**LAB ASSIGNMENT NO 7**

**Adnan Shehzad (610534)**

**Closure and Module Pattern**

**Q1: Solution:**

document.write(x) : Undefined

document.write(a): 8

document.write(b) : 8

document.write(b): 9

document.write(b): 10

document.write(x): 1

Q2: Solution:

Global Scope:

Any variable declared outside the function blocks belongs to the global scope and can be accessible from anywhere like

Var x=1;

Const x=Function(){

Var a=0;

}

Here x has the global scope and can be accessed from anywhere.

Local Scope:

Any variable declared inside certain block and which have limited access is knows as local storage. Like

Var x=1;

Const x=Function(){

Var a=0;

}

Now here Var a has the local storage which is only accessible inside the function.

Q3: Solution:

Statement in scope C have access to variables defined in Scope A.

Statements in scope C have access to variables defined in Scope B.

Q4: Solution:

document.write(myFunction()) : 81

document.write(myFunction()) : 25

Q5: Solution:

Alert(foo) : 10

Q6: Solution:

 const x = (function() {

            const count = {

                counter: 0,

                add: function() {

                    return this.counter += 1;

                },

                reset: function() {

                    return this.counter = 0;

                }

            }

            console.log(count.add());

            console.log(count.add());

            console.log(count.add());

            console.log(count.add());

            console.log(count.reset());

            console.log(count.add());

        })();

Q7: Solution:

Free variable are variable that are neither local variable nor passed as a parameter. In the context of closure method

Var counter is the free variable as its not local as well as not passed as parameter.

Q8: Solution:

const make\_adder = function(x) {

let counter = 0

return function() {

counter += x;

}

}

const add5 = make\_adder(5);

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

Q 9: Solution:

If we can simply put all our code in IIFE function , then the name from the global namespace has been removed now and scope changes from global scope to functional scope.

Q 10: Solution:

 <script>

        //Object Literal for qUESTION NO 10

        const Employee = (function() {

            let name = "";

            let age = 23;

            let salary = "";

            function setSalary(salary) {

                this.salary = salary;

            }

            function setage(age) {

                this.age = age;

            }

            function setName(name) {

                this.name = name;

            }

            function getName() {

                return this.name;

            }

            function getAge() {

                return this.age;

            }

            function getSalary() {

                return this.salary;

            }

            function IncreaseSalary(percentage) {

                this.setSalary(this.getSalary + (this.getSalary \* percentage) / 100);

            }

            function incrementAge() {

                this.setage(this.getAge + 1);

            }

            //Public Members

            return {

                setAge: setage,

                setSalary: setSalary,

                setName: setName,

                getName: getName,

                getAge: getAge,

                getSalary: getSalary,

                IncreaseSalary: IncreaseSalary,

                incrementAge: incrementAge

            }

        })();

        Employee.setAge(23);

        Employee.setName("Adnan");

        Employee.setSalary(10000);

        console.log(`Employee Age is ${Employee.getAge()} \n Employee Name is ${Employee.getName()} \n Employee Salary is ${Employee.getSalary()}`);

    </script>

Q11: Solution:

Employee.address=””;

Employee.setAddress=function(adress){

This.address=address;

}

Employee.getAddress=function(){

return this.address;

}