Q.1 Carefully observe this example.
a) Is the InnerFunction() a closure?
b) What is output of this program

```
function OuterFunction()
{ var outerVariable = 100;
function InnerFunction() {
alert(outerVariable);
}
return InnerFunction;
}
var innerFunc = OuterFunction();
innerFunc();
```

Ans:

a) Yes, InnerFuction() is a closure. A closure is a function that has access to its outer function's scope, including variables declared in the outer function. In this case, InnerFunction() accesses the outerVariable declared in OuterFunction().

b) The output of the program is an alert dialog with the message "100". This happens because when **InnerFunction()** is returned from **OuterFunction()** and assigned to **innerFunc**, it still has access to the **outerVariable** from its lexical scope. When **innerFunc()** is called, it alerts the value of **outerVariable**, which is100

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

Ans. In programming, a closure and a scope are related ideas, especially in languages like JavaScript that support functions as first-class citizens.
The term "scope" describes how variables and functions are visible and accessible inside a specific context or code segment.

• Scopes may be local or worldwide. Local scope describes variables and functions that are only visible within a certain block of code, like within a function, whereas global scope refers to variables and functions that are accessible throughout the entire programme.
• While the opposite is not true, variables defined in an outer scope can be accessed by inner scopes. Lexical scoping is the name given to this concept.
• JavaScript employs function scope and block scope, just like a lot of other programming languages.

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

Ans. The concept of lexical scope, which is often referred to as static scope, governs computer languages and establishes the visibility and accessibility of variables according to their declaration locations within the source code. The physical organisation of the code—more especially, the placement of variable declarations—defines the lexical scope.

Regarding lexical scoping:

Global scope variables are those that are defined outside of any function and can be accessed from anywhere in the code.
Unless specifically provided or returned, variables declared inside a function are only available within that function and are part of its local scope.

The variables declared in their outer (enclosing) scope are accessible to nested functions. This implies that variables from outer functions can be accessed by inside functions.

Q.4 Output of following closure?

```
for (var i = 0; i < 3; i++) {
setTimeout(function log() {
console.log(i); // What is logged?
}, 1000);
}
```

Ans:

3

3

3