CS 121 Week 7 Worksheet: Reference Variables, Scope,

Prototypes, and Default Parameters

**Topics:**

* Passing values by "value" and by "reference"
* Function Prototypes
* Local, Global, and Static Variables
* Default Parameters

**Some Notes/Definitions:**

* Arguments are sent into the function (i.e. what is used in the function call), whereas parameters are what the function is defined with (i.e. in the function header).
* Arguments in a function are either passed *by value* or *by reference*
  + - **By value**: Send a copy (not the variable itself) of the argument in the function
      * These are sometimes referred to as **in-parameters** (in-param) as they only go **into** the function.
    - **By reference**: Use the **same** variable in the function (as the one we're passing) in the function.
      * These are sometimes referred to as **out-parameters** (out-param) as they go back **out** of the function.
* **Function prototypes** are a way to let the compiler know what functions we **plan** to make.
* **Default parameters** are a way to "assume" function parameters
  + - ex) *void print\_line(char ch, int num = 50); //the second part is a default parameter*

which allows us to call *print\_line('c');* if we want *num = 50*

* + - We must implement default parameters starting from the right (e.g. the line

*void print\_line(char ch = 'c', int num);* is invalid because making the function call would not make sense)

* + - The default parameters can be overridden (e.g. *print\_line('-', 40);*)
* **Local variables** belong to **one** function **only** (doesn't "reach" any others)
* **Global variables** are defined outside all functions. They can be accessed by **all** functions.
* **Static variables** keep their values even when you leave their scope.

Use the information above and from the class PowerPoint notes (CH6.1), your notes, or online for extra help to do the following problems.

**Syntax Practice:**

1. Write function prototypes with the following specifications. Be sure to list (on the side) what your function prototype's return type, name/identifier, in-param, out-param, and guess on its purpose:
2. A function *init\_student* that returns a boolean, takes in an integer called *id*, double called *grade*, and a string called *name*, where each parameter is called by reference.
3. A function called *conv\_temp* that returns a double, takes in a double called *temp* and a boolean called *is\_cels* that has a default value of true, where each parameter is called by value.
4. A function called *output\_char* that returns nothing, takes in an integer named *amount* with default value of 20, a character named *ch* that is called by value, and an ostream object called *out\_source* that is called by reference and has a default value equal to the *cout* object.
5. Write a function (not *main*) that takes in an integer and prints itself squared. Have *main* call the function with an integer argument equal to 12.
6. Write a function that swaps two doubles. This should affect the variables where the function call is made (the variables being the two doubles passed into the function). Have *main* call this function with two doubles as arguments, one called *first\_num* which equals 10 and the other called *second\_num* which equals 44.
7. Fix all the errors in the below program. After doing so, what will the output of the program be?

#include <iostrea>

sing namespace std;

const double TAX\_RATE = .81;

double get cost(double);

int main()

{

double bp = 12.00; //$12.00 is the base price before tax

double actual\_cost = get\_cost(bp);

cout << "Before tax: $" << bp << end

cout << "Tax rate: $0" < TAX\_RATE << end;

cout < "After applying tax: $ << actual cost << end;

cout << "Same thing as before: $" << get\_cost(bp) << endl;

system(pause);

return 0;

}

double get\_cost(double)

return (base + (TAX\_RATE \* base));

**Concept Questions:**

1. Why are global variables considered "bad practice" if they are not constant?
2. When, do you think, passing by value is better than by reference? By reference rather by value?
3. What's the output of the following code? Why? Can we put *cout << num << endl;* inside of *int main()*? Why or why not?

#include <iostream>

using namespace std;

void print\_static()

{

static int num = 0;

num++;

cout << "Value of num: " << num << endl;

}

int main()

{

for(int i = 0; i < 5; ++i)

{

print\_static();

}

system("pause");

return 0;

}

1. In what situation could having a static variable be useful? Avoid using the previous example as your "situation".
2. What is the purpose of using default parameters? How can they be helpful? List at least two scenarios.