**Javascript Assignment 9**

1. Carefully observe this example.

**a) Is the InnerFunction() a closure?**

Yes

**b) What is output of this program.**

function OuterFunction()

{ var outerVariable = 100;

function InnerFunction() {

alert(outerVariable);

}

return InnerFunction;

}

var innerFunc = OuterFunction();

innerFunc();

**Output:**

C:\Users\ADMIN\Documents\c\JS\_CODE\assign\_9.js:4

alert(outerVariable);

ReferenceError: alert is not defined

**2. What is the difference between a closure and a scope ?**

**Closure:** I. If we define any inner function within another function.

A closure is a function enclosed with references to the variables in its outer scope. Closures allow functions to maintain connections with outer variables, even outside the scope of the variables.

II. function sayWord(word) {

return () => console.log(word);

}

const sayHello = sayWord("hello");

sayHello(); // "hello"

**Scope**: I. a scope in JavaScript defines what variables you have access to. There are two kinds of scope – global scope and local scope. Scope is access. Scope allows to limit access to certain variables to specific areas.

Ex: local scope/function scope example

function sayMyName(){

let myName = "Achal"

console.log(myName) // "Achal"

}

console.log(myName) // undefined //we can’t use variable which is defined inside function, outside the function

**3. What is a lexical scope and how is it related to closure?**

JavaScript implements a scoping mechanism named lexical scoping (or static scoping). Lexical scoping means that the accessibility of variables is determined by the position of the variables inside the nested scopes.

Ex:

const myGlobal = 0;

function func() {

const myVar = 1;

console.log(myGlobal); // logs "0"

function innerOfFunc() {

const myInnerVar = 2;

console.log(myVar, myGlobal); // logs "1 0"

function innerOfInnerOfFunc() {

console.log(myInnerVar, myVar, myGlobal); // logs "2 1 0"

}

innerOfInnerOfFunc();

}

innerOfFunc();

}

func();

**4. Output of following closure ?**

for (var i = 0; i < 3; i++) {

setTimeout(function log() {

console.log(i); // What is logged?

}, 1000);

}

**Output:**

3

3

3