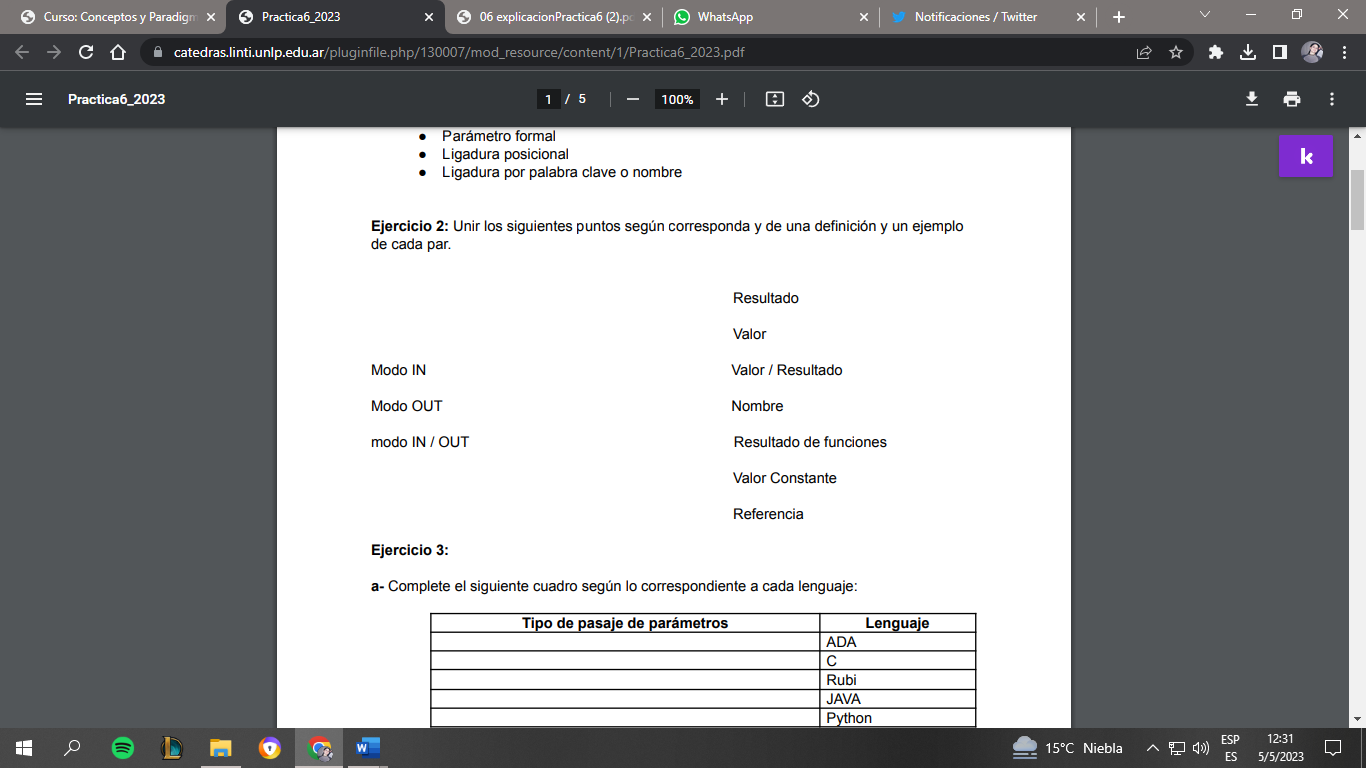
**Práctica 6**

**Ejercicio 1:**

**a- Explique brevemente los siguientes conceptos**

* **Parámetro:** es una forma de compartir datos entre diferentes unidades. Es la más flexible y permite la transferencia de diferentes datos en cada llamada. Proporciona ventajas en legibilidad y modificabilidad. Nos permiten compartir los datos en forma abstracta ya que indican con precisión qué es exactamente lo que se comparte
* **Parámetro real:** : es un valor u otra entidad utilizada para pasar a un procedimiento o función. Están en la parte de la invocación
* **Parámetro formal:** es una variable utilizada para recibir valores de entrada en una rutina, subrutina etc. Se ponen en la parte de la declaración. Es una variable local a su entorno.
* **Ligadura posicional:** los parámetros formales y reales se ligan según la posición en la llamada y en la declaración.
* **Ligadura por palabra clave o nombre:** los parámetros formales y reales se ligan por el nombre. Se debe conocer los nombres de los parámetros formales.

**Ejercicio 2: Unir los siguientes puntos según corresponda y de una definición y un ejemplo de cada par.**



**Ejercicio 3:**

1. **Complete el siguiente cuadro según lo correspondiente a cada lenguaje:**

|  |  |
| --- | --- |
| **Tipo de pasaje de parámetros** | **Lenguaje** |
| * Por defecto con copia IN * Por resultado OUT * IN OUT   + Para los tipos primitivos indica por valor-resultado   + Para los tipos no primitivos, datos compuestos (arreglo , registro) se hace por referencia. | ADA |
| * Por valor (Si se necesita por referencia se usa punteros) * Permite pasaje por valor constante, agregando const | C |
| * Por valor, pero si se pasa un objeto "mutable", no se hace una copia sino que se trabaja sobre él. | Ruby |
| * El único mecanismo contemplado es el paso por copia de valor. Pero como las variables de tipo no primitivos son todas referencias a variables anónimas en la heap, el paso por valor de una de estas variables son en realidad un paso por referencia de las variables | JAVA |
| * Se puede pasar de dos formas:   + Inmutables: actuara como por valor   + Mutables: No se hace una copia sino que se trabaja sobre él. | Python |

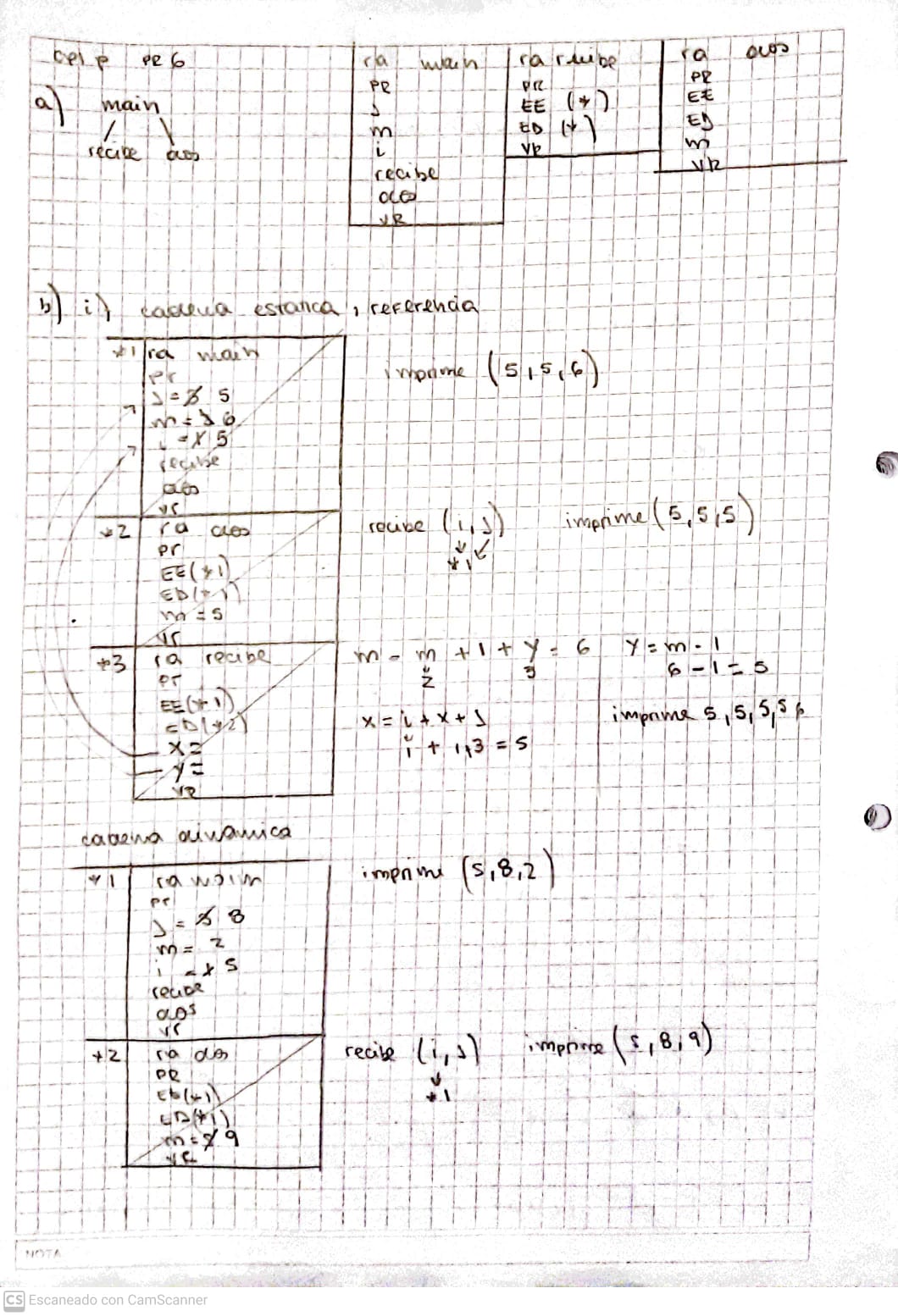
1. **Ada es más seguro que Pascal, respecto al pasaje de parámetros en las funciones. Explique por qué.**
2. **Explique cómo maneja Ada los tipos de parámetros in-out de acuerdo al tipo de dato**

**Ejercicio 4: Sea el siguiente programa escrito en Pascal-like**

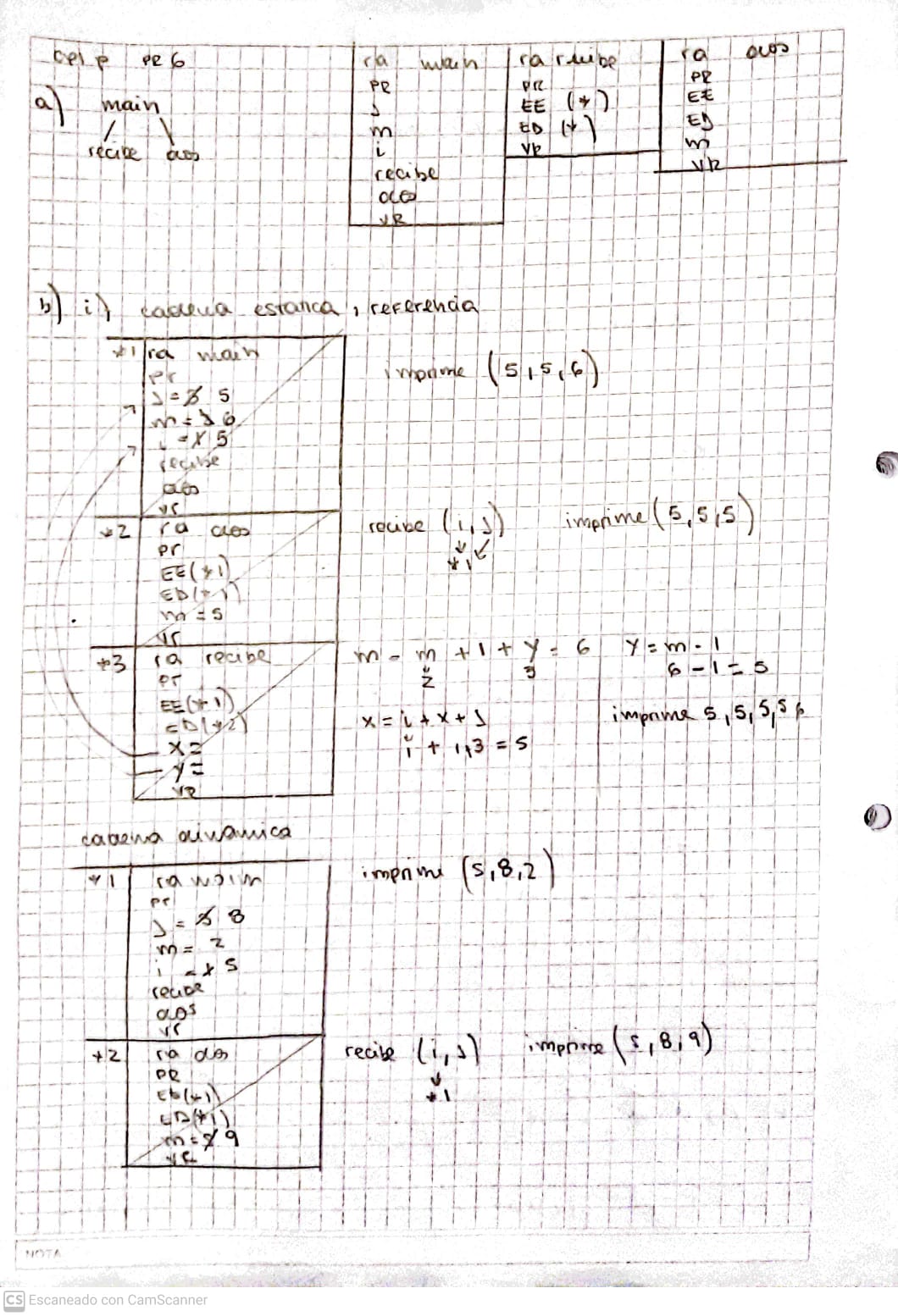
Captura de pantalla de computadora

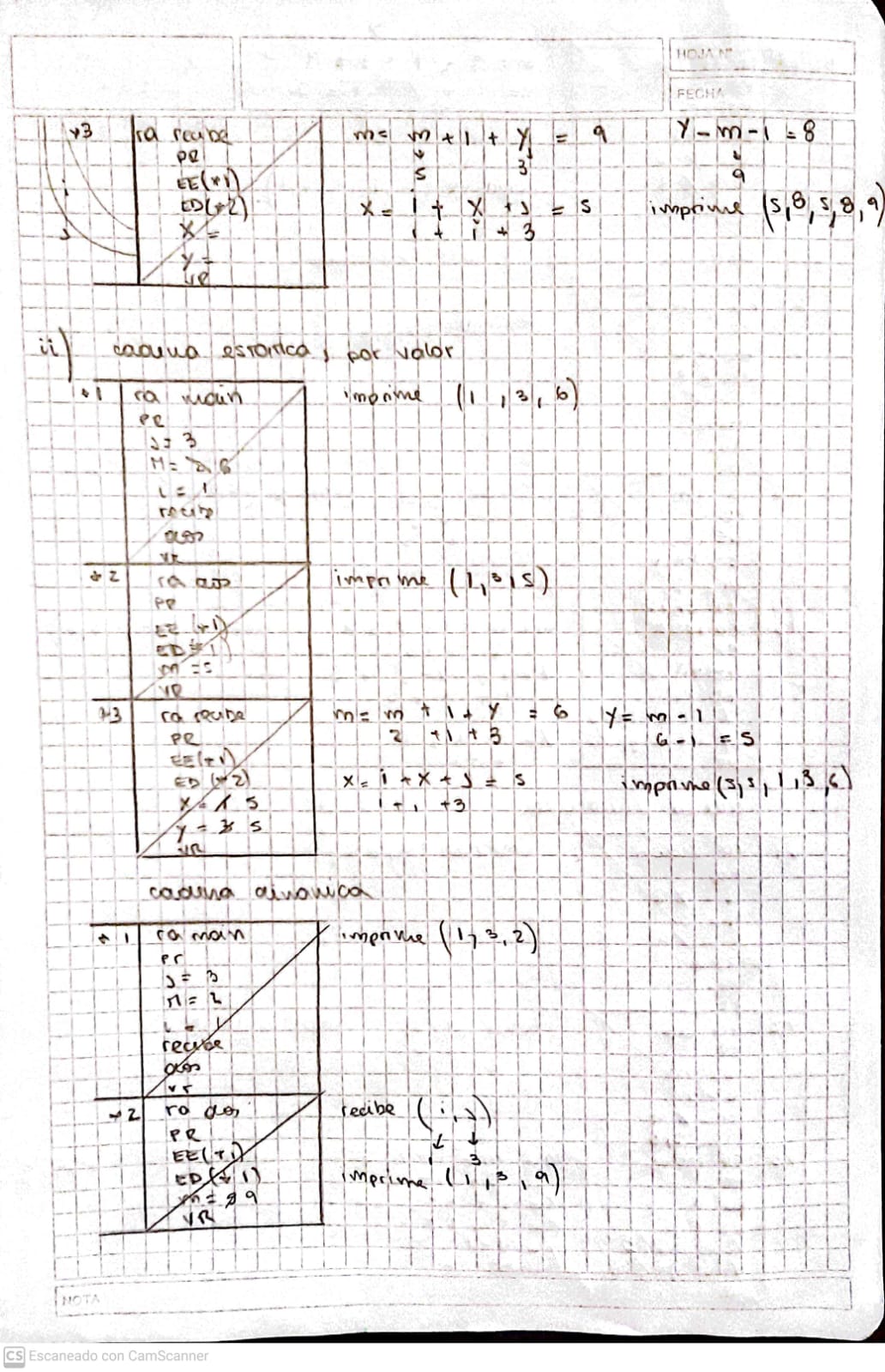
Descripción generada automáticamente

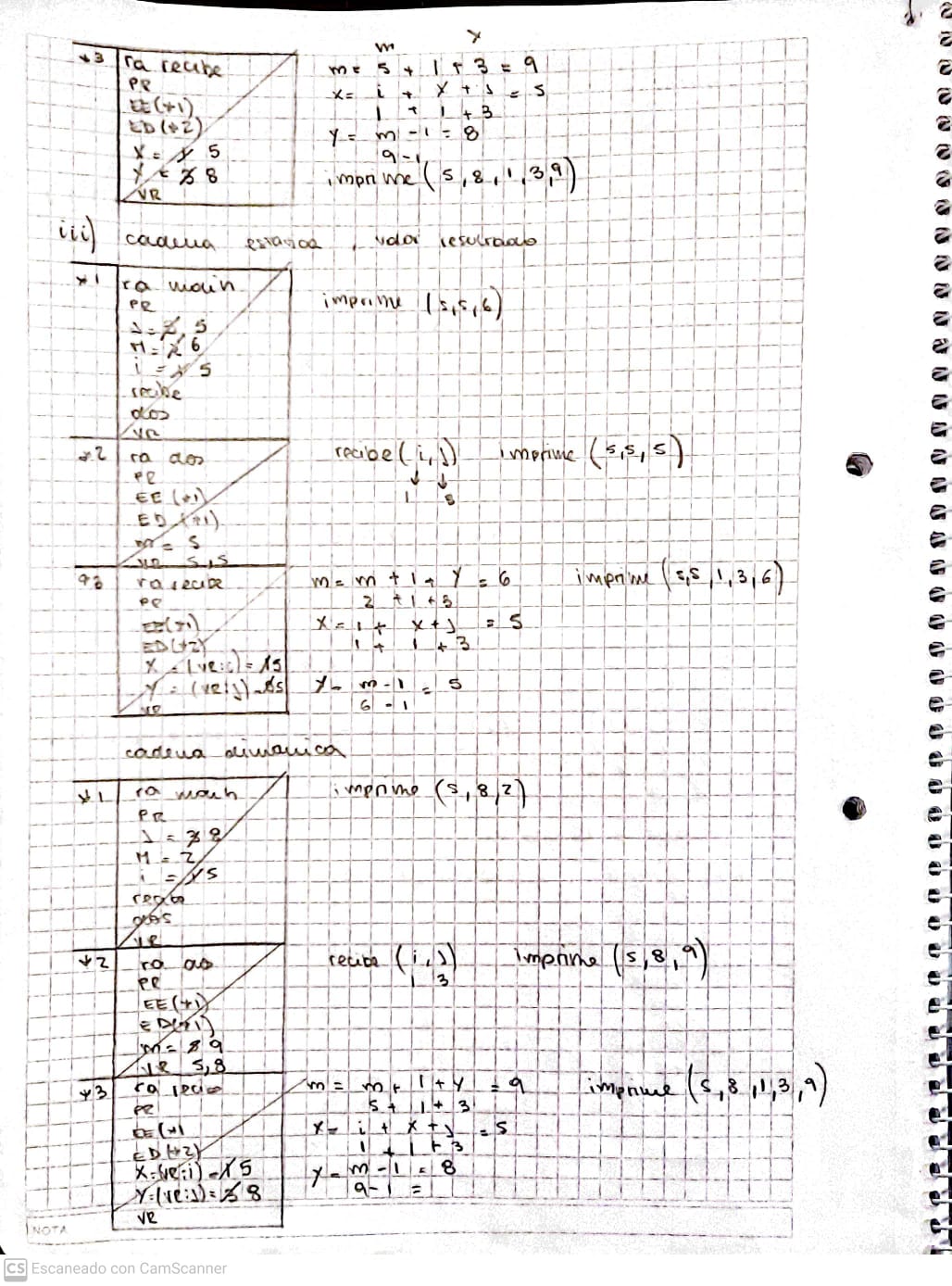
1. **Arme el árbol de anidamiento sintáctico y el registro de activación de cada una de las unidades.**

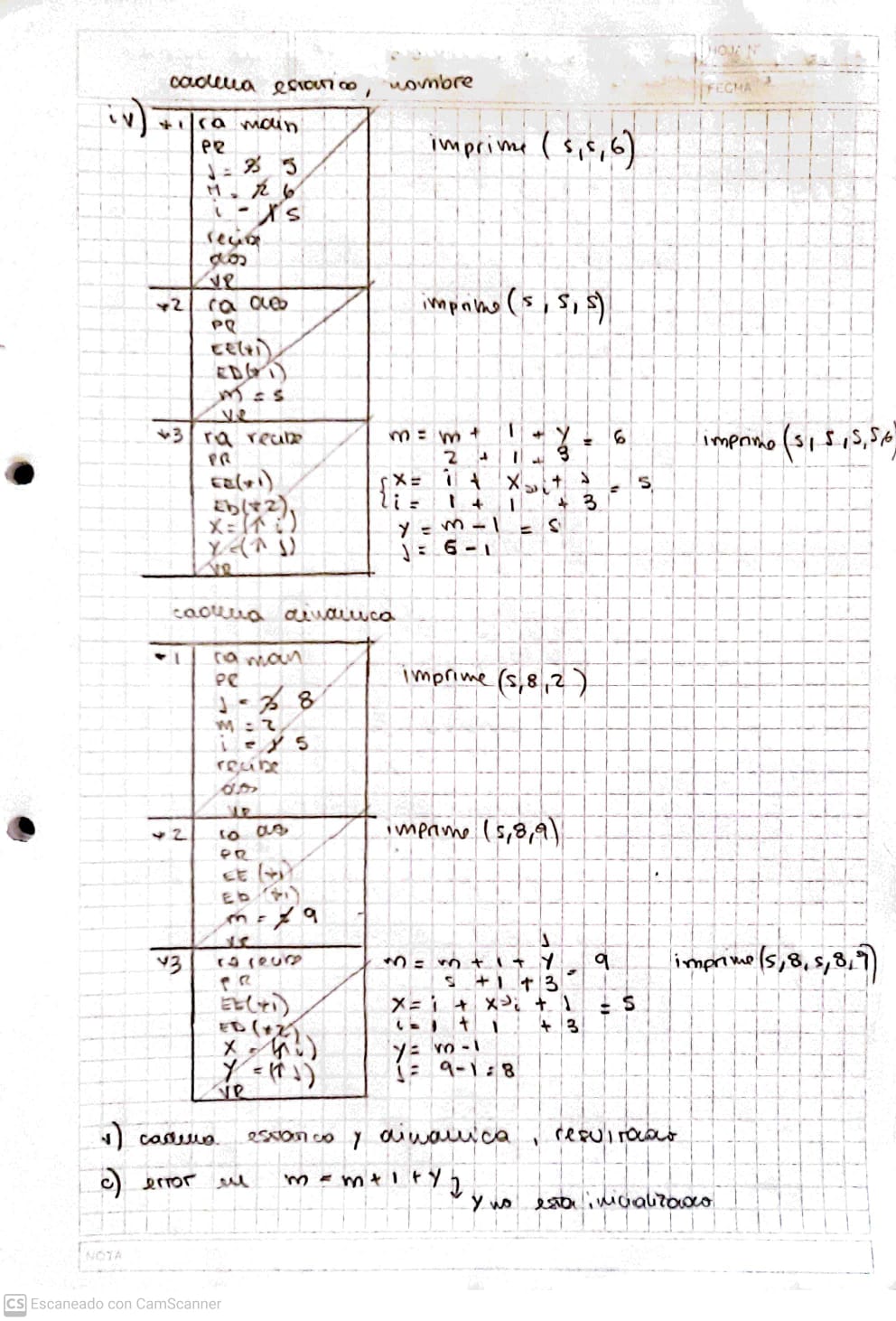
****

1. **Decir qué imprime el programa suponiendo que para todas las variables que se pasan el pasaje de parámetros es por: (Deberá hacer la pila estática y dinámica para cada caso)**
2. **Referencia.**
3. **Valor**
4. **Valor Resultado**
5. **Nombre**
6. **Resultado.**

****

****

****

****

1. **¿Existió algún caso que no pudo realizarlo porque saltó algún tipo de error? Diga cuál y por qué.**

Si, en la cadena estática y dinámica resultado hay error en m = m + 1 + y ya que y no esta inicializada

1. **¿Dará el mismo resultado si se trata de un lenguaje que sigue la cadena dinámica? Justifique la respuesta realizando las pilas de activación**

No, no será el mismo resultado, se puede observar en el punto b como no tienen los mismos resultados e impresiones.