Pregunta 4 (6 puntos) (a)

(a) ¿Qué son las pruebas efectivas y sistemáticas? Referencia: **https://www.effective-software-testing.com/effective-and-systematic (1 punto)**

Las pruebas efectivas y sistemáticas son un enfoque disciplinado y estructurado para diseñar y ejecutar pruebas de software. Se basan en comprender los requisitos, diseñar casos de prueba exhaustivos y estructurados, utilizar pruebas basadas en especificaciones y propiedades, considerar las dependencias y emplear técnicas de simulación. El objetivo es mejorar la calidad del software al garantizar pruebas completas y significativas. Este enfoque ayuda a identificar y corregir errores de manera más eficiente, evitando la pérdida de tiempo y recursos en pruebas innecesarias. Además, proporciona una base sólida para el diseño y mantenimiento del conjunto de pruebas. Al seguir este enfoque, los ingenieros de pruebas pueden obtener resultados más confiables y contribuir a un software más robusto y confiable.

(b) Pruebas (2 puntos)

Dada esta especificación:

Dada esta especificación:

```
/**
* Dividir una cadena en un carácter delimitador.
*
* @param texto          una cadena
* @param delimitador          un delimitador por el cual dividir la cadena
* @param límite               un límite superior en el número de elementos a devolver:
                        si límite < 0, no hay límite superior; límite != 0
* @retorna  una  lista de cadenas [s1, s2, ..., sN] tales que:
*          - texto = s1 + delimitación + s2 + delimitación + ... + delimitación + sN
*           - N <= límite si límite > 0
*          - none de s1, s2, ..., sN contiene delimitador
* @throws IllegalArgumentException si límite > 0
*
*          y hay más de limite-1 ocurrencias de delimitador en el texto.
* /

public static List<String> split(String text, char delim, int limit);
```

(a) Comienza a implementar una estrategia de prueba sistemática para esta función escribiendo una buena partición del espacio de entrada solo en el límite de entrada, es decir, la partición no debe mencionar ni el texto ni el delimitador.

a) Límite < 0: Prueba con un límite negativo, lo que significa que no hay un límite superior en el número de elementos a devolver. Por ejemplo, limit = -1.
b) Límite = 0: Prueba con un límite igual a cero, lo que implica que no se deben devolver elementos. Por ejemplo, limit = 0.
c) Límite > 0: Prueba con un límite positivo para limitar el número máximo de elementos a devolver. Por ejemplo, limit = 3.

(b) Ahora, escriba una buena partición del espacio de entrada sobre la relación entre el límite y las ocurrencias del delimitador en el texto. Tu partición debe mencionar las tres entradas.
    a) Sin ocurrencias del delimitador en el texto: Prueba con un texto que no contiene el delimitador. Por ejemplo, texto = "Hola Mundo" y delimitador = ','.
    b) Menos de (límite-1) ocurrencias del delimitador en el texto: Prueba con un texto que contiene menos ocurrencias del delimitador que el límite especificado menos uno. Por ejemplo, texto = "Uno-Dos-Tres-Cuatro-Cinco", delimitador = '-' y límite = 3.
    c) Igual o más de (límite-1) ocurrencias del delimitador en el texto: Prueba con un texto que contiene igual o más ocurrencias del delimitador que el límite especificado menos uno. Esto debería generar una excepción IllegalArgumentException. Por ejemplo, texto = "Uno-Dos-Tres-Cuatro-Cinco-Seis", delimitador = '-' y límite = 4.