INTRODUCTION TO FUNCTIONS

Introduction to Functions

- A function is a tool that makes code easier to read and more efficient
- To some extent, you can think of Computer Science functions as akin to mathematical functions
- A function in C++ is like a machine that only knows how to do one specific task, but it always completes it successfully.

C++ Function Syntax

Prototype (Above main):

```
returnType functionName(functionParameters) ;
```

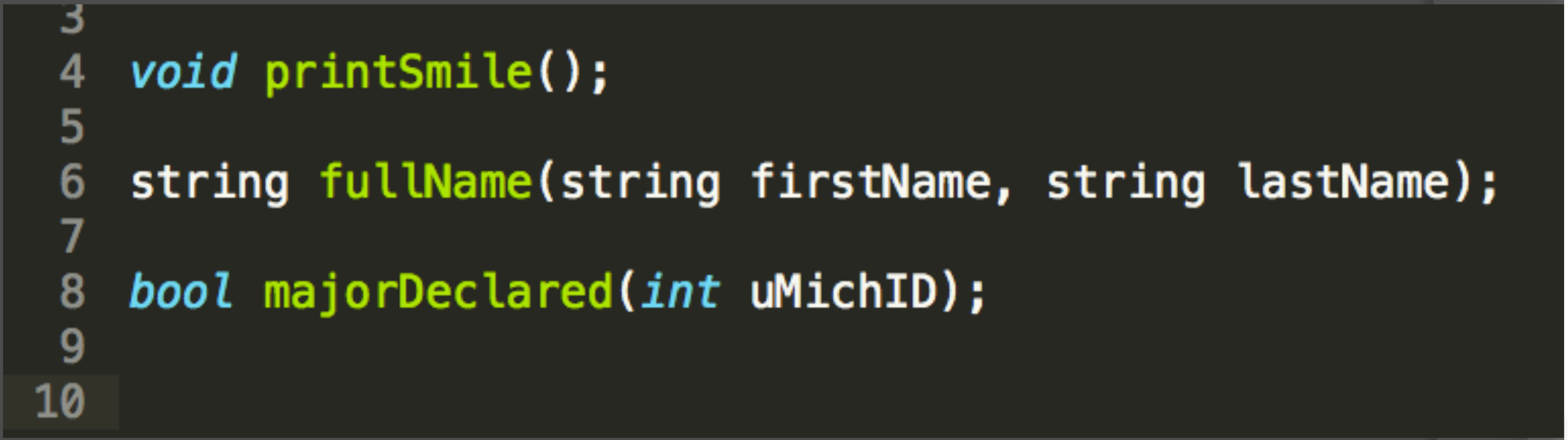
Definition (Below main):

```
returnType functionName(functionParameters) {
      //body of function
      //return statement if needed
}
```

What is the prototype?

- In C++, the prototype of a function is a line that goes above the main function. It basically provides a blueprint for the function, detailing the function name, inputs (parameters) and output (return value)
- It is necessary because it tells the compiler that the function exists and saves room for it, before it sees the function called in main.
- The prototype is identical to the first line of the function definition, except it ends in a semicolon (which is traditionally written below main in the file)
- Prototypes are not needed if the function definition is placed before main. This is, however, not the best choice style wise because it makes it more difficult to find the main function when debugging and sharing code.

Function Prototype Examples



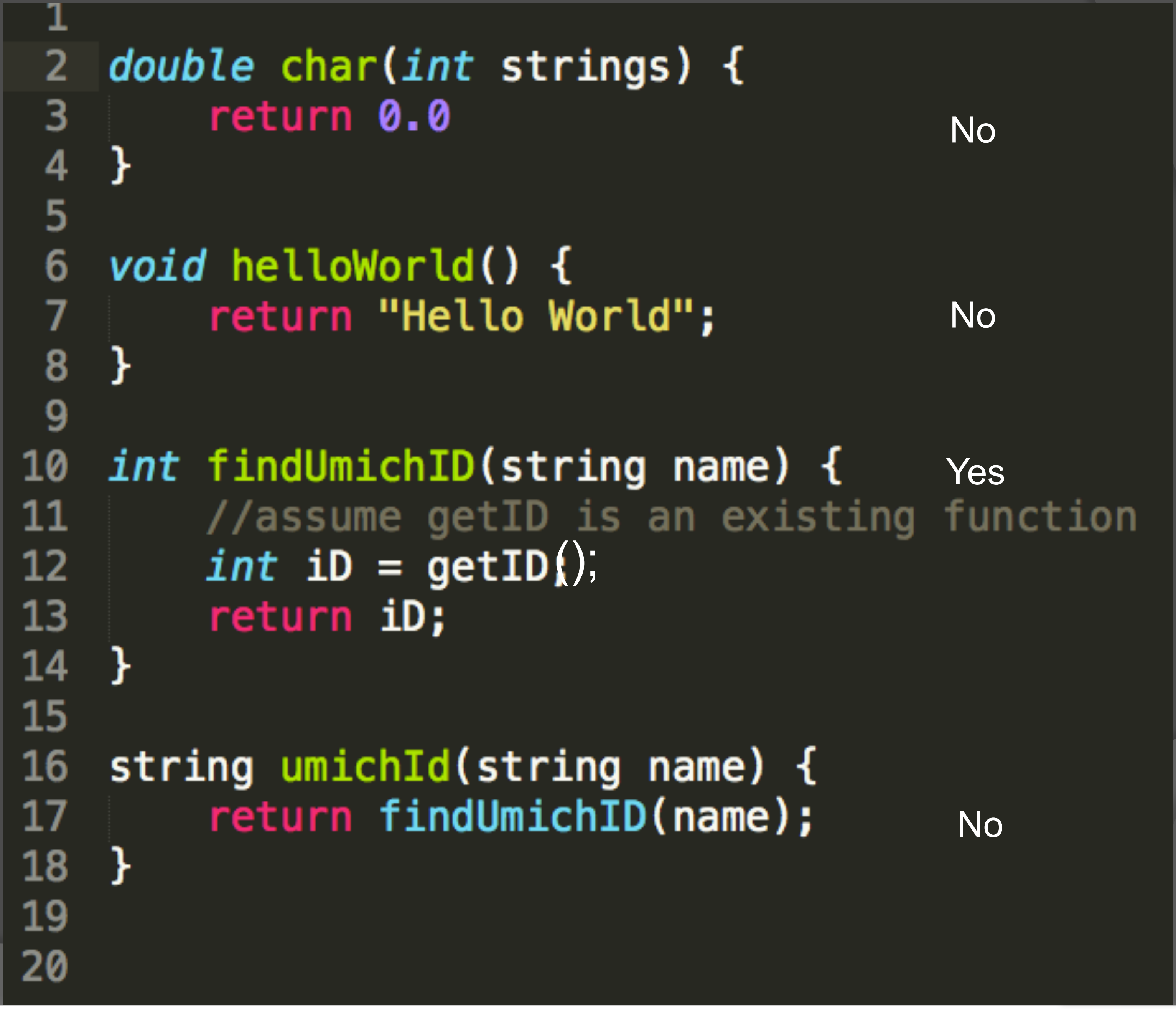
```
void printSmile();

string fullName(string firstName, string lastName);

bool majorDeclared(int uMichID);

```

Function Definition Practice Problem

Write a function that takes a amount in US dollars and converts it to Euros!

(solution found in the main functions wiki page)

Are these working functions? Why, why not?



```

double char(int strings) {
    return 0.0
}

void helloWorld() {
    return "Hello World";
}

int findUmichID(string name) {
    //assume getID is an existing function
    int iD = getID();
    return iD;
}

string umichId(string name) {
    return findUmichID(name);
}


```

No

No

Yes

No

Scope

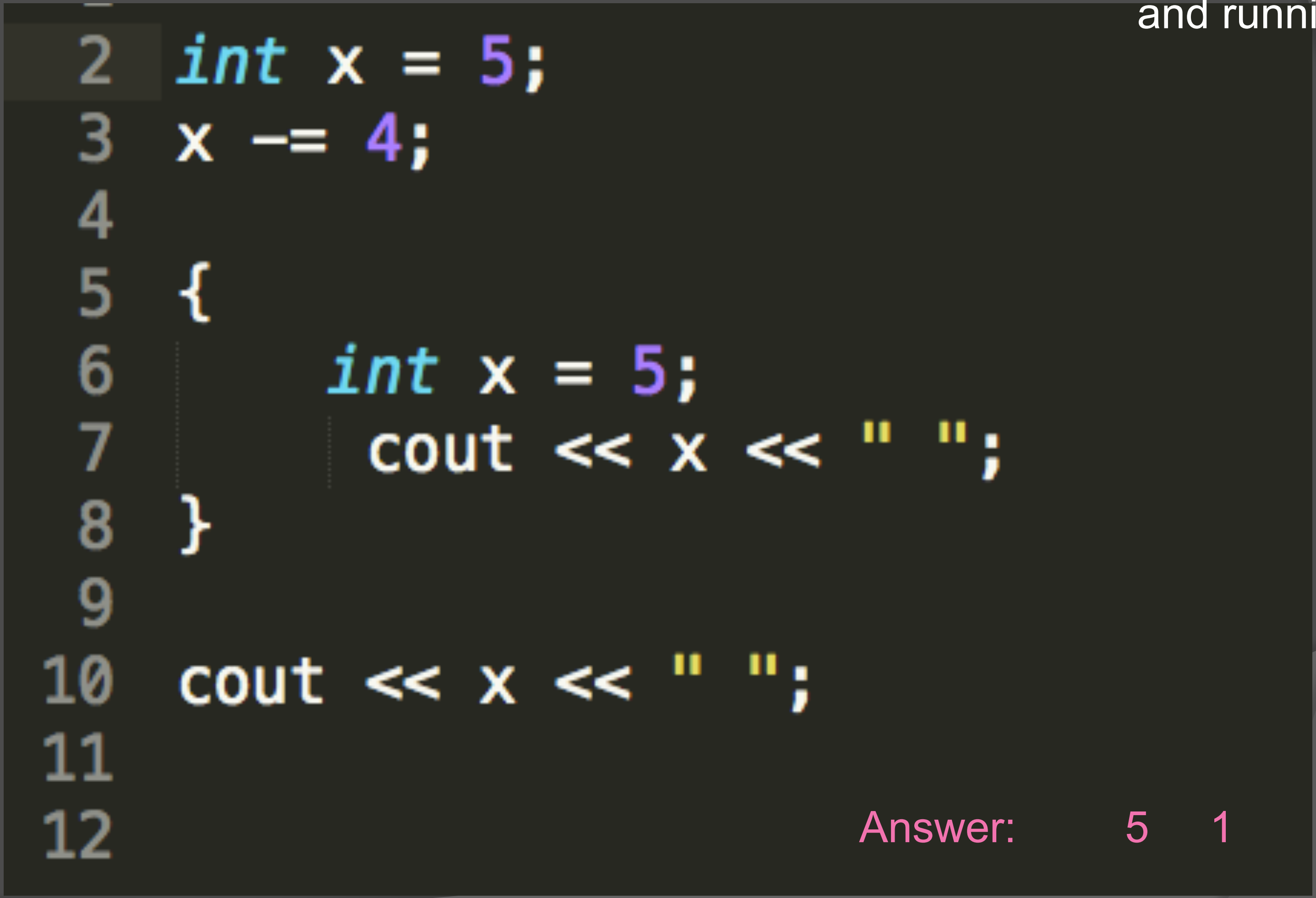
- The scope of a variable is the region of the code that that variable is accessible
- Scope is determined by the nearest set of { }
  - When you define a variable, it can only be used within the closest curly braces to where it was defined
- This goes for separate functions, if statements, and loops
- Variables defined in a function are called local variables

Scope Example

```
for (int i = 0; i < 10; ++i) {

    cout << “Hi!\n”;
}
 //Does not compile because i is now out of scope
cout << i;
```

Another Scope Example:

What does this code print? Try typing it out and running it.



```
int x = 5;
x -= 4;

{
	int x = 5;
	cout << x << " ";
}

cout << x << " ";


```

Answer: 5 1