```
1. What will be the output of this code?
    console.log(x);
    var x=5;

    // output -- undefined
```

Explanation: We know that In JS when we declare a variable using var, that declaration is hoisted to the top of its scope, the declaration may be before or after the console.log. In the above code we see that the console.log is executed first and the variable is declared after the console.log so the variable declared will be hoisted at the top of their scope. The events will be considered as declaration first "var x," and the "console. log(x);” next so the output is "undefined".

```
2. What will be the output of this code?
    console.log(a);
    var a;

    // output : undefined
```

Explanation: We know that In JS when we declare a variable using var, that declaration is hoisted to the top of its scope, the declaration may be before or after the console.log. In the above code we see that the console.log is executed first and the variable is declared after the console.log so the variable declared will be hoisted at the top of their scope. The events will be considered as declaration first "var a," and the "console.log(a);" next so the output is"undefined".

```
3. What will be the output of this code?
    console.log(b);
    b=10;
    var b;

 // output : undefined
```

Explanation: Similar to previous question, the variable b is declared using var. This means that the declaration is hoisted to the top of the scope.so when JS Engine compiles the code, it hoists the declaration of b so the events will be considered as declaration first "var b;" and the "console.log(b):* next and it executes as the variable is declared but value is not assigned so the output is "undefined".

```
4. What will happen here?
    console.log(c);

    // output -- ReferenceError: c is not defined
```

Explanation: In the above output we see "ReferenceError: c is not defined" this is because until variables are declared with var, let, const. which are hoisted and initialized to undefined but variables declared with let, const will be hoisted but not initialized. This executing them before their declaration results in "ReferenceError" When the JS engine executing console. log (c);, it looks for the declaration of c. Since there is no declaration of c anywhere in the scope and it wasn't been defined with var, let, or const, the engine gives ReferenceError and give the statement as "c is not defined"

```
6. What will be the output of this code?
    console.log(e);  // output -- undefined
    var e=10;
    console.log(e);  // output -- 10
    e=20;
    console.log(e);   // output -- 20
```

Explanation:
* Here the console.log(e) given above the e declaration and the variable e is declared with var, so the declaration is hoisted to the top of the scope. so it hasn't been assigned any value to the declaration, so the output is undefined here
* Now in the next line value is assigned to e and the output for console.log(e); here is 10.
* Now the value is updated here as e=20; so the output for console log(e); here is 20.

```
7. What will be the output of this code?
    console.log(f);   // output -- undefined
    var f=100;
    var f;
    console.log(f);   // output -- 100
```

Explanation:
* Here the console.log(f) given above the f declaration and the variable f is declared with var, so the declaration is hoisted to the top of the scope. And no value is assigned to the declaration, so the output is undefined here.
* Here, f is assigned with value 100
* And in next line we have declared f, but the f is already declared by this there gonna be nothing change

* Then in the next line console.log(f); will be executed and we get the output as 100 from the f declaration with var.

```
8. What will be the output of this code?
    console.log(g);      // output -- undefined
    var g= g+1;
    console.log(g);      // output -- NaN
```

Explanation:
console.log(g):
* When this line executes, the variable g is declared due to hoisting but not yet assigned a value. So console. log(g); will print output as undefined.
var 9=9+1% :
* Here, we declared a variable g with var and assigned a value g. Then at this point the value is undefined and then adding we are adding +1 to g meaning undefined +1 gives output as NaN (Not a Number) because undefined is not a number so while adding string with a number gives NaN as output.
console. log (g);:
* When console.log(g); is executed again, it prints NaN, which is the current value of g.

```
9. What will be the output of this code?
    var h;
    console.log(h);      // output -- undefined
    h=50;
    console.log(h);      // output -- 50
```

Explanation:
var h;:
* Here the line var h; declares a variable h. At this point, h is created but not initialized with any value.
console.log(h);:

* When this line executes, h has been declared but not assigned any value. Therefore, console.log(h); will print the output as undefined.

```
10. What will be the output of this code?
    console.log(i);      // output -- undefined
    i=10;
    var i = 5;
    console.log(i);      // output -- 5
```

Explanation:
console. log(::
* Here when this line executes, the variable i has been declared (due to hoisting) but not yet initialized with a value.And therefore, console.log(i); will print output as undefined.
i = 10;:
* This line will assign the value 10 to i.