Question 1:

The output is undefined889101.

The value of x is undefined at the beginning of the function, because we declare var x = 10; inside the function at the end. So, because of hoisting we declare x with undefined at first and then we write the value of x before assigning the value.

Value of ‘a’ is 8 because we it’s passed on the function.

The inner function also uses same values of a, b & c as arguments, so when we assign value of a to b, ‘b’ also becomes 8.

The value of ‘b’ after f(a,b,c) is executed is 9, because this is what the value of b is on the outer function.

After we finish calling c(8,9,10), the value of b is 10, because the global variable b has the value of 10, and x will be 1 for the same reason.

Question 2:

Global Scope: Variables declared outside of any function have global scope, which means that these variables can be accessed and modified from any function.

Local Scope: Variables declared inside of a function with ‘var’ keyword have local scope. These variables are available just inside the function which means that these variables can’t be accessed or modified from outside that function.

Question 3:

1. Do statements in Scope A have access to variables defined in Scope B and C?

* No.

1. Do statements in Scope B have access to variables defined in Scope A?

* Yes.

1. Do statements in Scope B have access to variables defined in Scope C?

* No.

1. Do statements in Scope C have access to variables defined in Scope A?

* Yes.

1. Do statements in Scope C have access to variables defined in Scope B?

* Yes.

The reason behind these answers is because JavaScript is lexical scope, inner levels have access to the variables to outer levels, but outer level don’t have access to inner levels variables. In JS, every function gets its own inner scope.

Question 4:

The output is 8125.

When myFunction() is first called we have x = 9, so it will return 81, and the second time x’s value is changed to 5, hence returning 25 when myFunction() is called the second time.

Question 5:

The alert will print out 10.

Because we declared var foo again inside the function, it will be hoisted to the beginning of the function where the foo is yet undefined. Since we have foo variable in the local scope, the function won’t check the global scope foo variable. Foo is undefined at the beginning of the function because no value has been assigned to it yet. So, it will satisfy the if condition and the value 10 is assigned, and alert will just return 10.