TASK-2

1. What will be the output of the code?

```
let x=5;

let y=x;

x=10;

console.log(x);

console.log(y);
```

Answer:x value is 10

Y value is 5

Reason: Here's the reasoning: Initially, y is assigned the value of x, which is 5. When x is later changed to 10, it doesn't affect y because y holds a copy of the original value. Therefore, console.log(x) outputs 10, while console.log(y) outputs 5.

2. What will be the output of the code?

```
let obj1 = { name : "Alice"};

let obj2 = obj1;

obj1.name ="bob";

console.log(obj1.name);

console.log(obj2.name);
```

answer:bob

bob

Reason: Here's the reasoning: obj2 is assigned a reference to obj1, so both variables point to the same object in memory. When obj1.name is changed to "bob", it updates the shared object, making obj2.name also reflect this change. Thus, both console.log statements output "bob".

3. What will be the output of the code?

```
let a = "hello";

let b = 42;

let c = true;

let d = {key:"value"};

let e = null;

let f = undefined;

console.log(typeof a);

console.log(typeof b);

console.log(typeof  c);

console.log(typeof d);

console.log(typeof e);

console.log(typeof f);
```

Answer: string

number

boolean

object

object

undefined

Reason: Here's the reasoning: The typeof operator returns a string indicating the type of the unevaluated operand. For a, it returns "string"; for b, it returns "number"; for c, it returns "boolean"; for d, it returns "object" (since objects are of type object); for e, it also returns "object" (due to how null is treated in JavaScript); and for f, it returns "undefined"

4. What will be the output of the code?

```
let numbers = [10,20,30,40,50];

console.log(numbers[2]);
```

console.log(numbers[0]);

console.log(numbers[numbers.length - 1]);

Answer:30

10

50

Reason: Here's the reasoning: numbers[2] accesses the third element of the array, which is 30. numbers[0] accesses the first element, which is 10. Finally, numbers[numbers.length - 1] uses the length of the array to access the last element, 50.

5. What will be the output of the code?

let fruits = ["apple","banana","mango"];

fruits[1] = "orange";

console.log(fruits);

Answer: ['apple', 'orange', 'mango']

Reason: Here's the reasoning: The code modifies the second element of the fruits array (index 1) from "banana" to "orange". When console.log(fruits) is called, it displays the updated array with "orange" in place of "banana", while the other elements remain unchanged. Thus, the final array is ["apple", "orange", "mango"].

6. What will be the output of the code?

let matrix = [

[1, 2, 3],

[4, 5, 6],

[8, 9, 10]

];

console.log(matrix[1][2]);

console.log(matrix[2][0]);

Answer:6

8

Reason: Here's the reasoning: matrix[1][2] accesses the second row (index 1) and the third element (index 2), which is 6. Similarly, matrix[2][0] accesses the third row (index 2) and the first element (index 0), which is 8. Thus, the outputs are 6 and 8, respectively.

7. What will be the output of the code?

```
let person ={
   name: "john",
   age:25,
   city :"new york"
};
console.log(person.name);
console.log(person.age);
```

Answer: john

25

Reason: Here's the reasoning: The code accesses properties of the person object using dot notation. person.name retrieves the value associated with the name property, which is "john", while person.age retrieves the value associated with the age property, which is 25. Therefore, the console logs display "john" and 25.

8. What will be the output of the code?

```
let car = {
   make:"toyato",
   model:"corolla",
   year:2021
};
console.log(car["make"]);
```

console.log(car["model"]);

Answer: toyato

corolla

Reason: Here's the reasoning: The code accesses properties of the car object using bracket notation. car["make"] retrieves the value of the make property, which is "toyato", and car["model"] retrieves the value of the model property, which is "corolla". Thus, the console logs display "toyato" and "corolla".

9. What will be the output of the code?

```
let book =
{
   title:"the great gatsby",
   author:"f.scott fitzgerloand"
};
book.author = "anonymous";
console.log(book.author);
```

Answer: anonymous

Reason: Here's the reasoning: The code creates a book object with an author property initially set to "f.scott fitzgerloand". The line book.author = "anonymous" updates the author property to "anonymous". Therefore, when console.log(book.author) is executed, it displays the updated value, which is "anonymous"

10. What will be the output of the code?

```
let student = {
   name:"alice",
   grade:"A"
};
student.age = 20;
```

console.log(student);

Answer: { name: 'alice', grade: 'A', age: 20 }

Reason: Here's the reasoning: The code creates a student object with properties name and grade. The line student.age = 20 adds a new property, age, to the object. When console.log(student) is executed, it displays the entire object, including the newly added age property