Rhea Mae Edwards

Syntax errors (What does the compiler say!!!)

- Remove a semicolon from line 13.
  equal.cpp: In function ‘bool is_equal(char*, char*)’:
  equal.cpp:14: error: expected ‘;’ before ‘}’ token
  The complier states that there is a missing semicolon on line 14, before the curly brace, which seems reasonable since the missing semicolon is just at the end of line 13.
- Remove the curly brace from line 4.
  equal.cpp:5: error: expected initializer before ‘int’
  equal.cpp:6: error: expected unqualified-id before ‘for’
  equal.cpp:6: error: ‘i’ was not declared in this scope
  equal.cpp:6: error: expected constructor, destructor, or type conversion before ‘!=’ token
  equal.cpp:6: error: expected constructor, destructor, or type conversion before ‘++’ token
  equal.cpp:10: error: expected unqualified-id before ‘if’
  equal.cpp:13: error: expected unqualified-id before ‘return’
  equal.cpp:14: error: expected declaration before ‘}’ token
  The complier states that most of the information that is supposed to be within that function, that would have been enclosed by that removed curly brace on line 4, are not globally stated, not making much sense or not correctly qualified.
- Remove the brackets from the argv arguments in the call to is_equal() on line 18.
  equal.cpp: In function ‘int main(int, char**)’:
  equal.cpp:18: error: ‘argv2’ was not declared in this scope
  The complier stated that what was supposed to refer to the third argument in the command line, became a variable that was not declared within the function.
- Remove the asterisk/splat from the argv in main on line 16.
  equal.cpp:16: warning: second argument of ‘int main(int, char*)’ should be ‘char **’
  equal.cpp: In function ‘int main(int, char*)’:
  equal.cpp:18: error: invalid conversion from ‘char’ to ‘char*’
  equal.cpp:18: error: initializing argument 1 of ‘bool is_equal(char*, char*)’
  equal.cpp:18: error: invalid conversion from ‘char’ to ‘char*’
  equal.cpp:18: error: initializing argument 2 of ‘bool is_equal(char*, char*)’
  The compliers warns the author that the other parameter should have another asterisk/splat, and then on line 18, there is a change in the type of variables being used which is not correct.
- Remove the return type, bool, from the is_equal() function on line 4.
  equal.cpp:4: error: ISO C++ forbids declaration of ‘is_equal’ with no type
  The complier states that the author cannot have a function without a type of line 4, which is exactly the error in the code.
- Remove one of the parameters in the is_equal() function on line 4.
  equal.cpp: In function ‘bool is_equal(char*)’:
  equal.cpp:6: error: ‘str2’ was not declared in this scope
  equal.cpp:10: error: ‘str2’ was not declared in this scope
  equal.cpp: In function ‘int main(int, char**)’:
  equal.cpp:4: error: too many arguments to function ‘bool is_equal(char*)’
  equal.cpp:18: error: at this point in file

The complier states that the parameter I removed is not declared in that function, and how there are too many arguments stated when that function is called in the main function; both statements are correct due to the removal of one of the parameters in that one function.

- Comment out line 5, and declare i in the for loop on line 6.
  equal.cpp: In function 'bool is_equal(char*, char*)':
  equal.cpp:10: error: name lookup of 'i' changed for ISO 'for' scoping
  equal.cpp:10: note: (if you use '-fpermissive' G++ will accept your code)
  The complier states the name lookup in the following for loop has been changed, and there is an included note on how the code can be further accepted by the complier.

Logic errors (What is the output!!!)

- Put two asterisks/splats on the parameters of line 4, and put ampersand, &, in front of the arguments on line 18.
  The output is always false.

Let's use the gdb debugger… If we were to have this type of error and didn't know what is going on, where would be the first place you would look? You should think to yourself, what are the values of str1 and str2 when the function is_equal() is called, i.e. not before the call, but after the call!

Use the gdb debugger to set the appropriate breakpoint in the program (after the call to the is_equal() function but at the declaration of the local variables) and display the values of str1 and str2. Are these what you expect?

Make your changes back and look at what you are passing when the program is correct, rather than when you added the extra splat and ampersand.

- Comment out line 10, and run with ./string_equal hell hello.
  How would you use gdb to discover this error?
- Comment out line 16, and run with ./string_equal hello
  How would you use gdb to discover this error?