Comenzado el	martes, 30 de septiembre de 2025, 19:00
Estado	Finalizado
Finalizado en	martes, 30 de septiembre de 2025, 19:44
Tiempo empleado	43 minutos 10 segundos
Calificación	**8,67** de 10,00 (**86,67**%)

Pregunta 1

Correcta

Se puntúa 1,00 sobre 1,00

¿Cuál de las siguientes afirmaciones podríamos considerar como correctas?

Seleccione una:

a. Como testers, queremos siempre encontrar absolutamente todos los errores y defectos de un software

b. Los testers no necesitan comprender el negocio o los usuarios finales, solo deben enfocarse en probar el software

c. Como testers, queremos encontrar la mayor cantidad de errores con la menor cantidad de recursos posible ✔

d. Los testers no necesitan comunicarse con los desarrolladores o con otros miembros del equipo, ya que su trabajo es simplemente detectar errores en el software

Respuesta correcta

La respuesta correcta es: Como testers, queremos encontrar la mayor cantidad de errores con la menor cantidad de recursos posible

Pregunta 2

Correcta

Se puntúa 1,00 sobre 1,00

Para el siguiente algoritmo...

```
Algoritmo Testing
    definir num1, num2, num3 como entero;
    definir resultado como cadena;
    escribir "Ingresa tres números enteros:";
    leer num1, num2, num3;
    si (NO (num1 % 2 ≠ 0) Y NO (num2 % 3 ≠ 0)) entonces
        si (NO (num3 ≤ num1)) entonces
            resultado = "El 1er número es par, el 2do es divisible por 3, y el 3ro es mayor que el 1ro";
        sino
            resultado = "El 1er número es par, el 2do es divisible por 3, pero el 3ro no es mayor que el 1ro";
        finsi;
    sino
        si (NO (num1 % 5 ≠ 0) O NO (num2 % 2 ≠ 0)) entonces
            si (NO (num3 ≥ num2)) entonces
                resultado = "El 1er número es divisible por 5 o el 2do es par, y el 3ro es menor que el 2do";
            sino
                resultado = "El 1er número es divisible por 5 o el 2do es par,pero el 3ro no es menor que el 2do";
            finsi;
        sino
            resultado = "No se cumple ninguna condición";
        finsi;
    finsi;
    escribir "El resultado es: ", resultado;
FinAlgoritmo
```

¿Qué tipo de estrategia de prueba de Caja Blanca sería más adecuada para garantizar que cada combinación posible de resultados de las subexpresiones en una decisión compleja se evalúe al menos una vez, con el fin de detectar posibles errores en el manejo de cada ruta del flujo lógico?

Seleccione una:

- a. Cobertura de sentencia
- b. Cobertura de decisión
- c. Cobertura de condición ✔

Respuesta correcta

La respuesta correcta es: Cobertura de condición

Pregunta 3

Correcta

Se puntúa 1,00 sobre 1,00

Supongamos que tenemos una clase Usuario con los siguientes métodos:

- RegistrarUsuario(string nombre, string email): Registra un usuario si el nombre y el email son válidos.
- EsMayorDeEdad(int edad): Retorna un valor booleano indicando si el usuario es mayor de edad (18 años o más).
- CambiarEmail(string nuevoEmail): Cambia el email del usuario a uno nuevo si es válido.
- ObtenerNombre(): Retorna el nombre del usuario.
- ObtenerEmail(): Retorna el email del usuario.

A continuación, se presentan cinco pruebas unitarias. Cada una tiene un nombre genérico. Asociá cada prueba con uno de estos objetivos:

- Test1: Verifica que RegistrarUsuario() almacene correctamente el nombre del usuario.
- Test2: Verifica que CambiarEmail() actualice el email del usuario.
- Test3: Verifica que EsMayorDeEdad() retorne true si la edad es mayor a 18.
- Test4: Verifica que EsMayorDeEdad() retorne false si la edad es menor a 18.
- Test5: Verifica que RegistrarUsuario() lance una excepción si el email es inválido.

```
public class UsuarioTests
{
   private Usuario _usuario;
```

[SetUp] ✔

```
public void SetUp()
{
    _usuario = new Usuario();
}

[Test]
public void Test1()
{
   _usuario.RegistrarUsuario("Juan", "juan@correo.com");
    Assert.That(_usuario.ObtenerNombre(), Is.EqualTo("Juan"));
}

[Test]
public void Test2()
{
   _usuario.RegistrarUsuario("Ana", "ana@correo.com");
    _usuario.CambiarEmail("ana.nueva@correo.com");
    Assert.That(_usuario.ObtenerEmail(), Is.EqualTo("ana.nueva@correo.com"));
}

[Test]
public void Test3()
{
```

```
        bool resultado = _usuario.EsMayorDeEdad(22);
        Assert.That(resultado, Is.True);
    }

    [Test]
    public void Test4()
    {
        bool resultado = _usuario.EsMayorDeEdad(16);
        Assert.That(resultado, Is.False);
    }

    [Test]
    public void Test5()
    {
        Assert.Throws<ArgumentException>(() => _usuario.RegistrarUsuario("Pedro", "invalid-email"));
    }
}
```

```
Assert.That(ObtenerEmail(), Equal("ana.nueva@correo.com"));
```

```
bool resultado = _usuario.EsMenorDeEdad(16);
```

```
Assert.That(False);
```

Respuesta correcta

La respuesta correcta es:

Supongamos que tenemos una clase Usuario con los siguientes métodos:

- RegistrarUsuario(string nombre, string email): Registra un usuario si el nombre y el email son válidos.
- EsMayorDeEdad(int edad): Retorna un valor booleano indicando si el usuario es mayor de edad (18 años o más).
- CambiarEmail(string nuevoEmail): Cambia el email del usuario a uno nuevo si es válido.
- ObtenerNombre(): Retorna el nombre del usuario.
- ObtenerEmail(): Retorna el email del usuario.

A continuación, se presentan cinco pruebas unitarias. Cada una tiene un nombre genérico. Asociá cada prueba con uno de estos objetivos:

- Test1: Verifica que RegistrarUsuario() almacene correctamente el nombre del usuario.
- Test2: Verifica que CambiarEmail() actualice el email del usuario.
- Test3: Verifica que EsMayorDeEdad() retorne true si la edad es mayor a 18.
- Test4: Verifica que EsMayorDeEdad() retorne false si la edad es menor a 18.
- Test5: Verifica que RegistrarUsuario() lance una excepción si el email es inválido.

```
public class UsuarioTests
{
   private Usuario _usuario;

  [[SetUp]]
   public void SetUp()
   {
      [_usuario = new Usuario(); ]
   }

   [Test]
   public void Test1()
   {
```

```
    _usuario.RegistrarUsuario("Juan", "juan@correo.com");
    [Assert.That(_usuario.ObtenerNombre(), Is.EqualTo("Juan"));]
}

[Test]
public void Test2()
{
    _usuario.RegistrarUsuario("Ana", "ana@correo.com");
    [_usuario.CambiarEmail("ana.nueva@correo.com"); ]
    [ Assert.That(_usuario.ObtenerEmail(), Is.EqualTo("ana.nueva@correo.com"));]
}

[Test]
public void Test3()
{
    [bool resultado = _usuario.EsMayorDeEdad(22);]
    Assert.That(resultado, Is.True);
}

[Test]
public void Test4()
{
    [bool resultado = _usuario.EsMayorDeEdad(16);]
    [Assert.That(resultado, Is.False);]
```

```
    }

    [Test]
    public void Test5()
    {
        Assert.Throws<ArgumentException>(() => _usuario.RegistrarUsuario("Pedro", "invalid-email"));
    }
}
```

Pregunta 4

Correcta

Se puntúa 1,00 sobre 1,00

Queremos ver el resultado booleano de la siguiente expresión: “A es mayor que B y que C”. Vemos que la solución realizada es A>B>C. La solución correcta y el tipo de error que estamos viendo serían...

Seleccione una:

- a. (A>B) && (A>C). Estamos solucionando un error de cálculo
- b. (A>B) && (A>C). Estamos solucionando un error de comparación ✔
- c. (A>B) || (A>C). Estamos solucionando un error de comparación
- d. (A>B && >C). Estamos solucionando un error de entrada/salida

Respuesta correcta

La respuesta correcta es: (A>B) && (A>C). Estamos solucionando un error de comparación

Pregunta 5

Correcta

Se puntúa 1,00 sobre 1,00

Seleccionar la/s opción/es correcta/s sobre walkthough e inspecciones de código

- [x] a. Son mejoras a los procesos de pruebas de escritorios, ya que en ese método es un/a programador/a leyendo su propio código antes de probarlo. ✔
- [x] b. Un grupo de desarrolladores (entre 3 y 4 es un número óptimo) hacen la revisión. ✔
- [ ] c. Todos los participantes deben haber estado involucrados en el desarrollo
- [ ] d. Sólo un desarrollador se encarga de hacer la revisión
- [x] e. Sólo uno de los participantes es el autor del programa, por lo que la mayoría de los participantes de esta forma de testing no deben haber estado involucrados con el desarrollo. ✔

Respuesta correcta

Las respuestas correctas son: Un grupo de desarrolladores (entre 3 y 4 es un número óptimo) hacen la revisión. ,
Sólo uno de los participantes es el autor del programa, por lo que la mayoría de los participantes de esta forma de testing no deben haber estado involucrados con el desarrollo.,

Son mejoras a los procesos de pruebas de escritorios, ya que en ese método es un/a programador/a leyendo su propio código antes de probarlo.

Pregunta 6

Parcialmente correcta

Se puntúa 0,67 sobre 1,00

En las pruebas de Caja Blanca, indique cuáles de las siguientes afirmaciones son verdaderas (V) y cuáles son falsas (F)

- [ ] a. Permiten probar todos los caminos posibles dentro de un módulo.
- [x] b. Se basan en el conocimiento del código fuente del sistema ✔
- [ ] c. Su objetivo principal es evaluar la interfaz de usuario del software.
- [x] d. Son útiles para detectar errores de flujo de control, condiciones y bucles. ✔
- [ ] e. No requieren entender la lógica interna de los programas.

Respuesta parcialmente correcta.

Ha seleccionado correctamente 2.
Las respuestas correctas son:
Se basan en el conocimiento del código fuente del sistema,

Permiten probar todos los caminos posibles dentro de un módulo.,

Son útiles para detectar errores de flujo de control, condiciones y bucles.

Pregunta 7

Correcta

Se puntúa 1,00 sobre 1,00

Una prueba unitaria es...

Seleccione una:

a. Una pieza de código que invoca a una unidad de trabajo y controla un resultado final específico de una unidad de trabajo ✔

b. Asegurar que todas las funciones del sistema se integren sin errores

c. Verificar que el sistema funciona correctamente en su conjunto

Respuesta correcta

La respuesta correcta es: Una pieza de código que invoca a una unidad de trabajo y controla un resultado final específico de una unidad de trabajo

Pregunta 8

Incorrecta

Se puntúa 0,00 sobre 1,00

Marcar el error en el siguiente algoritmo, que debería detectar si un número es o no es primo:

```
1  Algoritmo EsPrimo
2      Definir numero, i, contador como Entero
3      Escribir "Ingrese un número:"
4      Leer numero
5      contador =  0
6      Para i = 1 Hasta numero Con Paso 1 Hacer
7          Si numero % i == 0 Entonces
8              contador = contador + 1
9          FinSi
10     FinPara
11     Si contador == 2 Entonces
12         Escribir "El número ", numero, " es primo."
13     Sino
14         Escribir "El número ", numero, " no es primo."
15     FinSi
16 FinAlgoritmo
```

Seleccione una:

- a. La condición del si de la línea 11 está mal implementada ✖
- b. El algoritmo no verifica si el usuario ingresa un número negativo o si ingresa 1
- c. El contador está mal, debería ser de 2 en 2
- d. El algoritmo no actualiza correctamente el contador en el bucle Para

Respuesta incorrecta.

La respuesta correcta es: El algoritmo no verifica si el usuario ingresa un número negativo o si ingresa 1

Pregunta 9

Correcta

Se puntúa 1,00 sobre 1,00

Supongamos que existe una clase CarritoDeCompras con los siguientes métodos ya implementados:

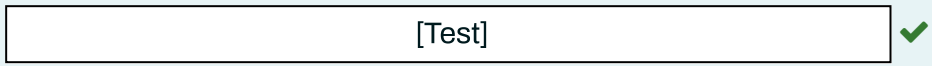
- AgregarProducto(string nombre, double precio) → Agrega un producto al carrito.
- ObtenerTotal() → Devuelve el total acumulado de precios en el carrito.
- ObtenerCantidadProductos() → Devuelve la cantidad total de productos agregados.
- Vaciar() → Elimina todos los productos del carrito y reinicia el total.

A continuación, se presentan tres pruebas unitarias. Cada una tiene un nombre genérico. Asociá cada prueba con uno de estos objetivos:

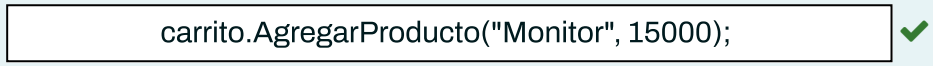
- Verificar que al agregar un producto, el total se actualiza correctamente.
- Verificar que al agregar varios productos, el total es la suma correcta.
- Verificar que al vaciar el carrito, el total y la cantidad de productos vuelven a cero.

```
public class CarritoDeComprasTests
{
  private CarritoDeCompras carrito;

  [SetUp]
  public void Init()
  {
    carrito = new CarritoDeCompras();
```

```
}

[Test]
public void Test1()
{
    carrito.AgregarProducto("Mouse", 1200);
    double total = carrito.ObtenerTotal();
    Assert.That(total, Is.EqualTo(1200));
}
```

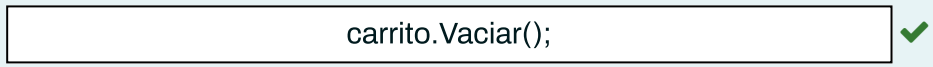



```
public void Test2()
{
    carrito.AgregarProducto("Teclado", 2500);
```




```
    double total = carrito.ObtenerTotal();
```


```
}

[Test]
public void Test3()
{
    carrito.AgregarProducto("Auriculares", 3000);
```



```
        Assert.That(carrito.ObtenerCantidadProductos(), Is.EqualTo(0));
        Assert.That(carrito.ObtenerTotal(), Is.EqualTo(0));
    }
}

}
```

```
Assert.That(carrito.ObtenerCantidadProductos() == (0));
```

```
Assert.That(total, Is.EqualTo(27500));
```

Respuesta correcta

La respuesta correcta es:

Supongamos que existe una clase CarritoDeCompras con los siguientes métodos ya implementados:

- AgregarProducto(string nombre, double precio) → Agrega un producto al carrito.
- ObtenerTotal() → Devuelve el total acumulado de precios en el carrito.
- ObtenerCantidadProductos() → Devuelve la cantidad total de productos agregados.
- Vaciar() → Elimina todos los productos del carrito y reinicia el total.

A continuación, se presentan tres pruebas unitarias. Cada una tiene un nombre genérico. Asociá cada prueba con uno de estos objetivos:

- Verificar que al agregar un producto, el total se actualiza correctamente.
- Verificar que al agregar varios productos, el total es la suma correcta.
- Verificar que al vaciar el carrito, el total y la cantidad de productos vuelven a cero.

```
public class CarritoDeComprasTests
{
  private CarritoDeCompras carrito;

  [[SetUp]]
  public void Init()
  {
    [carrito = new CarritoDeCompras();]
  }
```

```
[Test]
public void Test1()
{
    carrito.AgregarProducto("Mouse", 1200);
    double total = carrito.ObtenerTotal();
    Assert.That(total, Is.EqualTo(1200));
}

[ [Test]]
public void Test2()
{
    carrito.AgregarProducto("Teclado", 2500);
    [carrito.AgregarProducto("Monitor", 15000); ]
    double total = carrito.ObtenerTotal();
    [Assert.That(total, Is.EqualTo(17500));]
}

[Test]
public void Test3()
{
    carrito.AgregarProducto("Auriculares", 3000);
    [carrito.Vaciar();]
    [Assert.That(carrito.ObtenerCantidadProductos(), Is.EqualTo(0));]
    Assert.That(carrito.ObtenerTotal(), Is.EqualTo(0));
```

```
  }
}

}
```

Pregunta 10

Correcta

Se puntúa 1,00 sobre 1,00

¿Cuál de las siguientes afirmaciones describe mejor la técnica de caja negra?

Seleccione una:

- a. El tester tiene acceso al código fuente del software
- b. El tester conoce todos los detalles de diseño del software
- c. El tester examina el comportamiento del software sin conocer su implementación interna ✔
- d. El tester se enfoca únicamente en la estructura interna del software

Respuesta correcta

La respuesta correcta es: El tester examina el comportamiento del software sin conocer su implementación interna