1. undefined 8 8 9 10 1

2. Global scope is defined outside of all the functions, and can be seen inside functions and local scope is the scope inside functions.

3. (a) No, because the variables declared in B or C are local and are not visible in the outer environment.

(b) Yes, because the variables declared in the outer environment of the scope B are global, so they are visible inside of B, meaning that you can access them.

(c) No, because the variables inside of scope C are local and can be visible inside of C.

(d) Yes, because all the scopes inside functions have access in the outer environment's variables.

(e) Yes, because you can access the outer environment variables.

4. 81 and 25

5. 10

6.
```
var add = (function() {
        var counter = 0;
        return function() {
           return counter += 1;
        }
      })();

    var count = (function() {
       let counter = 0;
```

```
    let add = function() {
       counter += 1;
    }

    let reset = function() {
       counter = 0;
    }

    return {
       add: add,
       reset: reset
    }

})();
```

7. Free variables are variables that are not inside of the context of the function closure but are variables that are visible inside of the function closure.

8.
```
add5 = make_adder(5);
    add5();
    add5();
    add5();
    // final counter value is 15
    add7 = make_adder(7);
    add7();
    add7();
    add7();
    // final counter value is 21
```

```
    var make_adder = (function(inc) {
  var counter = 0;
  return function() {
    counter += inc;
    return counter;
  }
})();
```

9. You can remove all of them using the module pattern, so they are not going to be in the global namespace anymore.

10. Private Field: name

Private Field: age

Private Field: salary

Public Method: setAge(newAge)

Public Method: setSalary(newSalary)

Public Method: setName(newName)

Private Method: getAge( )

Private Method: getSalary( )

Private Method: getName( )

Public Method: increaseSalary(percentage)

// uses private getSalary( )

Public Method: incrementAge( )

// uses private getAge( )

```
var Employee = (function() {
  let name, age, salary;
```

```
let setAge = function(a) {
   age = a;
}

let setName = function(n) {
   name = n;
}

let setSalary = function(s) {
   salary = s;
}

let getAge = function() {
   return age;
}
let getName = function() {
   return name;
}
let getSalary = function() {
   return salary;
}
let increaseSalary = function(percentage) {
   setSalary(getSalary + (getSalary() * percentage / 100));
}
let incrementAge = function(inc) {
   setAge(getAge() + inc);
}
return {
   setAge: setAge,
```

```
        setSalary: setSalary,
        setName: setName,
        increaseSalary: increaseSalary,
        incrementAge: incrementAge
    }
})();
```

11.
```
var Employee = (function() {
    let name, age, salary;

    let getAge = function() {
        return age;
    }
    let getName = function() {
        return name;
    }
    let getSalary = function() {
        return salary;
    }

    return {
        setAge: function(a) {
            age = a;
        },
        setSalary: function(s) {
            salary = s;
        },
        setName: = function(n) {
            name = n;
```

```
    },
    increaseSalary: function(percentage) {
      setSalary(getSalary + (getSalary() * percentage / 100));
    },
    incrementAge: function(inc) {
      setAge(getAge() + inc);
    }

  }
})();
```

12.
```
var Employee = (function() {
  let name, age, salary;

  var emp = {};

  emp.setAge = function(a) {
    age = a;
  }
  emp.setName = function(n) {
    name = n;
  }
  emp.setSalary = function(s) {
    salary = s;
  }

  let getAge = function() {
    return age;
  }
```

```
    let getName = function() {
        return name;
    }
    let getSalary = function() {
        return salary;
    }
    emp.increaseSalary = function(percentage) {
        setSalary(getSalary + (getSalary() * percentage / 100));
    }
    emp.incrementAge = function(inc) {
        setAge(getAge() + inc);
    }
    return emp;

})();
```

13. Employee.address = “North st”;

```
Employee.setAddress = function(a) {
    this.address = a;
}
Employee.getAddress = function() {
    return this.address;
}
```

14. It’s going to execute the promise and return an alert with “Error: Hattori”

15. It’s going to show an alert with “Success Hattori” and Once the promise has settled, it cannot be changed.

Rejecting it after 500 mili seconds will not show an alert with “Error: Yoshi” because it will not effect the event handler.

16. success

error

Error caught