Calicia Perea
PA5 -Binding of global variables during subroutine passing
March 30, 2023

```
function sub1() {
  var x;
  function sub2() {
    alert(x); // creates a dialog box with value of x
  };
  function sub3() {
    var x;
    x = 3;
    sub4(sub2);
  };
  function sub4(subx) {
    var x;
    x = 4;
    subx();
  };
  x = 1;
  sub3();
};
```

The code defines four nested functions (`sub1()`, `sub2()`, `sub3()`, and `sub4()`) and then calls `sub1()`.

When `sub1()` is called, it initializes a variable x and sets its value to 1. Then it calls `sub3()`.

When `sub3()` is called, it initializes a new variable x and sets its value to 3. It then calls `sub4()` and passes it a reference to `sub2()`.

When `sub4()` is called, it initializes a new variable x and sets its value to 4. It then calls the function that was passed to it as an argument (`subx()`), which is a reference to `sub2()`.

When `sub2()` is called, it displays an alert dialog box with the value of x, which is the value of x in the scope in which `sub2()` was defined. In this case, that value is 1, since `sub2()` was defined inside `sub1()`.

So the output of this code will be an alert dialog box with the value 1.

The code simply defines several nested functions and calls them in sequence.Although JavaScript doesn't have specific ways of binding called Shallow, Deep, or Ad Hoc, it does have ways to pass arguments to functions and to scope variables. Arguments can be passed by value or by reference, while lexical scoping ensures that variables declared inside a function are only visible within that function and any nested functions.