Exercise 1:

He decidido utilizar Javascript para poder trabajar desde 0 sin necesidad de apoyarme en el ejemplo dado.

A)

```
//Bucle Version
function IsPalindrome(text)
{
    //Se declaran las variables lengthword para saber la longitud del string, y revword donde se almacenará la palabra invertida.
    var lengthword = text.length;
    var revword = '';

    //A través del for simulando un for inverso se rellena la palabra revword con ayuda de lengthword que nos permite recorrer la palabra desde el final hacia el principio.
    for(i in text)
    {
        lengthword  -= 1;
        revword += text[lengthword];
    }

    //Se compara el texto invertido con el texto original.
    return revword == text;
}
console.log(IsPalindrome('reconocer'));


//NoBucle Version
function IsPalindromeNoLoop(text)
{
    //Se divide el texto y se junta de manera invertida.
    var revword = text.split('').reverse().join('');

    //Se compara el texto invertido con el texto original.
    return revword == text;
}
console.log(IsPalindromeNoLoop('radar'));
```

B)

La versión con bucle es menos óptima ya que el número de operaciones que ha de hacer es mayor, ya que realiza una comprobación y modificación de la variable cada vez que entra en el bucle. Por otra parte la opción sin bucle es más directa y limpia y solo ha de hacer una operación,aunque la carga en memoria podría ser mayor a la del bucle al realizar la obtención de la palabra invertida ya que esta se hace de golpe en vez de progresivamente.

Exercise 2:

Para este ejercicio he decidido hacer uso de C#, y así hacer algo en un lenguaje diferente al del ejercicio anterior en el que he usado Javascript, como se requería de una simulación he optado por hacer uso de Unity y poder mostrar directamente el resultado, he enlazado imágenes y un vídeo en el que enseño el resultado final. He optado también por tener escrito el código en un solo script y así centralizar todo el ejercicio, aunque en otra situación habría separado los objetos del script de movimiento.

Resultado final:
https://youtu.be/lx1iMirrZM0

A) Los usuarios podrán hacer uso de las variables públicas para elegir el vehículo que desean seleccionar, he decidido que la simulación fuera lo más visual posible, las propiedades de cada vehículo se asignan al iniciar la simulación mientras que el movimiento se asigna y se actualiza en la función Update, en tiempo real.

```
using System;
using System.Collections;
using System.Collections.Generic;
using UnityEngine;
using UnityEngine.UI;

public class testvehicleA : MonoBehaviour
{
    //Variables públicas
    public enum vehicleType {coche , moto}; //Lista de tipos de vehículos.
    public vehicleType vehicle;
    public Text nameLabel, wheelLabel, speedLabel; //Etiquetas para
visualizar los distintos tipos de estados del vehiculo.

    //Variables privadas
    private string vehicleName =""; //Nombre del vehículo
    private int speedVehicleModifier = 1; //Velocidad máxima del vehículo
    private int speedVehicle = 0; //Velocidad actual del vehículo
    private int wheelNumber =0; //Número de ruedas del vehículo

    void Start()
    {
        //Al iniciar se asignan los diferentes vehículos y sus propiedades
según los valores seleccionados por el usuario.

        switch(vehicle)
            {
            case vehicleType.coche:
                vehicleName = "coche";
                wheelNumber = 4;
                break;
```

```
            case vehicleType.moto:
                vehicleName = "moto";
                wheelNumber = 2;
                break;
            }

            //Se asigna el valor de las etiquetas en base al valor
introducido.
            nameLabel.text = vehicleName.ToString();
            wheelLabel.text = wheelNumber.ToString();
    }

    void Update()
    {
        //Movimiento del vehiculo

        gameObject.transform.position = new Vector2 (transform.position.x +
(speedVehicle),transform.position.y);

        if(Input.GetButtonDown("Accelerate"))speedVehicle =
speedVehicleModifier;

        //Se asignan las etiquetas de velocidad.
        speedLabel.text = speedVehicle.ToString();
    }
}
```

B)Se añade un input nuevo que frene al vehículo y asigne su velocidad a 0, similar a como se ha implementado el input de acelerar.

```
    void Update()
    {
        //Movimiento del vehiculo

        gameObject.transform.position = new Vector2 (transform.position.x +
(speedVehicle),transform.position.y);

        if(Input.GetButtonDown("Accelerate"))speedVehicle =
speedVehicleModifier;

        //Al pulsar esté botón el vehículo frena y cesa su movimiento.
        if(Input.GetButtonDown("Cancel"))speedVehicle =0;

        //Se asignan las etiquetas de velocidad.
        speedLabel.text = speedVehicle.ToString();
    }
}
```

C)Se añade el vehículo bicicleta, se añade de igual manera que se ha añadido el resto de vehículos. También he decidido añadir que la velocidad sea diferente dependiendo del vehículo en cuestión.

```
using System;
using System.Collections;
using System.Collections.Generic;
using UnityEngine;
using UnityEngine.UI;

public class testvehicleA : MonoBehaviour
{
    //Variables públicas
    public enum vehicleType {coche , moto, bicicleta}; //Lista de tipos de
vehículos.
    public vehicleType vehicle;
    public Text nameLabel, wheelLabel, speedLabel; //Etiquetas para
visualizar los distintos tipos de estados del vehiculo.

    //Variables privadas
    private string vehicleName =""; //Nombre del vehículo
    private int speedVehicleModifier = 1; //Velocidad máxima del vehículo
    private int speedVehicle = 0; //Velocidad actual del vehículo
    private int wheelNumber =0; //Número de ruedas del vehículo

    void Start()
    {
        //Al iniciar se asignan los diferentes vehículos y sus propiedades
según los valores seleccionados por el usuario.

        switch(vehicle)
            {
            case vehicleType.coche:
                vehicleName = "coche";
                speedVehicleModifier = 3;
                wheelNumber = 4;
                break;

            case vehicleType.moto:
                vehicleName = "moto";
                speedVehicleModifier = 2;
                wheelNumber = 2;
                break;

            case vehicleType.bicicleta:
                vehicleName = "bicicleta";
                speedVehicleModifier = 1;
                wheelNumber = 2;
```

```
                break;
            }

            //Se asigna el valor de las etiquetas en base al valor
introducido.
            nameLabel.text = vehicleName.ToString();
            wheelLabel.text = wheelNumber.ToString();
    }

    void Update()
    {
        //Movimiento del vehiculo

        gameObject.transform.position = new Vector2 (transform.position.x +
(speedVehicle),transform.position.y);

        if(Input.GetButtonDown("Accelerate"))speedVehicle =
speedVehicleModifier;

        //Al pulsar esté botón el vehiculo frena y cesa su movimiento.
        if(Input.GetButtonDown("Stop"))speedVehicle =0;

        //Se asignan las etiquetas de velocidad.
        speedLabel.text = speedVehicle.ToString();
    }
}
```

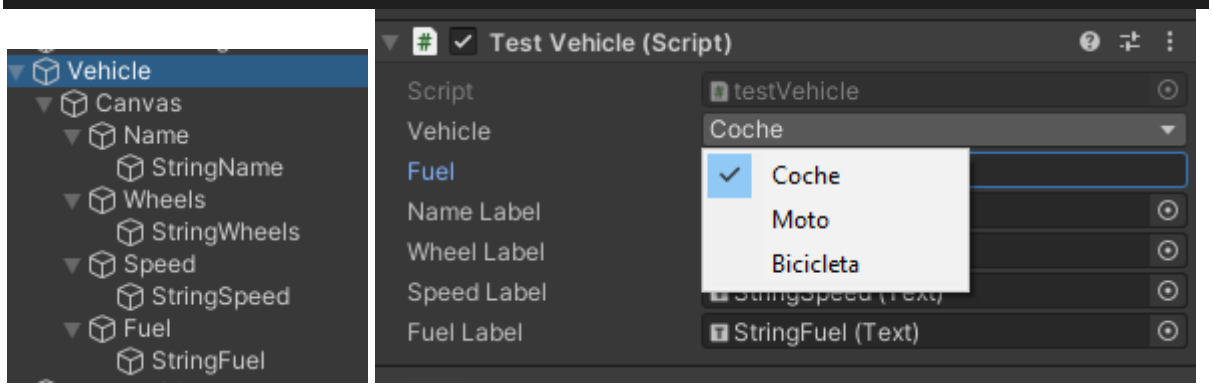
D) Se añade el componente de repostar. Esto requiere un cambio de enfoque a la estructura del código. Ya que ahora contemplo que un vehículo pueda tener o no gasolina, y pueda o no repostar.

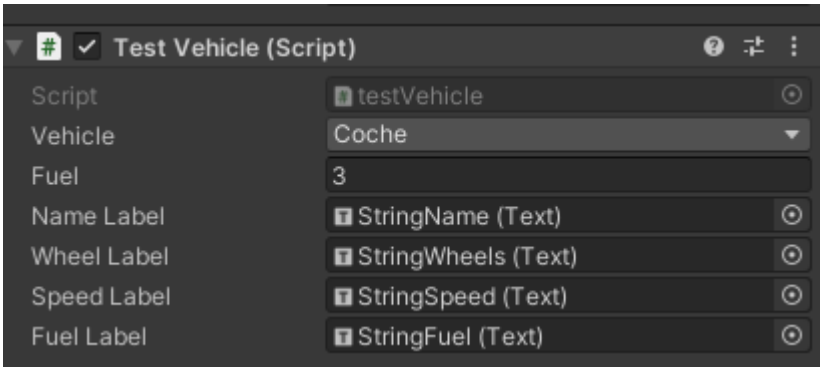
```
using System;
using System.Collections;
using System.Collections.Generic;
using UnityEngine;
using UnityEngine.UI;

public class testVehicle : MonoBehaviour
{
    //Variables públicas
    public enum vehicleType {coche , moto, bicicleta}; //Lista de tipos de
vehículos.
    public vehicleType vehicle;
    public float fuel =0; //Cantidad máxima de combustible del vehiculo, se
utilizara como contador.
    public Text nameLabel, wheelLabel, speedLabel, fuelLabel; //Etiquetas
para visualizar los distintos tipos de estados del vehiculo.
```

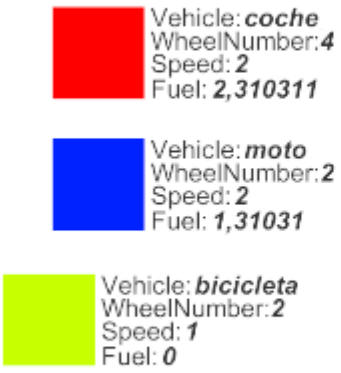
```
    //Variables privadas
    private string vehicleName =""; //Nombre del vehículo
    private int speedVehicleModifier = 1; //Velocidad máxima del vehículo
    private int speedVehicle = 0; //Velocidad actual del vehículo
    private int wheelNumber =0; //Número de ruedas del vehículo
    private float fuelconsume=0.0f; //Cantidad del combustible a consumir
    private bool repost = false; //Indica si el vehículo necesita repostar o
no
    private bool moving = false; //Indica si el vehículo esta en movimiento o
esta quieto.

    void Start()
    {
        //Al iniciar se asignan los diferentes vehículos y sus propiedades
según los valores seleccionados por el usuario.

        fuelconsume = fuel; //Se llena el combustible de los vehiculos con su
combustible máximo.

        switch(vehicle)
            {
            case vehicleType.coche:
                vehicleName = "coche";
                speedVehicleModifier = 2;
                wheelNumber = 4;
                repost = true;
                break;

            case vehicleType.moto:
                vehicleName = "moto";
                speedVehicleModifier = 2;
                wheelNumber = 2;
                repost = true;
                break;

            case vehicleType.bicicleta:
                vehicleName = "bicicleta";
                speedVehicleModifier = 1;
                wheelNumber = 2;
                repost = false;
                break;
            }

            //Se asigna el valor de las etiquetas en base al valor
introducido.
            nameLabel.text = vehicleName.ToString();
            wheelLabel.text = wheelNumber.ToString();
    }
```

```
    void Update()
    {
        //En caso de moverse el vehiculo cambiara de posición y en caso de
que necesite gasolina para funcionar, esta se ira consumiendo.
        if (moving == true){
            speedVehicle = speedVehicleModifier;
            gameObject.transform.position = new Vector2 (transform.position.x
+ (speedVehicle),transform.position.y);

            if (repost == true){
                fuelconsume -= Time.deltaTime;
                if (fuelconsume <= 0.0f)
                {
                    moving = false;
                    speedVehicle =0;
                }
            }
        }

        // Se comprueba si el vehiculo en cuestión necesita gasolina o no
antes de arrancar.
        switch(repost)
            {
            case true:
                if(Input.GetButtonDown("Accelerate") && (fuelconsume >
0.0f))moving = true;
                break;

            case false:
                if(Input.GetButtonDown("Accelerate"))moving = true;
                break;
            }

        //Al pulsar esté botón el vehiculo frena y cesa su movimiento.
        if(Input.GetButtonDown("Stop"))
        {
            speedVehicle =0;
            moving = false;
        }

        //Al pulsar esté botón el vehiculo repostara y cesa su movimiento.
        if(Input.GetButtonDown("RepostFuel"))fuelconsume = fuel;

        //Se asignan las etiquetas de velocidad y gasolinas que estaran
actualizandose en tiempo real.
        speedLabel.text = speedVehicle.ToString();
        if (fuelconsume <= 0.0f)fuelLabel.text = "0";
```

```
        else fuelLabel.text = fuelconsume.ToString();
    }
}
```







Exercise 3:

A):

```
SELECT CustomerID, CustomerName, Address, PostalCode
FROM Customers
WHERE PostalCode LIKE '10%';
```

o

```
SELECT  *
FROM Customers
WHERE PostalCode LIKE '10%';
```

B)

```
SELECT Customers.CustomerID, Customers.CustomerName, Customers.Address,
Customers.PostalCode , count (Orders.CustomerID) as NumOrders
FROM Customers
LEFT JOIN Orders ON Orders.CustomerID=Customers.CustomerID
GROUP BY Customers.CustomerID;
```

o

```
SELECT Customers.* , count (Orders.CustomerID) as NumOrders
FROM Customers
LEFT JOIN Orders ON Orders.CustomerID=Customers.CustomerID
GROUP BY Customers.CustomerID;
```