**Assignment: 6**

1. undefined889101

undefined //because x is only hoisted and its value isn’t assigned yet in the scope.

8 // because the values of a is 8 while calling c(8,9,10).

8 // because we did b=a in f();

9 // the changes value inside f() isn’t visible outside in c().

10 // global value of b is 10

1 // global value of x is 1

1. Global Scope: The scope/visibility of any program element (variables/functions) defined globally from any other scope.

Eg: var x = 1;

Function f(){

Var y = 15;

}

//x is in the global scope but y is in the local function scope of function f().

Local Scope: The scope/visibility of variables/functions defined inside a function or a block.

1. a) No because variables in Scope B and C are local and local variables can be accessed only within the function or block. You can’t go from out to in due to scope chain.

b) Yes, because variables in Scope A is global and can be accessed from anywhere as nested functions have access to variables declared in their outer scope.

c) No, just like a).

d) Yes, just like b).

e) Yes, because B is the enclosing element of C.

4) 81 and 25 (8125) As there is no declaration of x in MyFunction(), it takes global x which is 9 so returns 9\*9 = 81. Then when x is changed into 5, it returns 5\*5 = 25

5) 10 because foo is defined within the function which is hoisted before execution, so it looks for it and initially it is undefined and since undefined is a falsey, the statement goes inside and sets foo to 10.

6)

const count = {

    counter: 0, //must set the counter here

    add: function() {

        return ++this.counter; //should use "this" keyword

    },

    reset: function() {

        return this.counter = 0;

    }

}

console.log(count.add()); //returns 1

console.log(count.add()); //returns 2

console.log(count.reset()); //returns 0

7) A free variable is a variable used in a function that is not one of its parameters or local variables. There is no free variable in add as counter is a local variable.

8)

function make\_adder(inc) {

    var counter = 0;

    return function() {

        return counter += inc;

    }

};

add5 = make\_adder(5);

add5();

add5();

console.log(add5()); //returns 15

9) Using let or const instead of var while declaring the variables and using function expressions instead of function declarations while declaring the functions will prevent hoisting and remove all the hoisted names from global namespace.

10 and 11) Not studied in class