## Given the below code, what happens when running this code:

#include <iostream>

#include <functional>

void apply\_twice(std::function<int(int)> f, int x = 2) {

std::cout << f(f(x)) << std::endl;

}

int compute(int& x) {

return x \*= 2;

}

int main() {

int a = 3;

apply\_twice(compute, a);

return 0;

}

1. Prints 12, because apply\_twice correctly applies compute twice, and int& is implicitly converted to int.
2. The program compiles successfully but exhibits undefined behavior, because apply\_twice passes int instead of int&, leading to garbage values in compute(int&).
3. Compilation fails, because std::function<int(int)> expects a function taking an int by value, but compute(int&) requires an lvalue reference, causing a type mismatch.
4. Compilation fails, because function names cannot be passed to std::function, requiring std::function<int(int)> f = compute;.

## Given the code below, if we wants apply\_twice to modify the argument, instead of just printing its value, which of the following modifications will NOT contribute to this behavior, without changing the function body?

#include <iostream>

#include <functional>

void apply\_twice(std::function<int(int)> f, int x = 2) {

std::cout << f(f(x)) << std::endl;

}

int compute(int& x) {

return x \*= 2;

}

int main() {

int a = 3;

apply\_twice(compute, a);

return 0;

}

1. Change std::function<int(int)> to std::function<int&(int&)> inside apply\_twice.
2. Change int x = 2 to int& x = 2 inside apply\_twice.
3. Change int compute(int& x) to int& compute(int& x).
4. None of the above.

## Given the below code below, what happenes when running this code?

#include <iostream>

int main() {

    unsigned int age{13};

    auto say\_my\_age = [age]() {

        std::cout << "You are " << age << " years old!";

        ++age;

    };

    say\_my\_age();

    std::cout << "Age " << age << std::endl;

    return 0;

}

1. You are 13 years old! Age 13.
2. You are 13 years old! Age 14.
3. Compilation fails, because we cannot assign a lambda to a variable.
4. Compilation fails, because we cannot change the value of “age”.