**Answer Homework7(1-8)**

**QUESTION #1**

X: undefined //x at this point is undefined after hoisting before the initialization

a: 8 // a has a value of 8 which is passed a parameter to the function *c().*

b: 8 // b is assigned the value of a which is 8 within the scope of function *f().*

b: 9 // b is later assigned the value of c which is 9 within the scope of function *f().*

b: 10 // finally b has a value of 10 which is got from the global scope

x: 1 // x has a value of 1 which is got from the global scope

**QUESTION #2**

The Global Scope:-is the scope in which variables or functions are known throughout the application. In other words, variables declared globally have a global scope.

Local Scope:- is the scope where variables or inner functions are known within the function in which they are declared. Variables or functions declared within a function are scoped to that function.

**QUESTION #3**

1. Do statements in Scope A have access to variables defined in Scope B and C? NO
2. Do statements in Scope B have access to variables defined in Scope A? YES
3. Do statements in Scope B have access to variables defined in Scope C? NO
4. Do statements in Scope C have access to variables defined in Scope A? YES
5. Do statements in Scope C have access to variables defined in Scope B? YES

**QUESTION #4**

81: myFunction() returns 81 ,x is 9 is global scope.

25: myFunction() later returns 25 after the value of x has been altered to 5.

**QUESTION #5**

The alert prints out the 10 as the value. This is because, when the function *bar()*

**QUESTION #6**

const countObject = (function(){

var counter = 0;

function add() {

counter += 1;

}

function reset() {

counter = 0;

}

return {

add: add,

reset: reset,

counter: function() {

return counter;

}

}

})();

console.log('Using object approach');

countObject.add();

console.log("count " + countObject.counter());

countObject.reset();

console.log("count " + countObject.counter());

**QUESTION #7**

The free variable is counter.

A free variable is a variable referred to by a function that is not one of its parameters or local variables.

**QUESTION #8**

const make\_adder = function (val) {

let counter = 0;

return

function () {

counter += val;

console.log(counter)

}

};

console.log('Make adder');

const add5 = make\_adder(5);

add5();

add5();

add5();

const add7 = make\_adder(7);

add7(); add7(); add7();