***Task II***

***DIFFERENCE BETWEEN COPY BY VALUE AND COPY BY REFERENCE :***

A distinct **copy of** the **value is** passed to the function; changes to it have no effect outside the function. A **reference** to the **value is** passed to the function. If the function modifies the **value** through the passed **reference**, the modification **is** visible outside the function.

***What happens in the memory after we assigning a value to it?***

**Copy by value**

In a primitive data-type when a variable is assigned a value we can imagine that a box is created in the memory. This box has a sticker attached to it i.e. the variable name. Inside the box the value assigned to the variable is stored.

Var x=17;

Var y=’xyz’;

Var z=null;

Where 17 and xyz are the values for the variables x and y respectively .

Var x=17;

Var y=’xyz’;

Var z=null;

Var a=x;

Var b=y;

Here a,b variables got the values of x and y .

The changes made in a and b wont affect or impact the variables x,y

This phenomenon of copying values is known as call by value.

***Copy by reference:***

Let user={name : “ram”};

Let admin=user;

Admin.name=”shyam”;// value changes

Alert(user.name);//name change to shyam

The change in one variables affects the another variable while copy

As both variables are storing the address of the memory location.

When one changes the values in the allocated memory it is reflected in the other as well.

Copy by reference is like having two keys of the same room shared between two variables.

**How do you copy by value a composite data type?**

There are 3 ways to copy by value for composite data types.

1. Using the spread (...) operator
2. Using the Object.assign() method
3. Using the JSON.stringify() and JSON.parse() methods

***SPREAD OPERATOR:***

Let a=[1,2,3];

Let b=[…a];

Altering values in b wont affect the elements in a;

***OBJECT.ASSIGN():***

Let b=object.assign([],a);

***JSON.STRINGIFY() AND JSON.PARSE():***

LET B=JSON.parse(JSON.stringify(a));