1. Undefined 8 8 9 10 1
2. Any declaration outside function is in global scope while all declarations inside a function are locally scoped.
3. (a) Do statements in Scope A have access to variables defined in Scope B and C? **NO**

(b) Do statements in Scope B have access to variables defined in Scope A? **YES**

(c) Do statements in Scope B have access to variables defined in Scope C? **NO**

(d) Do statements in Scope C have access to variables defined in Scope A? **YES**

(e) Do statements in Scope C have access to variables defined in Scope B? **YES  
4,** 8125

**5**, 10

**6,** var count = (function(){

var counter =0;

return {

add: function(){

return counter += 1;

},

reset: function(){

return counter = 0;

}

}

})();

console.log(count.add());

**7,** A free variable is a variable used within a function, which is neither a formal parameter to the function nor defined in the function's body (and in scope at the point of the variable's use). Counter is free variable in add.

8, add5 = make\_adder(5);

add5( ); add5( ); add5( );

add7 = make\_adder(7);

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

var count = (function(){

var counter =0;

return {

add: function(){

return counter += 1;

},

reset: function(){

return counter = 0;

},

make\_adder : function(inc){

return function(){ return counter +=inc;}

}

}

})();

var add5 = count.make\_addr(5);

console.log(add5());

**9,** Use either of Module Patterns or Object Literals.

**10,**

var employee =

(function(){

//fields

let name;

let age;

let salary;

//getter & setter methods

let setAge = function(newAge){this.age = newAge};

let setSalary = function(newSalary){this.salary = newSalary};

let setName = function(newName){this.name = newName};

let getAge = function(){return this.age;};

let getSalary = function(){return this.salary;};

let getName = function(){return this.name;};

//extra methods

let increaseSalary = function(percentage){

setSalary(getSalary()+(getSalary()\*percentage))

};

let incrementAge = function(){setAge(getAge()+1)};

return {

setName : setName,

setAge : setAge,

setSalary: setSalary,

increaseSalary : increaseSalary,

incrementAge: incrementAge

};

})();

**11,**

var employee =

(function(){

//fields

let name;

let age;

let salary;

//getter & setter methods

let getAge = function(){return age;};

let getSalary = function(){return salary;};

let getName = function(){return name;};

return {

setName : function(newName){name = newName},

setAge : function(newAge){age = newAge},

setSalary: function(newSalary){salary = newSalary},

increaseSalary : function(percentage){salary = getSalary() + (getSalary()\*percentage/100);},

incrementAge: function(){age =getAge()+1;}

};

})();

**12,**

var employee =

(function(){

//fields

let name;

let age;

let salary;

//getter and setter methods

let getAge = function(){return age;};

let getSalary = function(){return salary;};

let getName = function(){return name;};

let empO = {};

empO.setName = function(newName){name = newName};

empO.setAge = function(newAge){age = newAge};

empO.setSalary = function(newSalary){salary = newSalary};

empO.increaseSalary = function(percentage){salary = getSalary() + (getSalary()\*percentage/100);};

empO.incrementAge = function(){age =getAge()+1;};

return empO;

})();

**13,**

employee.address = "";

employee.setAddress = function(newAddress){this.address = newAddress;};

employee.getAddress = function(){return this.address;};

**14,** Error: Hattori

**15**, Success: Hattori

**16**, success error