**Homework 4, Fall 2013 C programming assignment #1**

**Do before Monday, September 16 , Interacting with C**

**at start of class**

Go to codepad.org, choose C, set your "pastes" to be private. Enter code in the edit window and submit to receive results.

Note: the reference, "the C handout", is to the "A Quick Introduction to the C Programming Language" handout.

1. Enter the following program from section 5 of the C handout. (Like Java, indenting is for the human reader.)

/\* C program #1 \*/

/\* Put your name here! \*/

#include <stdio.h>

main ( )

{

int s;

s = square(2);

printf("The square of 2 is %d \n", s);

return (0);

}

int square(int x)

{

return(x \* x);

}

**Submit**. Run with different values for parameter and change it in the output line as well, convincing yourself that the program works properly. Type the link (see near top of left on codepad) to your "Paste" below, so that I can see it.

2. Modify your code and change the program number to 2.

1. Change the return type of the square function to "double". Also, change the declaration of variable s to be of type double. **Submit**. What result do you get?

Errors

Copy the link to the paste here: <http://codepad.org/BqtnQjG3>

1. In the error message above, notice the reference to the "implicit declaration of 'square'". As mentioned in section 5 of the C handout, when a call to a previously undeclared function is made, C assumes that that function will return an int. This allows compilation to continue "in a single pass." Modify the program to declare the type of the method (to appear later) by adding following declaration of square (called the prototype) to your C code, right after the #include line: double square( ); and **Submit.** What is the result now (it should run, but are the results what you expect)?

Says the square of 2 is 0.

Fix the "printf" line to display a real number by changing "%d" to "%f". **Submit**.

Better? (Why or why not?)

Yes, it displays the number followed by “.0000”.

Copy the link to the paste here: <http://codepad.org/Wg7cg0cx>

**Note** that the prototype, added above in #3b, tells the compiler that function square will return a double. This tells the compiler all it must know if order to be able to translate the call to square in the main.

1. C is weakly typed, but maybe we can get it to do more for us, if we put more information in the program. Modify your program to include the following prototype instead of the one already there: double square(y);

**Submit**. Results?

It didn’t change for me.

Before moving on, change the function header (not the prototype, which should remain unchanged) and the call to include two arguments (square (2, 3)) and parameters (double square(int x, int y) ), respectively, and **Submit**. Any problems detected related to numbers of parameters? \_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_

It didn’t accept the 2nd parameter

What does this tell you about C's type checking as far as verifying parameter counts goes?

Not that great.

Let's see if it checks calls to functions? Change the call to include only a single argument (as in square(2)), while leaving the header with two parameters. What are the results?

Still works

What do you think might be happening to generate the result above?

It’s going by the declared prototype instead of the parameters in the actual method

Let's check out the value stored in y in the body of the function square. Add the line printf("y = %d \n", y); right before the "return" statement. **Submit**.

What do you think might be happening to generate the result above?

Nothing good

Copy the link to the paste here: <http://codepad.org/cT3LDbc9>

Once again, modify your program so that the function prototype is:

double square(int y);

and the function accepts only a single parameter, x. Now the prototype, the call and

the function definition agree in numbers of parameters and the prototype and

function agree in type of parameter/arugment. (Remove the printf statement from

the function. ) NOW, change the parameter in the

call to a double, such as 3.5, and change it in the output line as well – so your output

continues to make sense. Do you expect to get an error here? \_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_

**Submit**.

Result?

Didn’t account for the decimal

What do you think is happening?

The method still is an int so it’s cutting off at the decimal

Copy the link to the paste here: <http://codepad.org/npqilovf>

The Java compiler would detect an error in sending a double literal value into an int parameter. Obviously this is neither a compile-time or run-time error in C, it is important to know this! (As a C programmers you must be more disciplined . If you send that real number in by mistake, C doesn't know it, and, unlike Java, it **will** "narrow" the value for you; that is, it implicitly converts your real number to an int – surprise! (An implicit conversion is called a coercion.)

Would you prefer a compile-time error or a coercion? **Why?**

Compile time error, because I would be more likely to notice it

3. Create a new paste – we're starting on a new problem. Enter the following program.

/\* Your name goes here \*/

#include <stdio.h>

#define MAX 5

int main()

{

double data[MAX] = { 34.0, 27.0, 45.0, 82.5, 22.1 };

int i;

for (i = 0; i < MAX ; i++)

printf("data[%d]=%5.2f\n", i, data[i]);

return 0;

}

1. **Submit**. Results?

works

1. Try putting the declaration for i as part of the initialization of the loop control variable like we often do in Java (and C++). ( That is, comment out the declaration of variable i and change the for to “for(int i; …” What happens?

breaks

1. Put the code back like it was before (in part a). Trying changing the formatting values in the printf statement and tell me what happens with each change:
   1. Change "%5.2f" to %8.3f"

Nothing changed

* 1. Change %8.3f" to %f"

Still the same

* 1. Change %f" to %15f"

Still working the same

* 1. Change %f" to %2.3f"

Output still the same

* 1. Change %d" to %5d"

Spaced out the [] in regards to the subset of the data

* 1. Change %d" to %f"

All subset data values 0.

* 1. Change the %f back to %d in part f and change %2.3f" to "%d"

Everything became 0 except for a large neg in data[4]

* 1. Change the formats back to the original and write the link to your paste below here.

<http://codepad.org/v3eAFJbH>

4. Modify the program in part 3 and change your array declaration to the following:

char data[MAX] = { 'a', 't', '8', '\*', 'B' }; and your

printf statement to: printf("data[%d]=%c\n", i, data[i]);

**Submit**. Results?

errors

Now, change the "%c" format in your print statement to "%d". Can a char be interpreted as an integer? Where do these numbers mean?

No, at least I got an error when trying

Let's see if this would this work the other way. That is, I wonder if we could store an array of integer and them print them out as characters… Give it a try… modify the code so that data contains the integers { 32, 69, 115, 550, 551 } Can you explain what is happening here?

Either Codepad hates me or I’m doing something wrong: <http://codepad.org/GAGryvd0>

Now let's check how well C does in checking array bounds, at compile time or run-time. Using the same code as in the immediate previous problem, change the loop termination condition to <= MAX, so it should run off the end of the array! **Submit**.

Any problem detected during execution?

The same as before

Let's see other access errors might be detected…

Can an array be indexed by something other than an integer?

Try changing data[i] in the printf statement to data[100.0]. **Submit**. Result?

Try changing data[100.0] to data['a']. **Submit**. Result? (Weird? Why or why not?)

“array subscript is not an integer”

Set the reference back to data[i]. **Submit**. Write the link to your final paste below:

<http://codepad.org/bagZt00Y>