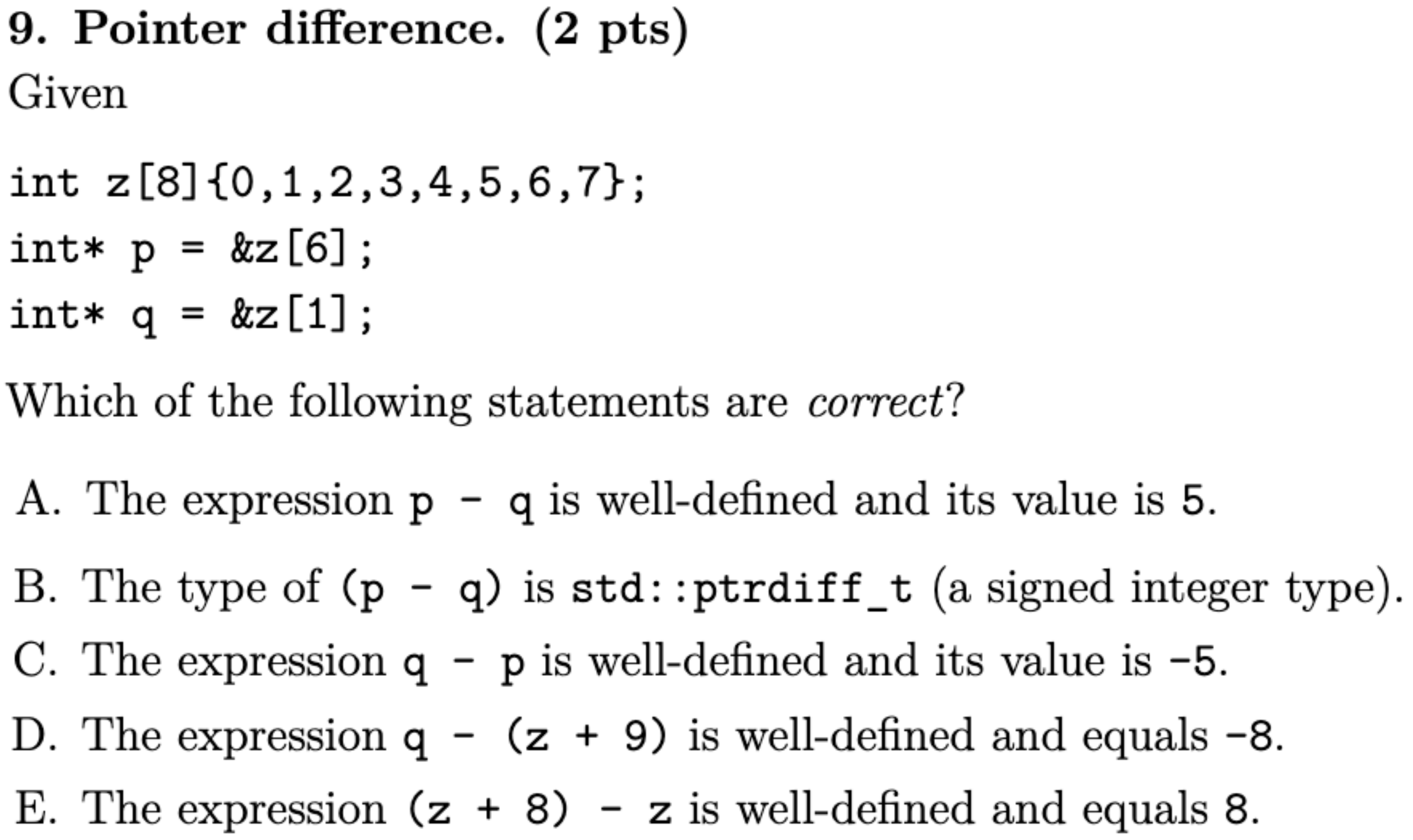
C. The declaration `int *pa, pb;` makes both `pa` and `pb` pointers.

D. Passing `&x, &y` to `swap` swaps `x` and `y`.

Answer: A, D

- A: Correct. `swap` exchanges the values stored at the two addresses.
- D: Correct. Passing `&x, &y` swaps the actual objects `x` and `y`.
- B: Incorrect. `swap(int a, int b)` uses pass-by-value and does not affect callers' variables.
- C: Incorrect. In `int *pa, pb;`, only `pa` is a pointer; `pb` is an `int`.

Homework



9. Pointer difference. (2 pts)

Given

```
int z[8]{0,1,2,3,4,5,6,7};
int* p = &z[6];
int* q = &z[1];
```

Which of the following statements are *correct*?

A. The expression `p - q` is well-defined and its value is `5`.

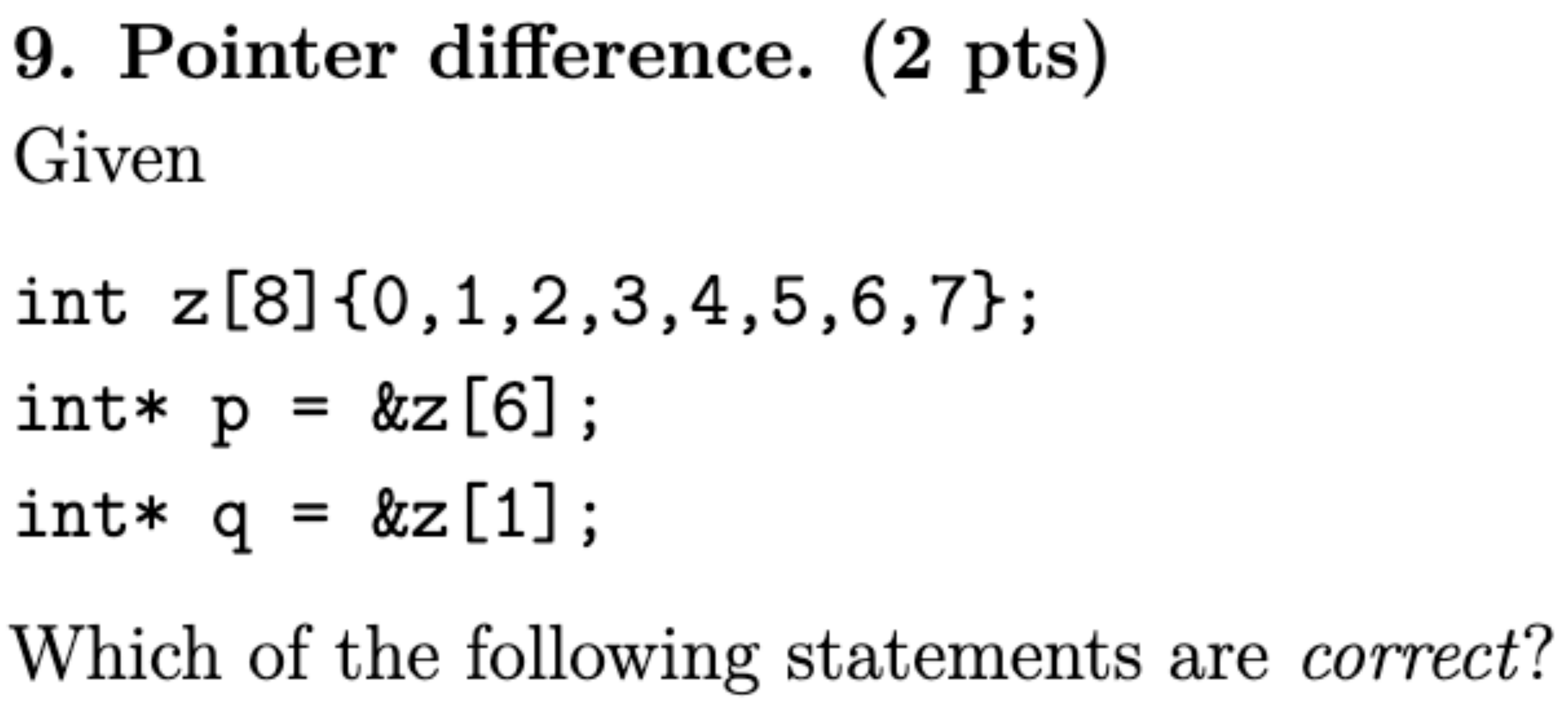
B. The type of `(p - q)` is `std::ptrdiff_t` (a signed integer type).

C. The expression `q - p` is well-defined and its value is `-5`.

D. The expression `q - (z + 9)` is well-defined and equals `-8`.

E. The expression `(z + 8) - z` is well-defined and equals `8`.

Homework

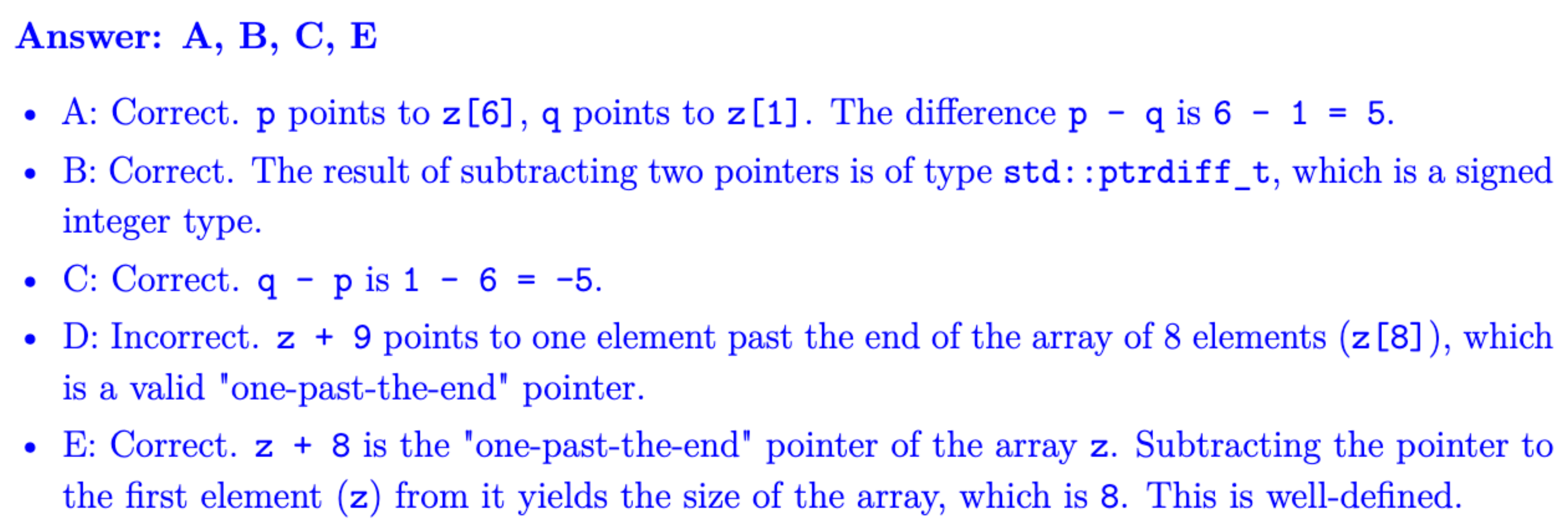


9. Pointer difference. (2 pts)

Given

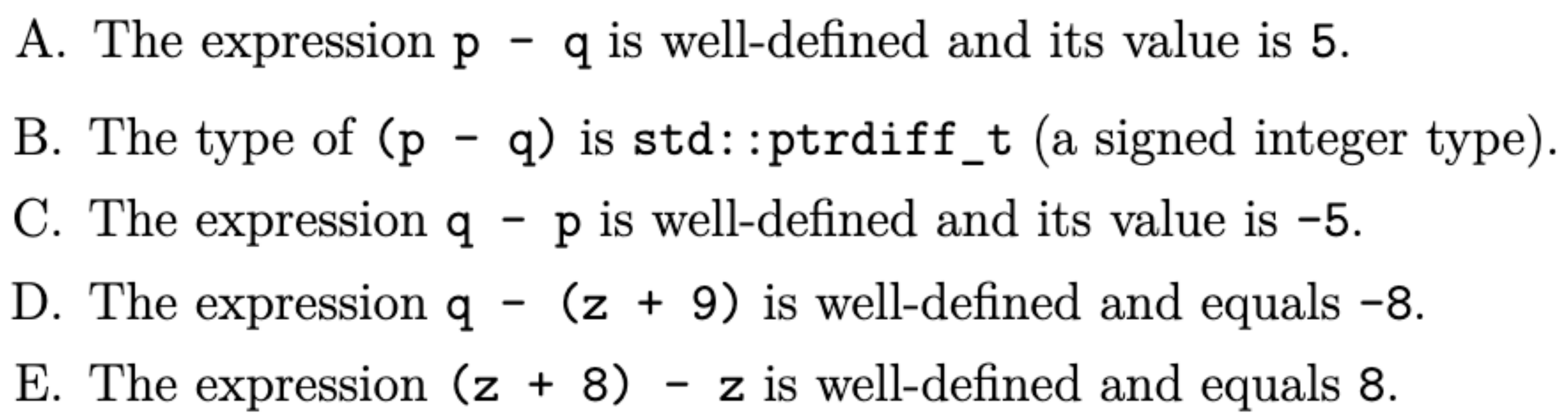
```
int z[8]{0,1,2,3,4,5,6,7};
int* p = &z[6];
int* q = &z[1];
```

Which of the following statements are *correct*?



Answer: A, B, C, E

- A: Correct. `p` points to `z[6]`, `q` points to `z[1]`. The difference `p - q` is `6 - 1 = 5`.
- B: Correct. The result of subtracting two pointers is of type `std::ptrdiff_t`, which is a signed integer type.
- C: Correct. `q - p` is `1 - 6 = -5`.
- D: Incorrect. `z + 9` points to one element past the end of the array of 8 elements (`z[8]`), which is a valid "one-past-the-end" pointer.
- E: Correct. `z + 8` is the "one-past-the-end" pointer of the array `z`. Subtracting the pointer to the first element (`z`) from it yields the size of the array, which is `8`. This is well-defined.



A. The expression `p - q` is well-defined and its value is `5`.

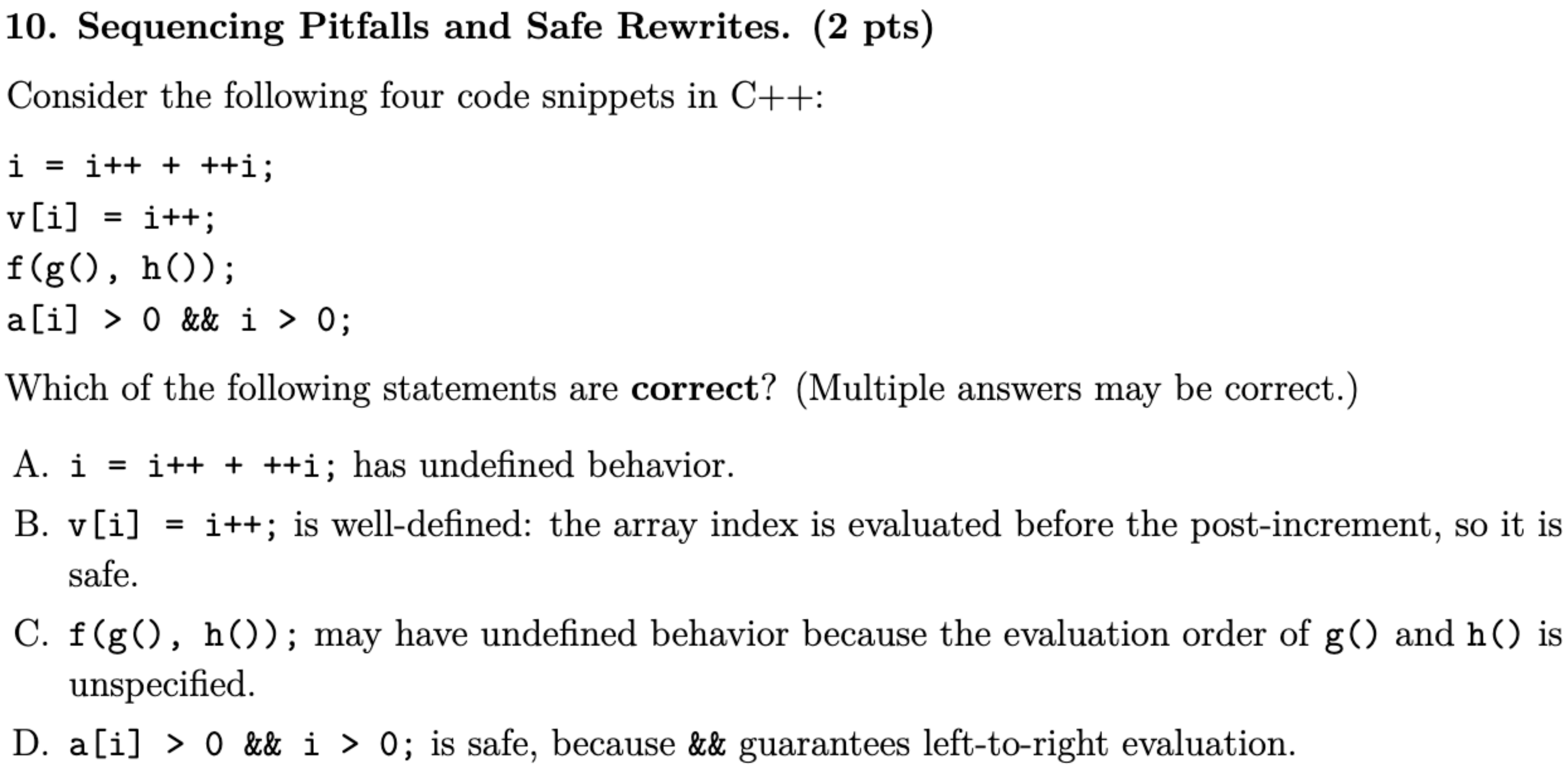
B. The type of `(p - q)` is `std::ptrdiff_t` (a signed integer type).

C. The expression `q - p` is well-defined and its value is `-5`.

D. The expression `q - (z + 9)` is well-defined and equals `-8`.

E. The expression `(z + 8) - z` is well-defined and equals `8`.

Homework



10. Sequencing Pitfalls and Safe Rewrites. (2 pts)

Consider the following four code snippets in C++:

```
i = i++ + ++i;
v[i] = i++;
f(g(), h());
a[i] > 0 && i > 0;
```

Which of the following statements are **correct**? (Multiple answers may be correct.)

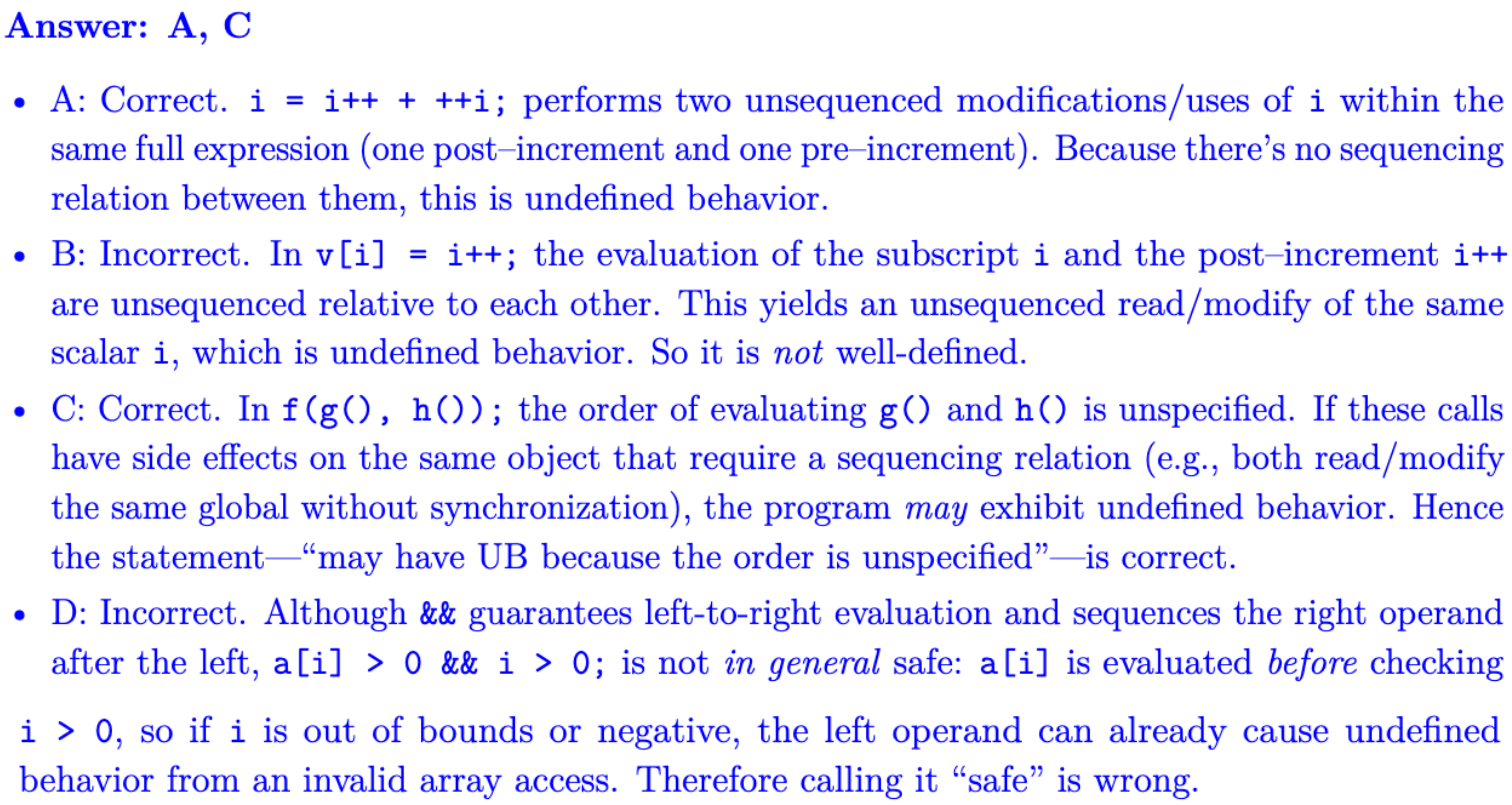
A. `i = i++ + ++i;` has undefined behavior.

B. `v[i] = i++;` is well-defined: the array index is evaluated before the post-increment, so it is safe.

C. `f(g(), h());` may have undefined behavior because the evaluation order of `g()` and `h()` is unspecified.

D. `a[i] > 0 && i > 0;` is safe, because `&&` guarantees left-to-right evaluation.

Homework



Answer: A, C

- A: Correct. `i = i++ + ++i;` performs two unsequenced modifications/uses of `i` within the same full expression (one post–increment and one pre–increment). Because there's no sequencing relation between them, this is undefined behavior.
- B: Incorrect. In `v[i] = i++;` the evaluation of the subscript `i` and the post–increment `i++` are unsequenced relative to each other. This yields an unsequenced read/modify of the same scalar `i`, which is undefined behavior. So it is *not* well-defined.
- C: Correct. In `f(g(), h());` the order of evaluating `g()` and `h()` is unspecified. If these calls have side effects on the same object that require a sequencing relation (e.g., both read/modify the same global without synchronization), the program *may* exhibit undefined behavior. Hence the statement—"may have UB because the order is unspecified"—is correct.
- D: Incorrect. Although `&&` guarantees left-to-right evaluation and sequences the right operand after the left, `a[i] > 0 && i > 0;` is not *in general* safe: `a[i]` is evaluated *before* checking

`i > 0`, so if `i` is out of bounds or negative, the left operand can already cause undefined behavior from an invalid array access. Therefore calling it "safe" is wrong.

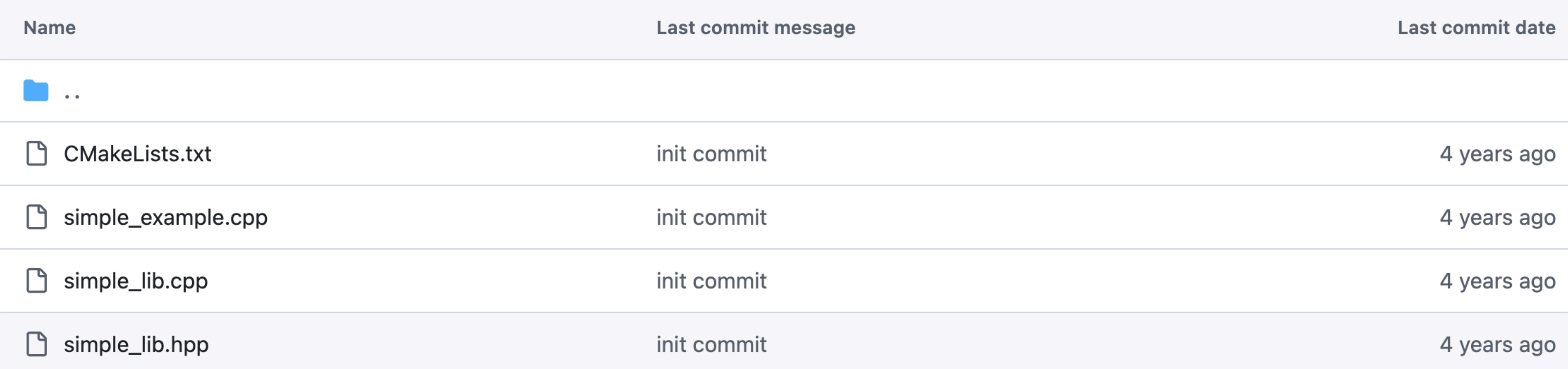
CMake

Why Do I Need a Good Build System?

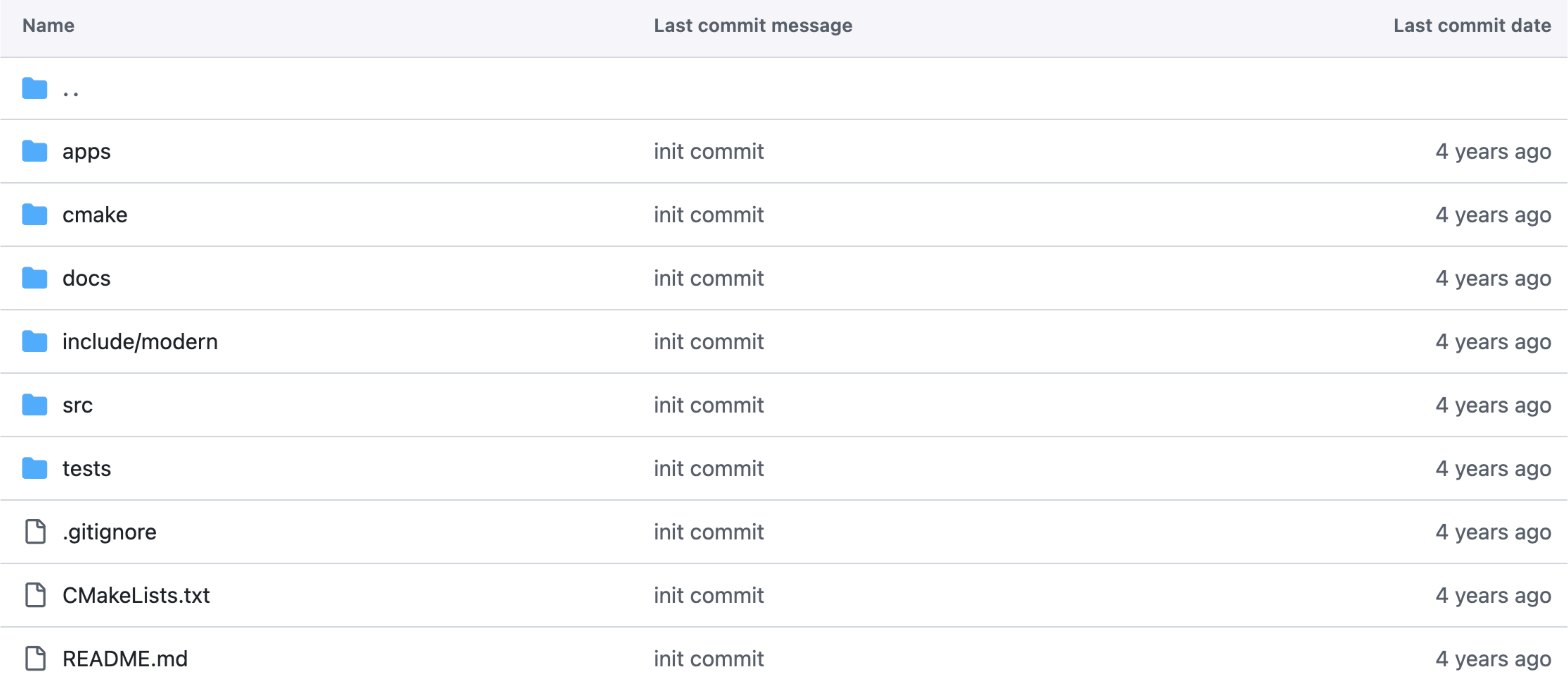
- You want to avoid hard-coding paths.
- You need to build the software on more than one machine.
- You must support multiple operating systems (even different Unix variants).
- You want to support multiple compilers.
- You prefer to describe your program's structure logically, not as a pile of flags and commands.
- You want to use third-party libraries.
- You'd like tools such as Clang-Tidy to assist your coding.
- You want to use a debugger effectively.
- You want to build and maintain a hug project.

CMake

Why Do I Need a Good Build System?



Name	Last commit message	Last commit date
..		
CMakeLists.txt	init commit	4 years ago
simple_example.cpp	init commit	4 years ago
simple_lib.cpp	init commit	4 years ago
simple_lib.hpp	init commit	4 years ago

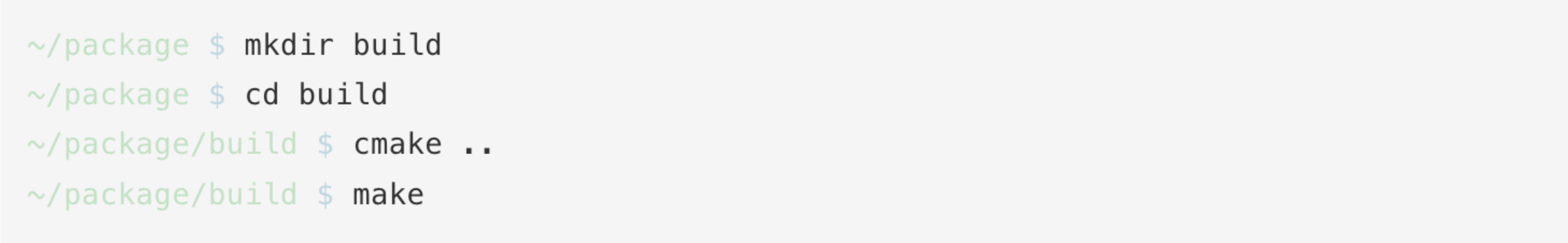


Name	Last commit message	Last commit date
..		
apps	init commit	4 years ago
cmake	init commit	4 years ago
docs	init commit	4 years ago
include/modern	init commit	4 years ago
src	init commit	4 years ago
tests	init commit	4 years ago
.gitignore	init commit	4 years ago
CMakeLists.txt	init commit	4 years ago
README.md	init commit	4 years ago

CMake

Building a project

- Unless otherwise noted, you should always make a build directory and build from there.



```
~/package $ mkdir build
~/package $ cd build
~/package/build $ cmake ..
~/package/build $ make
```

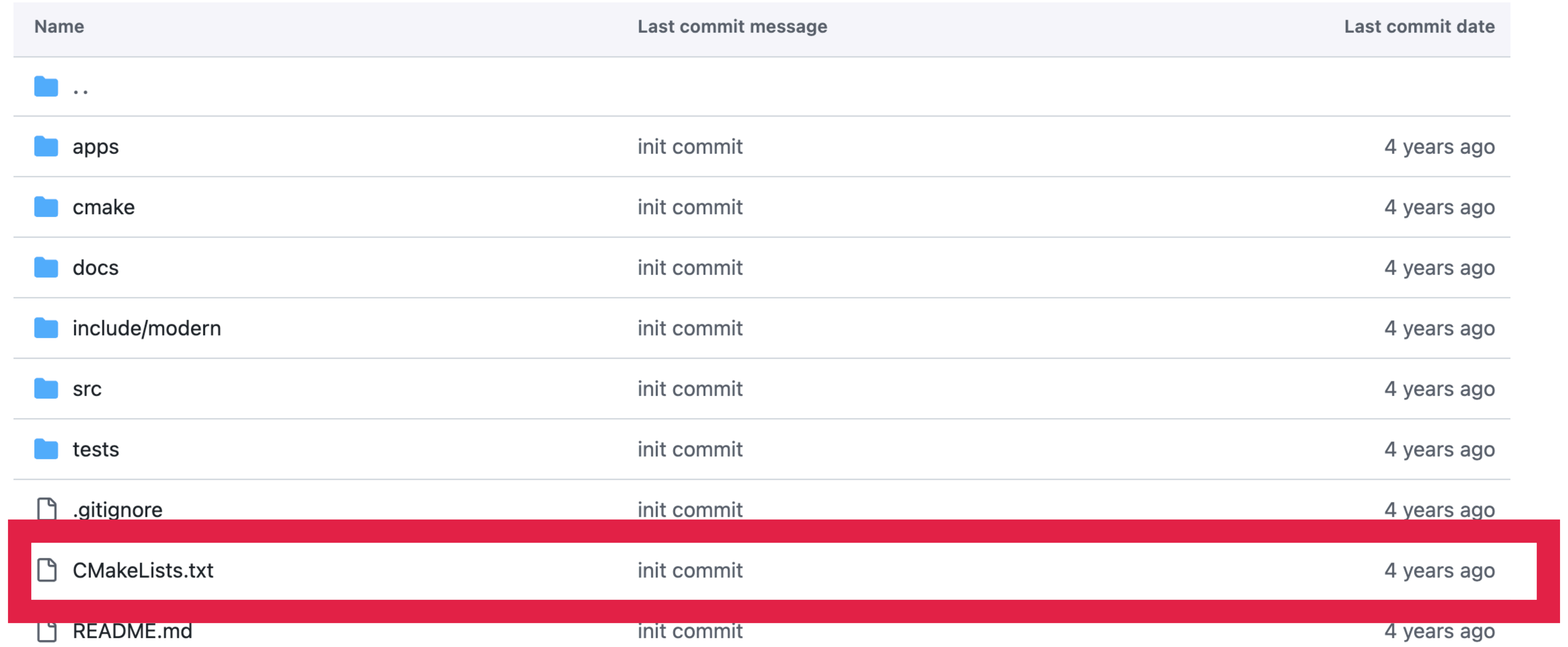
CMake

Standard options

- `-DCMAKE_BUILD_TYPE=`
  - Pick from Release, RelWithDebInfo, Debug, or sometimes more.
- `-DCMAKE_INSTALL_PREFIX=`
  - The location to install to. System install on UNIX would often be `/usr/local` (the default), user directories are often `~/.local`, or you can pick a folder.
- `-DBUILD_SHARED_LIBS=`
  - You can set this `ON` or `OFF` to control the default for shared libraries (the author can pick one vs. the other explicitly instead of using the default, though)
- `-DBUILD_TESTING=`
  - This is a common name for enabling tests, not all packages use it, though, sometimes with good reason.

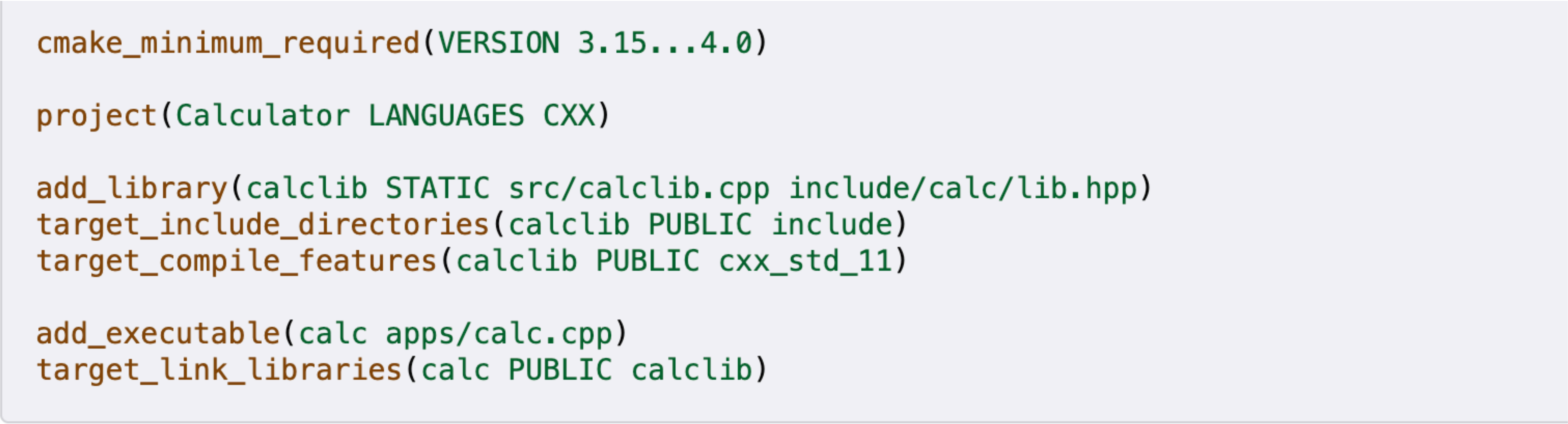
CMake

CMakeLists.txt



Name	Last commit message	Last commit date
..		
apps	init commit	4 years ago
cmake	init commit	4 years ago
docs	init commit	4 years ago
include/modern	init commit	4 years ago
src	init commit	4 years ago
tests	init commit	4 years ago
.gitignore	init commit	4 years ago
CMakeLists.txt	init commit	4 years ago
README.md	init commit	4 years ago

- See if you can follow the following file.
- It makes a simple C++11 library and a program using it.

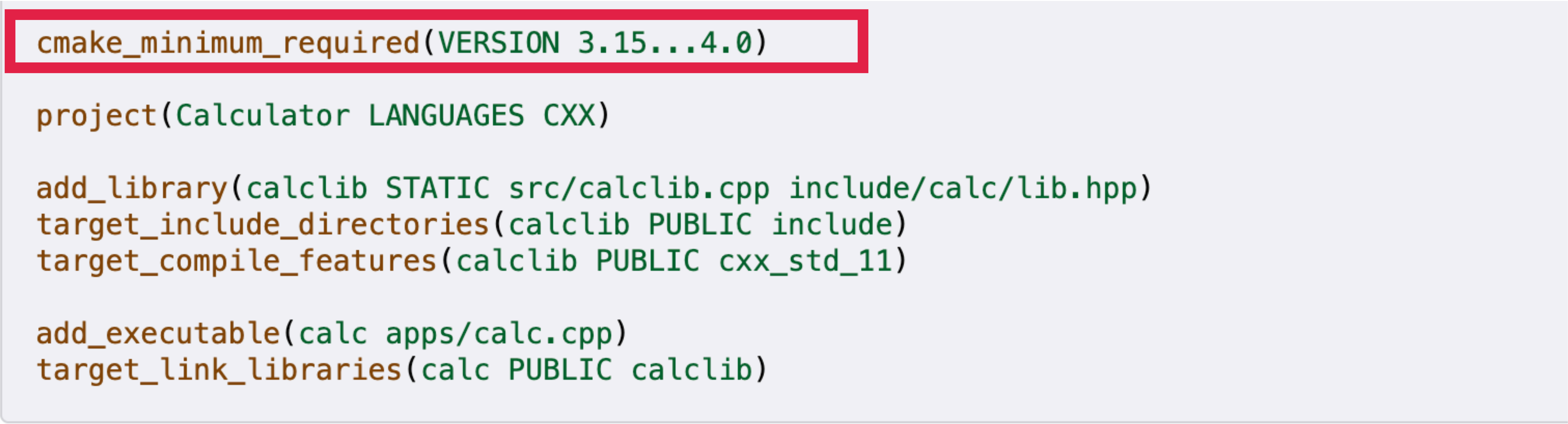


```
cmake_minimum_required(VERSION 3.15...4.0)

project(Calculator LANGUAGES CXX)

add_library(calclib STATIC src/calclib.cpp include/calc/lib.hpp)
target_include_directories(calclib PUBLIC include)
target_compile_features(calclib PUBLIC cxx_std_11)

add_executable(calc apps/calc.cpp)
target_link_libraries(calc PUBLIC calclib)
```

CMake

Minimum Version

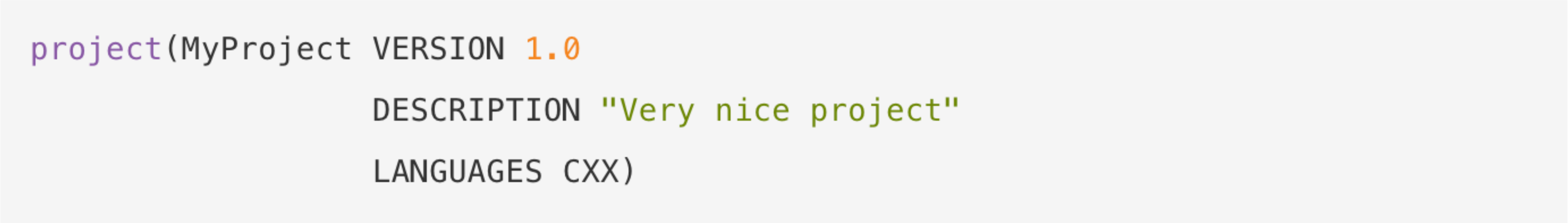
- Here's the first line of every CMakeLists.txt
- which is the required name of the file CMake looks for



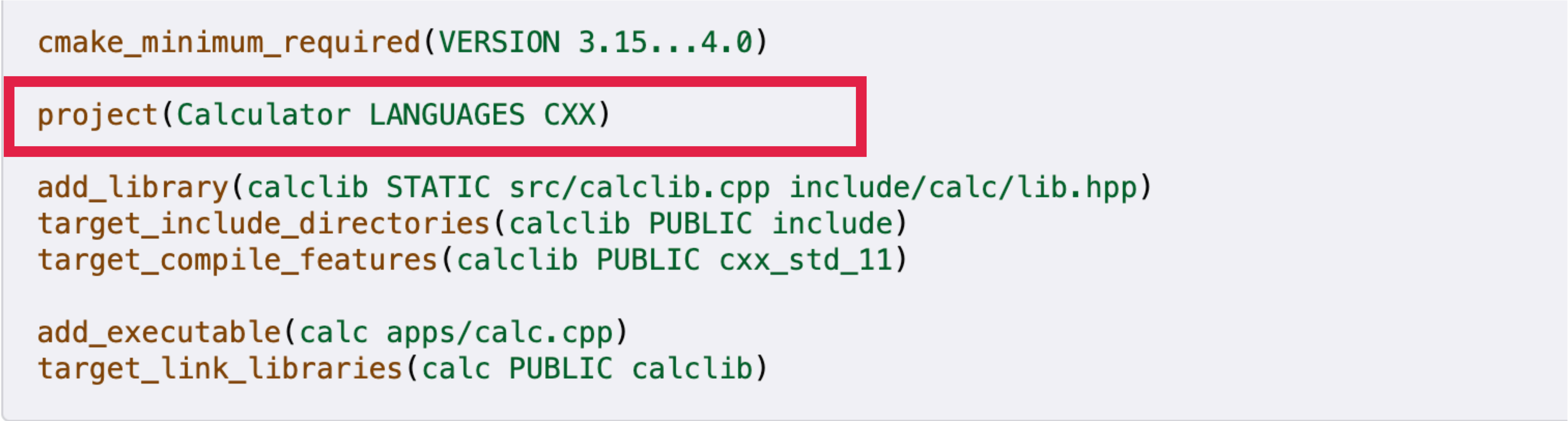
```
cmake_minimum_required(VERSION 3.15...4.0)

project(Calculator LANGUAGES CXX)

add_library(calclib STATIC src/calclib.cpp include/calc/lib.hpp)
target_include_directories(calclib PUBLIC include)
target_compile_features(calclib PUBLIC cxx_std_11)

add_executable(calc apps/calc.cpp)
target_link_libraries(calc PUBLIC calclib)
```

CMake

Setting a project



```
project(MyProject VERSION 1.0
                  DESCRIPTION "Very nice project"
                  LANGUAGES CXX)
```



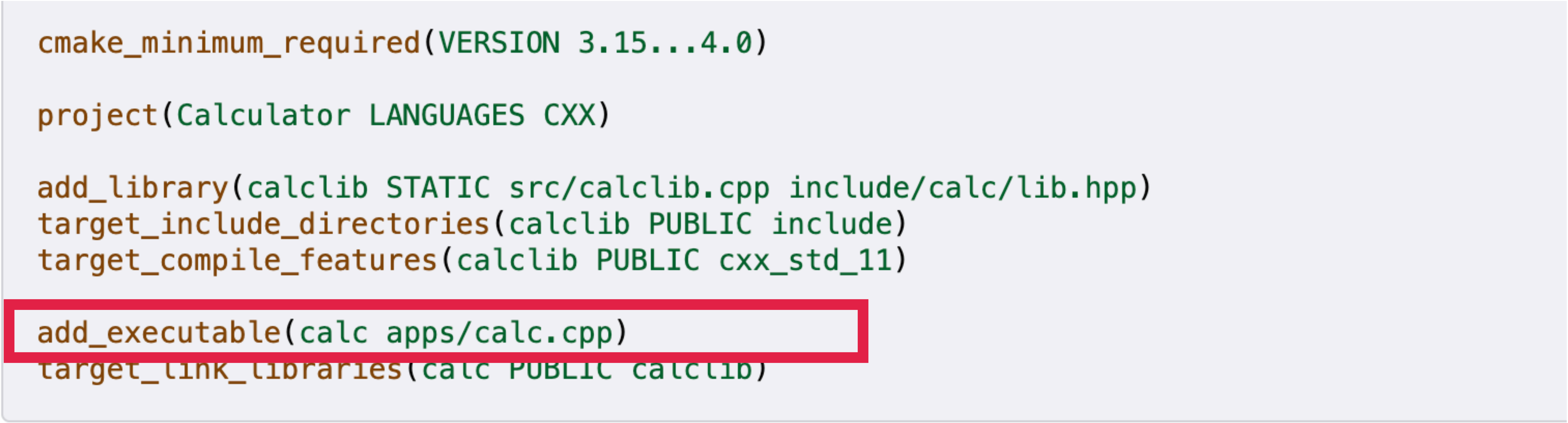
```
cmake_minimum_required(VERSION 3.15...4.0)

project(Calculator LANGUAGES CXX)

add_library(calclib STATIC src/calclib.cpp include/calc/lib.hpp)
target_include_directories(calclib PUBLIC include)
target_compile_features(calclib PUBLIC cxx_std_11)

add_executable(calc apps/calc.cpp)
target_link_libraries(calc PUBLIC calclib)
```

CMake

Making an executable

```
add_executable(one two.cpp three.h)
```

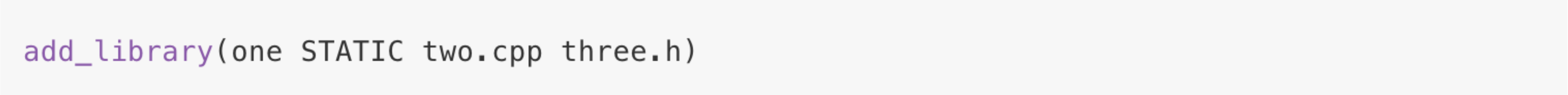
- Creates an executable target named one.
- Compiles two.cpp.
- Ignores three.h for compilation.



```
cmake_minimum_required(VERSION 3.15...4.0)

project(Calculator LANGUAGES CXX)

add_library(calclib STATIC src/calclib.cpp include/calc/lib.hpp)
target_include_directories(calclib PUBLIC include)
target_compile_features(calclib PUBLIC cxx_std_11)

add_executable(calc apps/calc.cpp)
target_link_libraries(calc PUBLIC calclib)
```

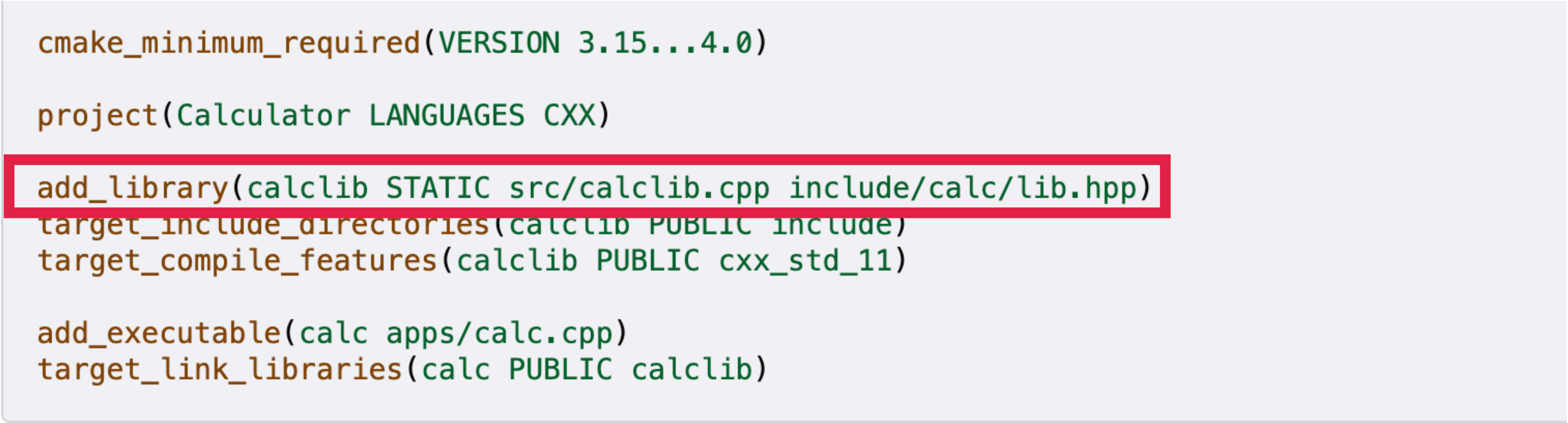
CMake

Making a library



```
add_library(one STATIC two.cpp three.h)
```

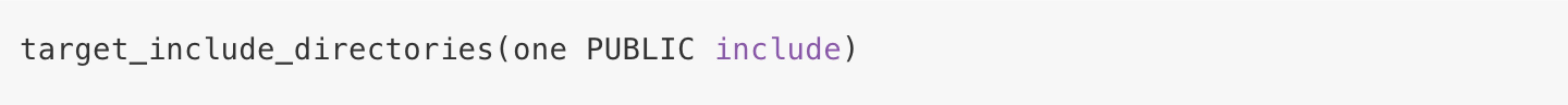
- Purpose: Create a library target named one.
- Types: STATIC (static), SHARED (dynamic).



```
cmake_minimum_required(VERSION 3.15...4.0)

project(Calculator LANGUAGES CXX)

add_library(calclib STATIC src/calclib.cpp include/calc/lib.hpp)
target_include_directories(calclib PUBLIC include)
target_compile_features(calclib PUBLIC cxx_std_11)

add_executable(calc apps/calc.cpp)
target_link_libraries(calc PUBLIC calclib)
```

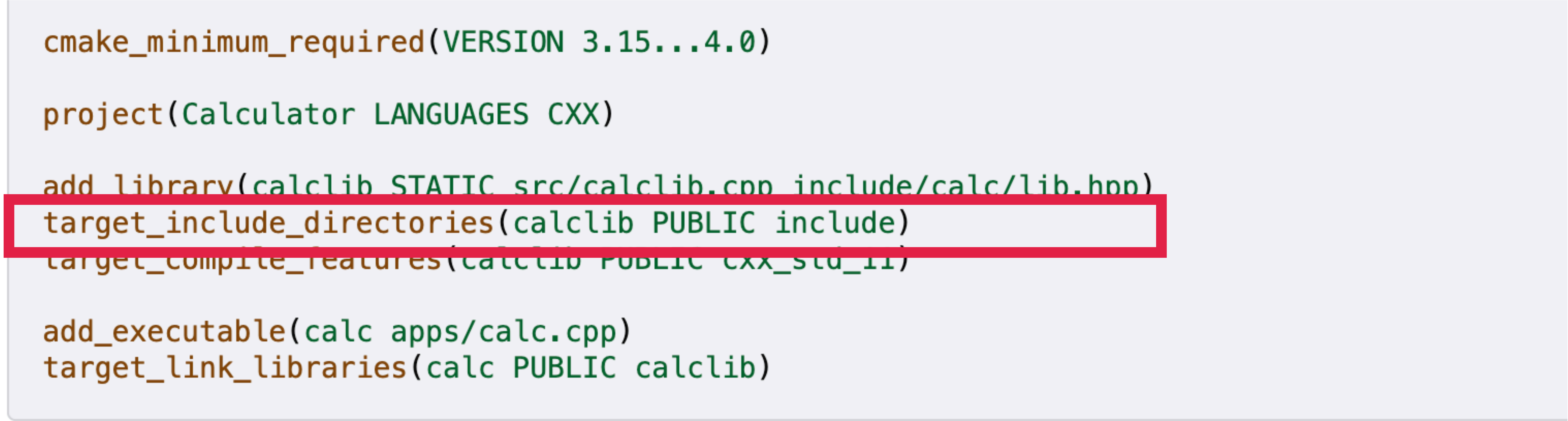
CMake

Making a Library



```
target_include_directories(one PUBLIC include)
```

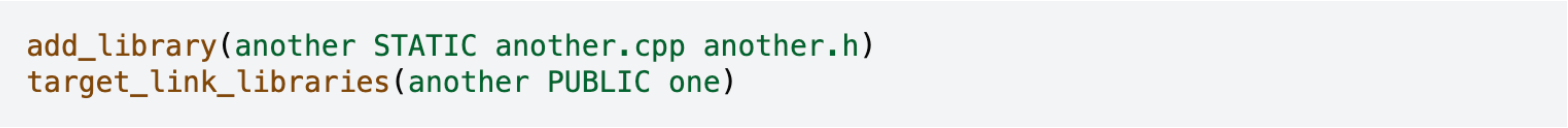
- adds an include directory to a target.



```
cmake_minimum_required(VERSION 3.15...4.0)

project(Calculator LANGUAGES CXX)

add_library(calclib STATIC src/calclib.cpp include/calc/lib.hpp)
target_include_directories(calclib PUBLIC include)
target_compile_features(calclib PUBLIC cxx_std_11)

add_executable(calc apps/calc.cpp)
target_link_libraries(calc PUBLIC calclib)
```

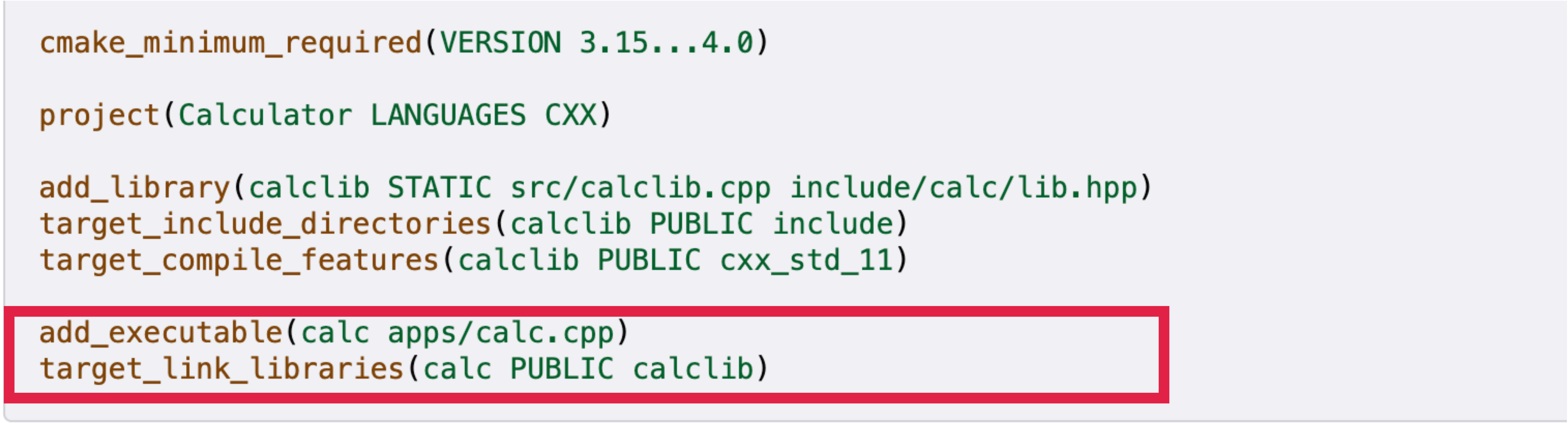
CMake

Making a Library



```
add_library(another STATIC another.cpp another.h)
target_link_libraries(another PUBLIC one)
```

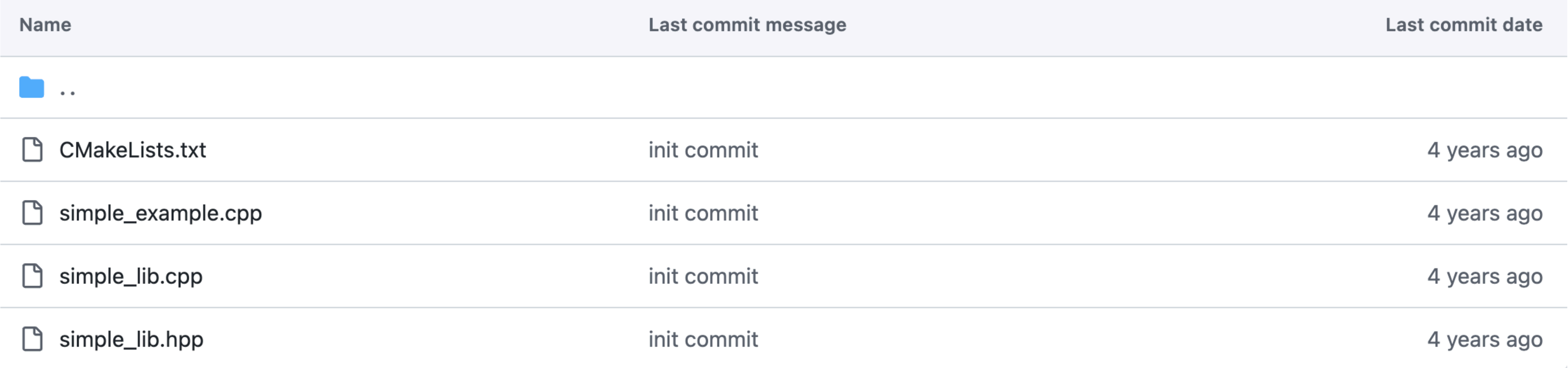
- adds an include directory to a target.



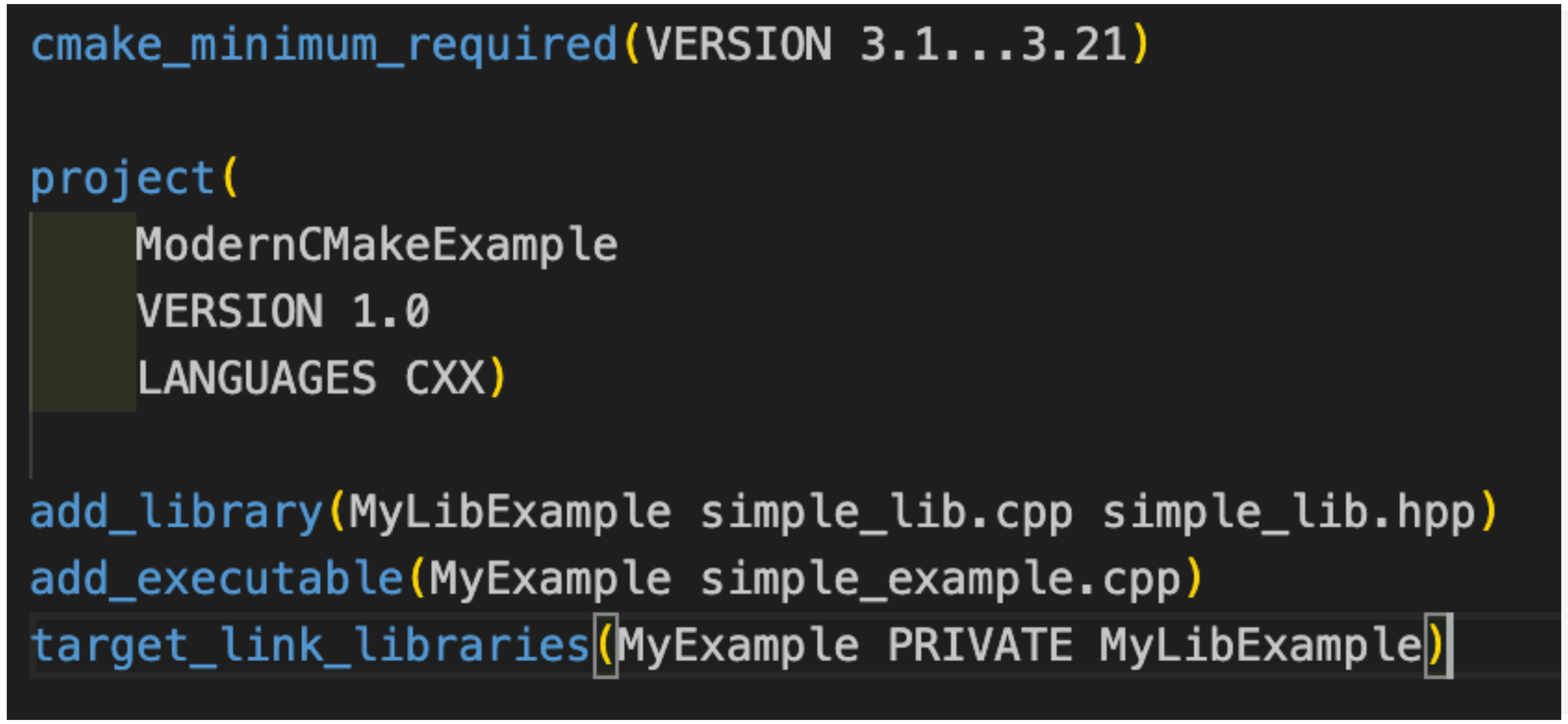
```
cmake_minimum_required(VERSION 3.15...4.0)

project(Calculator LANGUAGES CXX)

add_library(calclib STATIC src/calclib.cpp include/calc/lib.hpp)
target_include_directories(calclib PUBLIC include)
target_compile_features(calclib PUBLIC cxx_std_11)

add_executable(calc apps/calc.cpp)
target_link_libraries(calc PUBLIC calclib)
```

CMake

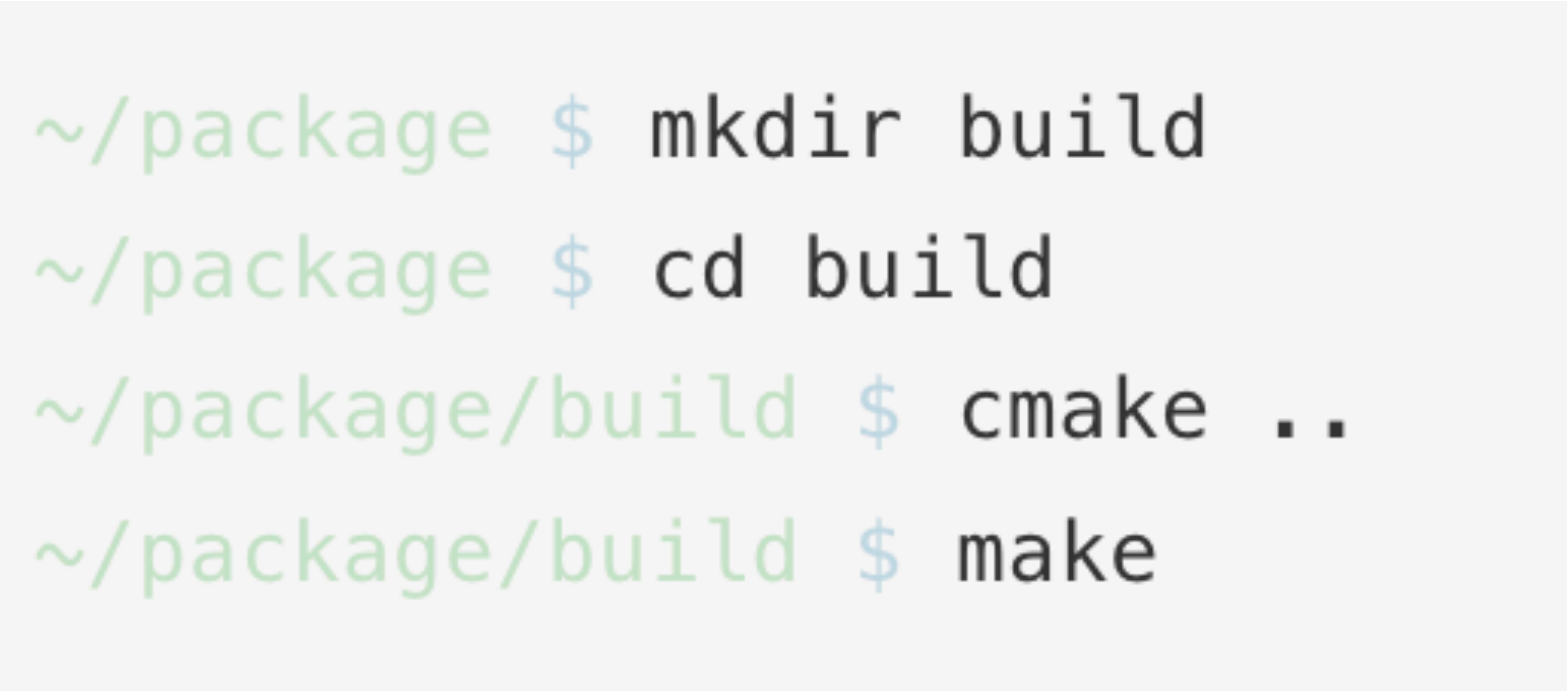
A simple example



Name	Last commit message	Last commit date
..		
CMakeLists.txt	init commit	4 years ago
simple_example.cpp	init commit	4 years ago
simple_lib.cpp	init commit	4 years ago
simple_lib.hpp	init commit	4 years ago



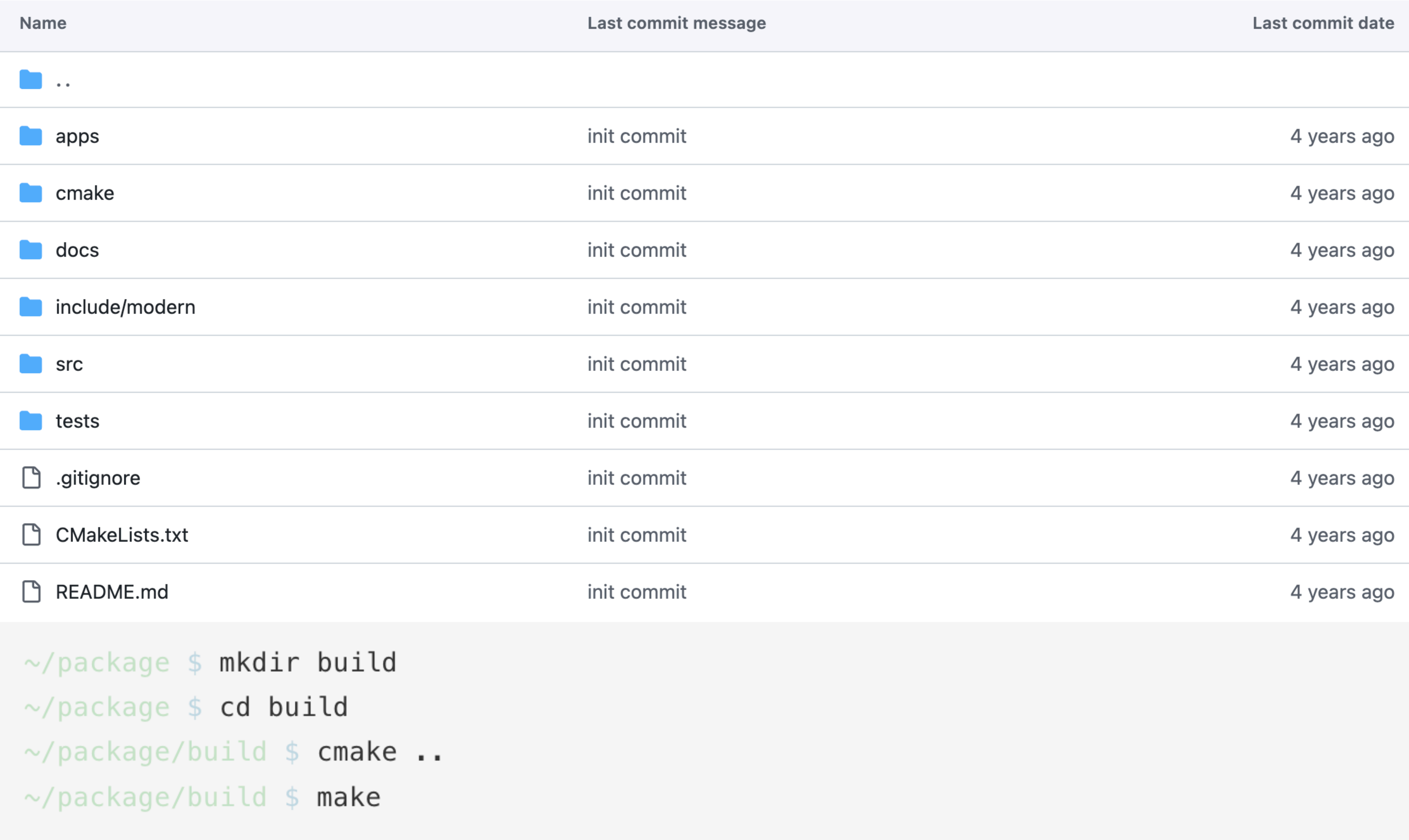
```
cmake_minimum_required(VERSION 3.1...3.21)

project(
    ModernCMakeExample
    VERSION 1.0
    LANGUAGES CXX)

add_library(MyLibExample simple_lib.cpp simple_lib.hpp)
add_executable(MyExample simple_example.cpp)
target_link_libraries(MyExample PRIVATE MyLibExample)
```



```
~/package $ mkdir build
~/package $ cd build
~/package/build $ cmake ..
~/package/build $ make
```

CMake

A simple example



Name	Last commit message	Last commit date
📁 ..		
📁 apps	init commit	4 years ago
📁 cmake	init commit	4 years ago
📁 docs	init commit	4 years ago
📁 include/modern	init commit	4 years ago
📁 src	init commit	4 years ago
📁 tests	init commit	4 years ago
📄 .gitignore	init commit	4 years ago
📄 CMakeLists.txt	init commit	4 years ago
📄 README.md	init commit	4 years ago

```
~/package $ mkdir build
~/package $ cd build
~/package/build $ cmake ..
~/package/build $ make
```