There is a plp_5_6.cpp file also included in this repository that contains code that tests all of this information. It may help to look at this file while reading through the document.

Discussion Questions

1. **What is the syntax for declaring a function in your language?**
   To declare a function in C++ you must first specify a type, such as int or string. This is for the type of value that will be returned from the function. If there will not be returning a value, you must use void. After this is the name of the function, which should be descriptive enough to describe its purpose. For example, a function that returns the product of two numbers could be called getProduct. Next is the parameters of the function, which refers to what is being passed to it. You will also need to define what type they are. These go inside of parentheses after the function name. If the function will not be returning anything, leave them empty. The next component is a set of matching curly braces, inside of which the code for that function will be written. To sum it up:
   ```
   type functionName(type parameter1, type parameter2, ...){ code; }
   ```

2. **Are there any rules about where the function has to be placed in your code file so that it can run (i.e., before main, after main, in the same file, in the same folder, etc)?**
   In C++ a function can be placed either before or after main(). However, it is important to recognize that in this language functions need to be declared before they are used. Thus, if you opt to place your functions after main() you will need to declare the function before main(). This is done in a similar way to how a function is declared, but instead of including the curly braces with the code you simply put a semi-colon after the parentheses. In short:
   ```
   type functionName(type parameter1, type parameter2, …);
   ```

3. **Does your language support recursive functions? If so, write one.**
   The C++ language does support recursive functions. This means that the function calls itself. However, it is important to note that steps must be taken to ensure that the function does not end up in a continuous loop. This can be done using if/else statements to check a condition before entering the block of code in which the function calls itself. Thus, if the condition is not met, the recursivity of the function will end.

4. **Can functions in your language accept multiple parameters? Can they be of different data types?**
   Functions in C++ can accept multiple parameters(see Discussion Question 1). The parameters can also be of multiple data types.

5. **Can functions in your language return multiple values at the same time?**
   Functions in C++ do have the capability to return multiple values at the same time. To utilize this, you return the multiple variables in the function by separating them with commas, like so:

return variable1, variable2;

Then, in main, you would assign the returning values to multiple variables also using commas. It would look like this:

variable1, variable2 = functionName(parameters);

6. **Declare a variable (say, x) in the main body of your program. Then declare a variable of the same name inside of a loop. Is there a conflict? Is the old variable overwritten or do you now have two variables of the same name?**
   There is no conflict when doing this, however you must remember that the variable inside of the loop will only be available for use within that loop. Thus, once you exit the loop and try to use the other variable with the same name, the program will be accessing the variable that was created outside of the loop.

7. **What if the other x is inside a function?**
   Like before, there is no conflict when there are two variables that have the same name when one is in a function. However, you must also remember that the value of the variable inside the function is local to that function, which means that if you would like to access it outside of the function you must return it and place it into another variable wherever it was called from.

8. **Can you have variables that are globally accessible? What are the rules for creating them?**
   You can have variables that are globally accessible. To do this in C++ you must define the variable at the very top of the program under all of your #include statements.

9. **Are variables passed by value or by reference? (Hint: write a function that alters its input, but doesn't return it. Pass it a variable, and see if the alteration is visible in main after you call the function)**
   Variables in C++ are passed by value, which means that when you pass a variable, and thus its value, to a function, and that function alters that value, the original variable's value is not also changed.

10. **If you run this code (or the equivalent) in your language, what is the output? What does that tell you about how the language handles assignments?**
    char [] a = {'c','a','t'}
    char [] b = {'d','o','g'}
    a=b
    b[1] = 'u'
    print a
    print b

    The output of this code, in its C++ equivalent, is one vector that holds d,o,g and another that contains d,u,g. This means that when you assign the value of the first variable to the second variable, the second variable takes that value. However, if you change the value of the second

variable it does not affect the value of the first variable at all. Thus, when something is assigned a value it is unique to that variable.

References: http://www.cplusplus.com/doc/tutorial/functions/
http://www.cplusplus.com/doc/tutorial/variables/