1.

```
1   x = 1;
2   var a = 5;
3   var b = 10;
4   var c = function(a,b,c) {
5       console.log(x);
6       console.log(a);
7       function f(a, b, c){
8           b = a;
9           console.log(b);
10          b = c;
11          var x = 5;
12      }
13      f(a,b,c);
14      console.log(b);
15      var x = 10;
16  }
17
18  c(8, 9, 10);
19  console.log(b);
20  console.log(x);
21
```

Phase 1 – hoisting:

var a, b
var function c(a, b, c) { … }

Phase 2– execution/evaluation of statements and expressions:

x=1
a=5
b=10
call c(8 , 9, 10)

Next, in c() function's lexical environment:

Phase 1– hoisting:

var x
function f(a, b, c) { … }

Phase 2– execution of statements and expressions:

a=8, b=9, c=10 //assign value a, b, c at function call
document.write(x) → undefined (x is hoisted within c())
document.write(a) → 8 (a is assigned at function call)
call f(a, b, c)

Next in f() function's lexical environment:

Phase 1 – hoisting:

var x

Phase 2 execution of statements and expressions:

a=8, b=9, c=10 //assign value a, b, c at function call

b=a //b=8

document.write(b) → 8 (b is assigned to a=8 in the previous line)

b=c //b=9

x=5

//exit f()

document.write(b) → 9 (b is assigned at function call)

x=10

//exit c()

document.write(b) → 10 (b is assigned to 10 in the global scope)

document.write(x) → 1 (x is assigned to 1 in the global scope)

2.

Global scope means all of variables declared outside function we call and these variables have global scope.

Local scope means all of variables declared inside function we call and they have local scope.

3.

```
1   // Scope A
2   function XFunc () {
3     //Scope B
4     function YFunc () {
5       //Scope C
6     };
7   };
8
```

a) No. Because B and C are local scope.

b) Yes. Because A is global scope.

c) No. C is local scope.

d) Yes. Because A is global scope.

e) Yes. By closure.

4.

```
var x = 9;
function myFunction() {
    return x * x;
}
document.write(myFunction());
x = 5;
document.write(myFunction());
```

Phase 1: hoisting

var x
var myFunction() { … }

Phase 2: execution statements and expressions:

x=9
document.write(myFunction()); → 81 (JS Engine goes up to global scope and finds x = 9)
x=5
document.write(myFunction()); → 25 (JS Engine goes up to global scope and finds x = 5)

5.

```
var foo = 1;
function bar() {
        if (!foo) {
                var foo = 10;
        }
        alert(foo);
}
bar();
```

Phase 1: hoisting

Var foo
Var bar = function () { … }

Phase 2: execution statements and expressions:

foo=1
call bar()

Phase 1: hoisting

var foo → JS engine goes up to global scope and finds var foo

Phase 2: execution statements and expressions:

if(!foo) → true because !undefined = true
foo=10
alert(foo) → 10 (foo=10 in the previous if)
//exit