En este caso como no existe código todavía no es posible realizar estas pruebas por lo que las síguete parte es una guía a cómo realizar las pruebas con ejemplos descriptivos

**PRUEBA DE CAJA BLANCA**

**Criterios de cobertura**

- Encontrar fragmentos del programa que no son ejecutados por los casos de prueba.

- Crear casos de prueba adicionales que incrementen la cobertura.

- Determinar un valor cuantitativo de la cobertura (que es, de manera indirecta, una medida de la calidad del programa).

**1. Cobertura de sentencias:**

Comprueba el número de sentencias ejecutables que se han ejecutado. Mantenimiento Avanzado de Sistemas de Información.

En este caso sería revisar paso a paso el código de las funciones de búsqueda

Mostrar información

Agregar al carro de compras

Remover del carro de compras

Completar compra.

**2. Cobertura de decisiones:** 

Comprueba el número de decisiones ejecutadas, considerando que se ha ejecutado una decisión cuando se han recorrido todas sus posible ramas (la que la hace true y la que la hace false, pero también todas las posibles ramas de un switch).

En este caso esta prueba se da por medio de decidir si realizar o no la compra e iniciar o no sesión.

**3. Cobertura de condiciones:**

Comprueba el número de condiciones ejecutadas, entendiendo que se ha ejecutado una condición cuando se han ejecutado todas sus posibles ramas.

Utilizando el diagrama de casos de uso es posible obtener esta información.

**4. Cobertura de condiciones múltiples:**

Comprueba el número de condiciones múltiples ejecutadas, considerando que se ha ejecutado una condición múltiple cuando se han ejecutado todas sus correspondientes ramas con todas las posibles variantes de la instrucción condicional.

Esta sección requiere recorrer los diagramas de flujo correspondientes al sistema.

**5. Cobertura de condiciones/decisiones:**

Comprueba el número de condiciones y decisiones que se han ejecutado.

Manejar una tabla de decisiones para cada función con sus respectivas salidas

**6. Cobertura de caminos:**

 Comprueba el número de condiciones y decisiones que se han ejecutado.

Hacer un árbol de toma de decisiones si es posible

**7. Cobertura de funciones:**

Comprueba el número de funciones y procedimientos que han sido llamados.

Manejar una tabla de funciones y procedimientos.

**8. Cobertura de llamadas:**

Comprueba el número de llamadas a funciones y procedimientos que se han ejecutado.

Manejar una tabla de llamada de funciones y procedimientos.

**9. Cubrimiento de bucles:**

Comprueba el número de bucles que han sido ejecutados cero veces (excepto para bucles do -While), una vez y más de una vez.

Revisar cada ciclo y catalogar cuantas veces es llamado cada uno.

**10. Cubrimiento de carrera:**

Comprueba el número de tareas o hilos que han ejecutado simultáneamente el mismo bloque de código.

**11. Cobertura de operadores relacionales:**

Comprueba si se han ejecutado los valores límite en los operadores relacionales (>, <, >=, <=), ya que se asume la hipótesis de que estas situaciones son propensas a errores.

**12. Cobertura de tablas:**

Comprueba si se ha hecho referencia a todos los elementos de los arrays.

Trabajar con un array de ejemplo para revisar si se utilizan solos los elementos del mismo

**PRUEBA DE CAJA NEGRA**

**1. Particiones de Equivalencia:**

La partición de equivalencia es un método de prueba de Caja Negra que divide el campo de entrada de un programa en clases de datos de los que se pueden derivar casos de prueba.   
1. Identificar las clases de equivalencia.

2. Identificar los casos de prueba.

**2.Análisis de Valores Límite:**

Las condicione límite son aquellas que se hayan en los márgenes de la clase de equivalencia, tanto de entrada como de salida. Por ello, se ha desarrollado el análisis de valores límite como técnica de prueba. Esta técnica nos lleva a elegir los casos de prueba que ejerciten los valores límite.

**3. Métodos Basados en Grafos.**

**4. Pruebas de Comparación.**

**5.Análisis Causa-Efecto.**

- si un parámetro de entrada debe estar comprendido en un cierto rango, aparecen 3 clases de equivalencia: por debajo, en y por encima del rango.

- si una entrada requiere un valor concreto, aparecen 3 clases de equivalencia: por debajo, en y por encima del rango.

- si una entrada requiere un valor de entre los de un conjunto, aparecen 2 clases de equivalencia: en el conjunto o fuera de él.

- si una entrada es booleana, hay 2 clases: si o no. Los mismos criterios se aplican a las salidas esperadas: hay que intentar generar resultados en todas y cada una de las clases.