PRUEBA TÉCNICA (Tiempo Máx 2.5 Horas)

NOMBRE COMPLETO: _____ IDENTIFICACION: _____

La siguiente prueba es de selección múltiple con única respuesta. Por favor seleccione una única respuesta. **La pregunta nro. 14 es abierta y de carácter obligatorio.**

```
public class Persona
{
      public string identificacion;
      public string nombres;
      public string apellidos;
      public DateTime FechaNacimiento;
}
```

VECTOR PERSONAS:

1	2	3	4	5
Identificación: 79.658.321 Nombres: Pedro Estiven Apellidos: Gil, Barón Fecha Nmto.: 26/10/1979	Identificación: 41.582.329 Nombres: Ana María Apellidos: López, Torres Fecha Nmto.: 12/08/1954	Identificación: 84.632.206 Nombres: Eugenio Apellidos: Joya, Rivera Fecha Nmto.: 19/01/1984	Identificación: 7.456.977 Nombres: Carol Johanna Apellidos: Pérez, Castro Fecha Nmto.: 05/02/1975	Identificación: 15.608.542 Nombres: Pablo Raúl Apellidos: Téllez, Sánchez Fecha Nmto.: 19/01/1949

ALGORITMO

```
public static void main(string[] args) {
      List<Persona> personas = new List<Persona>();
      personas.Add("79658321", "Pedro Estiven", "Gil Barón",        DateTime(1979,
      10, 26));
      personas.Add("41582329", "Ana María", "López Torres", DateTime(1954, 8, 12));
      personas.Add("84632206", "Eugenio", "Joya Rivera", DateTime(1984, 5, 17));
      personas.Add("7456977", "Carol Johanna", "Pérez Castro", DateTime(1975, 2,
      5));
      personas.Add("15608542", "Pablo Raúl", "Téllez Sánchez", DateTime(1949, 1,
      19));
      string result = ProcesarPersona(personas);
}
```

```
public string ProcesarPersona (List<Persona> personas)
{
      decimal Resultado1 =  0;
      decimal Resultado2 = 0;
      int valor = 0;
      DateTime fechaActual = new DateTime(2018, 3, 23);
      foreach (Persona p in personas) {
            valor = calcularEdad(p.FechaNacimiento, fechaActual);
            if(valor>Resultado1){Resultado1=valor;}
            Resultado2 = Resultado2 + valor;
      }
      Resultado2 = Resultado2/personas.Length;
      return "Resultado 1:" + Resultado1 + "Resultado 2:" + Resultado2;
}
```

Dado el vector anterior y el algoritmo anterior, resuelva las preguntas 1 y 2:

1. ¿Cuál es el resultado después de ejecutar el algoritmo?

a. Resultado 1: 63.5, Resultado 2: 63
b. Resultado 1: 49.5, Resultado 2: 42
c. Resultado 1: 49, Resultado 2: 33
d. Resultado 1: 69, Resultado 2: 49

2. ¿Cuál de las siguientes opciones muestra la información completa de la persona?

a. `Console.writeline ("Identificación: {0};Nombres: {1};Apellidos: {2};Edad: {3}", p.identificacion, p.nombres, p.apellidos, calcularEdad(p.FechaNacimiento, fechaActual));`

b. `Console.writeline ("Identificación: " + p.identificacion + " Nombres: " + p.nombres + "Apellidos: " + p.apellidos + "Edad: " + p.edad);`

c. `Console.writeline (System.append ("Identificación: ", p.identificacion, " Nombres: ", p.nombres, "Apellidos: ", p.apellidos, "Edad: ", p.edad));`

d. a y b son correctas.

3. El concepto de las Siglas "M.V.C." corresponde con la siguiente definición:

a. Es un patrón de diseño útil para componentes de presentación.
b. Es una metodología de desarrollo para implementar proyectos de Ingeniería de Software que utilizan los equipos ágiles.
c. Es una tecnología propietaria de Microsoft.
d. Es un patrón de diseño encargado de establecer una estructura conceptual y tecnológica con artefactos o módulos.

4. ¿Cuál de los siguientes patrones de diseño garantiza que un objeto se pueda instanciar una única vez en tiempo de ejecución?

a. El patrón de diseño “Factory”
b. El patrón de diseño “Unique Object”
c. El patrón de diseño “Repository”
d. Ninguna de las anteriores

5. ¿Cuál de las siguientes definiciones corresponde a las siglas “O.R.M.”?

a. Es una abstracción de la base de datos con modelos específicos que representan Disparadores, Procedimientos Almacenados, Tablas, Vistas, Esquemas, Tipos de Datos, Funciones, Variables, Índices, Llaves Foráneas.
b. Es una técnica de programación que permite convertir datos entre una base de datos relacional y el sistema de tipos soportado por un lenguaje de programación.
c. Es un modelo que permite representar cualquier abstracción y conocimiento de un sistema de información formado por conjuntos de objetos.
d. Es un componente propietario de Microsoft que agrega capacidades de consulta a datos de manera nativa en los lenguajes de .NET

De acuerdo las siguientes tablas de base de datos, responder las preguntas 6, 7, 8 y 9:

ORDENES

ID	EmpleadoID	DetalleOrdenID
10248	1	1
10249	2	2
10250	3	3
10251	1	4
10252	3	5
10253	3	6
10254	3	7
10255	2	8
10255	3	9
10255	1	10
10255	2	11
10255	2	12
10255	2	13
10255	2	14

EMPLEADOS

ID	Apellido	Nombre	FechaNacimiento	Teléfono
1	Acosta	Fabian	8/12/1968	3003625986
2	Pérez	Luis Antonio	19/02/1952	362532563
3	Bunny	Lucy	30/08/1963	965322514
4	Toreto	Dominic	30/08/1970	362563321
5	Lopéz	Jose Luis	30/09/1983	7879854

6. ¿Cuál de las siguientes sentencias SQL, cumple con la siguiente consulta?: Consultar lista de empleados que tienen registrados un número de órdenes mayor o igual a cinco (5).

a.
```
SELECT Empleados.Nombre, COUNT(Ordenes.ID) AS
CantidadOrden
FROM (Ordenes
INNER JOIN Empleados ON Ordenes.EmpleadoId =
Empleados.ID)
HAVING COUNT(Ordenes.ID) >= 5
GROUP BY Empleados.Nombre;
```

b.
```
SELECT Empleados.Nombre, COUNT(Ordenes.ID) AS
CantidadOrden
FROM (Ordenes
INNER JOIN Empleados ON Ordenes.EmpleadoId =
Empleados.ID)
GROUP BY Empleados.Nombre
HAVING COUNT(Ordenes.ID) >= 5;
```

c.
```
SELECT Empleados.Nombre, Ordenes.OrdenId AS
CantidadOrden
FROM (Ordenes
INNER JOIN Empleados ON Ordenes.EmpleadoId =
Empleados.ID)
WHERE COUNT(Ordenes.ID) >= 5;
```

d.
```
SELECT Empleados.Nombre, SUM(Ordenes.ID) AS
CantidadOrden
FROM (Ordenes
INNER JOIN Empleados ON Ordenes.EmpleadoId =
Empleados.ID)
GROUP BY Empleados.Nombre
HAVING COUNT(Ordenes.ID) >= 5;
```

7. Consultar los empleados por nombre y apellido, cuya edad no se encuentren en el rango entre 40 y 50 años. ¿Cuál de las siguientes sentencias SQL, cumple con la condición anterior?:

a.
```
SELECT Empleados.Nombre, Empleados.Apellido
FROM Empleados
WHERE (datediff(yy,Empleados.FechaNacimiento,GETDATE())
not in (40,50);
```

b.
```
SELECT Empleados.Nombre, Empleados.Apellido
FROM Empleados
WHERE (datediff(dd,Empleados.FechaNacimiento,GETDATE())
> 40 and
(datediff(dd,Empleados.FechaNacimiento,GETDATE()) < 50
```

c.
```
SELECT Empleados.Nombre, Empleados.Apellido
FROM Empleados
WHERE (datediff(yy,Empleados.FechaNacimiento,GETDATE())
< 40 and
(datediff(yy,Empleados.FechaNacimiento,GETDATE()) > 50
```

d.
```
SELECT Empleados.Nombre, Empleados.Apellido
FROM Empleados
WHERE (datediff(yy,Empleados.FechaNacimiento,GETDATE())
not Between (40,50);
```

8. Consultar empleados cuyos nombres comienzan con la palabra "Lu". ¿Cuál de las siguientes sentencias SQL, cumple con la condición anterior?:

a.
```
SELECT  Empleado.Nombre
FROM Empleado
WHERE Empleado.Nombre IN (“Lu”);
```

b.
```
SELECT  Empleado.Nombre
FROM Empleado
WHERE Empleado.Nombre BEGIN WITH “Lu”;
```

c.
```
SELECT  Empleado.Nombre
FROM Empleado
WHERE Empleado.Nombre LIKE “Lu%”;
```

d.
```
SELECT  Empleado.Nombre
FROM Empleado
WHERE Empleado.Nombre STARTS WITH “Lu”;
```

9. Realizar una consulta que contenga el nombre del empleado y la cantidad de órdenes que tenga asociadas, adicionalmente, se deben incluir los empleados que no contienen órdenes asociadas. ¿Cuál de las siguientes sentencias SQL, cumple con la condición anterior?:

a.
```
SELECT  Empleado.Nombre, ISNULL(Count(Orden.ID),0)
as CantidadOrden
FROM Empleado, Orden
WHERE Empleado.ID = Orden.EmpleadoID
Group By Empleado.Nombre;
```

b.
```
SELECT  Empleado.Nombre,
ISNULL(Count(Orden.ID),0)) as CantidadOrden
FROM Empleado INNER JOIN Orden ON Empleado.ID =
Orden.EmpleadoID
Group By Empleado.Nombre;
```

c.
```
SELECT  Empleado.Nombre,
ISNULL(Count(Orden.ID),0)) as CantidadOrden
FROM Empleado RIGHT JOIN Orden ON Empleado.ID =
Orden.EmpleadoID
Group By Empleado.Nombre;
```

d.
```
SELECT  Empleado.Nombre, ISNULL(Count(Orden.ID),0)
as CantidadOrden
FROM Empleado LEFT JOIN Orden ON Empleado.ID =
Orden.EmpleadoID
Group By Empleado.Nombre;
```

HTML:

```
<ul id="listado">

</ul>

<input type="text" id="Nombres" />

<span id="Seleccionado"></span>
```

JS:

```
var listItems  = [
   "Diane",
   "Louis",
   "Fernando",
   "Jean",
    "Armando",
    "Fernanda"
];
```

10. Dado los elementos alojados en la variable listItems, seleccione el bloque de código que permita filtrar los elementos contenidos de la variable listItems. Ejemplo: Si el usuario filtra por la cadena "Fer", se listarían Fernando y Fernanda. Si inserta la cadena "do", se listarían "Fernando" y "Armando".

a.
```
for(var i=0;i<listItems.length;i++){
  $('#listado').append('<li
id='+listItems[i]+'>'+listItems[i]+'</li>')
}
$('#Nombres').autocomplete({
  source:listItems,
  minLength:1,
  select:function(event,ui){
    $('#Seleccionado').text('Seleccionado :
'+ui.item.label)
  }
})
```

b.
```
for(var i=0;i<listItems.length;i++){
  $('#listado').append('<li
id='+listItems[i]+'>'+listItems[i]+'</li>')
}
$('#Nombres').where({
  source:listItems,
  minLength:1,
  select:function(event,ui){
    $('#Seleccionado').text('Seleccionado :
'+ui.item.label)
  }
})
```

c.
```
for(var i=0;i<listItems.length;i++){
  $('#listado').append('<li
id='+listItems[i]+'>'+listItems[i]+'</li>')
}
$('#Nombres').where ({
  source:listItems,
  minLength:1,
  select:function(event,ui){
    $('#Seleccionar).text('Seleccionar :
'+ui.item.label)
  }
})
```

d.
```
for(var i=lista.length;i>0;i--){
  $('#listado').append('<li id='+ lista [i]+'>'+
lista [i]+'</li>')
}
$('#Nombres').autocomplete({
  source: lista,
  minLength:1,
  select:function(event,ui){
    $('#Seleccionar).text('Seleccionado :
'+ui.item.label)
  }
})
```

11. Dados el siguiente código en JavaScript y las siguientes sentencias. Seleccione las sentencias que al sustituirlas en el código coinciden con la funcionalidad que representa el algoritmo que valida y crea un objeto que describe características de una persona.

```
function CrearDescripcionPersonal(nombre, fechaNacimiento, identificacion, genero) {
    if (nombre.length < 20)
        return { descripcionError: "el nombre no puede contener menos de 20 letras" };
    if (__________I__________) == null)
        return { descripcionError: "el nombre no puede contener caracteres distintos a letras" };
    if (_________II__________) {
        return { descripcionError: "la identificacion debe contener entre 10 y 20 números" };
    }
    if (genero.match(/^H$|^M$/) == null) {
        return_________III_________________
    }
    return {
        nombre: nombre,
        fechaNacimiento: fechaNacimiento,
        identificacion: identificacion,
        genero: genero.includes("H") ? Genero.Hombre : Genero.Mujer
    };
}
```

```
1.    nombre.match(/^[a-zA-Z]+$/)
2.    descripcionError.value="el género sólo puede ser H o M";
3.    identificacion.length < 10 || identificacion.length > 20
4.    { excepcionError: "el género sólo puede ser Macho o Hembra" };
5.    nombre.match(/^a-Z$/)
6.     { descripcionError: "el género no puede contener caracteres distintos a letras" };
7.    identificacion.length >= 10 && identificacion.length <= 20
8.    match(nombre, /^a-Z$/)
9.    (nombre !== "a-z" && nombre !== "A-Z")
10.    10 < identificacion.length < 20
11.    { descripcionError: "el género sólo puede ser H o M" };
```

a. I:5, II:3 y III:2
b. I:9, II:10 y III:4
c. I:8, II:7 y III:6
d. I:1, II:3 y III:11

12. Seleccione cuál de las siguientes es una sentencia LINQ válida:

a. **var** consulta = **from** num in numeros **where** (num % 2 == 0) **select** num;
b. a y d son correctas.
c. **var** consulta = **select** persona **from** personas **where** persona.nombre = “Maria”;
d. **var** consulta = precios.Where (precio => precio < 50000).Select**(**precio => precio);

13. Suponga que usted está desarrollando un servicio Windows Communication Foundation (WCF) que contiene el siguiente contrato de operación:

```
[OperationContract]
NombreClientes ObtenerNombresClientes();
```

La operación devuelve los nombres de los clientes. Necesita desarrollar una definición para el contrato de operación que produce un XML con la siguiente estructura:

```
<s:Envelope xmlns:s="http://schemas.xmlsoap.org/soap/envelope/">
<s:Header />
 <s:Body>
 <Nombres xmlns=http://tempuri.org

xmlns:a="http://schemas.microsoft.com/2003/10/Serialization/Arrays"
            xmlns:i="http://www.w3.org/2001/XMLSchema-instance">
            <a:string>Cliente1</a:string>
            <a:string>Cliente2</a:string>
            <a:string>Cliente3</a:string>
 </Nombres>
 </s:Body>
</s:Envelope>
```

¿Qué segmento de código se debe utilizar?

a. `[MessageContract(IsWrapped = false)] public class NombreClientes { [MessageBodyMember] public string[] Nombres; }`
b. `[MessageContract(WrapperName = "")] public class Cliente { [MessageBodyMember] public string[] Nombres; }`
c. `[DataContract] public class NombreClientes { [DataMember] public string[] NombreClientes; }`
d. `[DataContract] public class Cliente { [DataMember(IsRequired = false)] public Array[] NombreClientes; }`

14. Un operador puede registrar personas con sus datos personales: documento de identidad, nombres, apellidos, fecha de nacimiento, e información de contacto; como números telefónicos, direcciones de correo electrónico y direcciones físicas. Se requiere lo siguiente:

I. No puede registrarse 2 o más personas con el mismo documento de identidad.
II. La información documento de identidad, nombres, apellidos y fecha de nacimiento son obligatorios. Los nombres y apellidos aceptan caracteres del alfabeto latino y no pueden contener valores numéricos. El documento de identidad sólo acepta valores alfanuméricos.
III. Al menos debe registrar una información de contacto que puede ser dirección de correo electrónico o una dirección física. Máximo se pueden registrar 2 números telefónicos, 2 correos electrónicos y 2 direcciones físicas por persona.
IV. La solución debe realizarse exclusivamente en C#.

Escriba el código fuente para el requerimiento anterior. Presente su solución teniendo en cuenta las capas de aplicación y datos. Utilizar tecnologías .NET y SQL Server (C#, Ado, WCF. Etc.)

14. Escriba el código fuente en este espacio.