|  | **Carátula para entrega de prácticas** | |
| --- | --- | --- |
| Facultad de Ingeniería | | Laboratorio de docencia |

Laboratorios de computación

salas A y B

| *Profesor(a):* | Ing. Karina García Morales |
| --- | --- |
| *Asignatura:* | Fundamentos de Programación (L) |
| *Grupo:* | 22 |
| *No. de práctica(s):* | Práctica 3. Solución de Problemas y Algoritmos. |
| *Integrante(s):* | González Márquez José Luis |
| *No. de lista o brigada:* | 20 |
| *Semestre:* | 2026-1 |
| *Fecha de entrega:* | Martes 23 de Septiembre de 2025 |
| *Observaciones:* |  |
|  |  |

CALIFICACIÓN: \_\_\_\_\_\_\_\_\_\_

**Objetivo**

El alumno elaborará algoritmos correctos y eficientes en la solución de problemas siguiendo las etapas de Análisis y Diseño pertenecientes al Ciclo de vida del software.

**Desarrollo**

Un problema informático se puede definir como el conjunto de instancias al cual corresponde un conjunto de soluciones, junto con una relación que asocia para cada instancia del problema un subconjunto de soluciones (posiblemente vacío).

Para poder solucionar un problema nos apoyamos en la Ingeniería de Software que de acuerdo con la IEEE se define como “La aplicación de un enfoque sistemático, disciplinado y cuantificable hacia el desarrollo, operación y mantenimiento del software". Por lo que el uso y establecimiento de principios de ingeniería sólidos, son básicos para obtener un software que sea económicamente fiable y funcione eficientemente.

La Ingeniería de Software provee métodos que indican cómo generar software. Estos métodos abarcan

una amplia gama de tareas:

• Planeación y estimación del proyecto.

• Análisis de requerimientos del sistema y software.

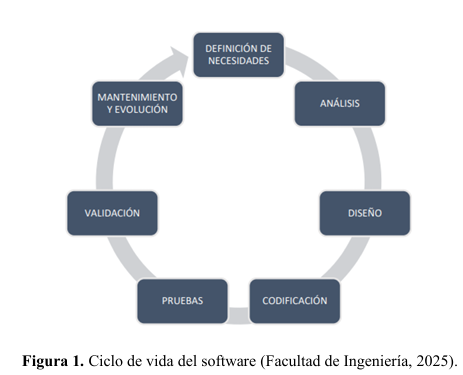
• Diseño de la estructura de datos, la arquitectura del programa y el procedimiento algorítmico.

• Codificación.

• Pruebas y mantenimiento (validación y verificación).

**Ciclo de vida del software**

La ISO (International Organization for Standarization) en su norma 12207 define al ciclo de vida de un software como: “Un marco de referencia que contiene las actividades y las tareas involucradas en el desarrollo,la explotación y el mantenimiento de un producto de software, abarcando desde la definición hasta la finalización de su uso” La Figura 1 muestra las actividades, también nombradas etapas, que se realizan en el ciclo de vida del software.



**Solución de problemas:**

Dentro del ciclo de vida del software, en el análisis se busca comprender la necesidad, es decir, entender el problema.

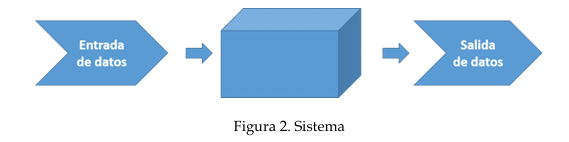
El análisis es el proceso para averiguar qué es lo que requiere el usuario del sistema de software (análisis de requisitos). Esta etapa permite definir las necesidades de forma clara y concisa (especificación de requisitos).

Por lo tanto, la etapa del análisis consiste en conocer qué es lo que está solicitando el usuario. Para ello es importante identificar dos grandes conjuntos dentro del sistema: el conjunto de entrada y el conjunto de salida. En la Figura 2 se muestra de manera esquemática los componentes del sistema.

El conjunto de entrada está compuesto por todos aquellos datos que pueden alimentar al sistema.

El conjunto de salida está compuesto por todos los datos que el sistema regresará como resultado del proceso. Estos datos se obtienen a partir de los datos de entrada.

La unión del conjunto de entrada y el conjunto de salida forman lo que se conoce como el dominio del problema, es decir, los valores que el problema puede manejar.



La etapa de análisis es crucial para la creación de un software de calidad, ya que si no se entiende qué es lo que se desea realizar, no se puede generar una solución. Sin embargo, es común caer en ambigüedades debido al mal entendimiento de los requerimientos iniciales.

***Algoritmos***

Una vez realizado el análisis, es decir, ya que se entendió qué es lo que está solicitando el usuario y ya identificado el conjunto de entrada y el conjunto de salida, se puede proceder al diseño de la solución, esto es, a la generación del algoritmo. Dentro del ciclo de vida del software, la creación de un algoritmo se encuentra en la etapa de diseño. Ver figura 1.

Durante el diseño se busca proponer una o varias alternativas viables para dar solución al problema y con base en esto tomar la mejor decisión para iniciar la construcción.

Un problema matemático es computable si éste puede ser resuelto, en principio, por un dispositivo computacional.

La teoría de la computabilidad es la parte de la computación que estudia los problemas de decisión que pueden ser resueltos con un algoritmo.

Un algoritmo se define como un conjunto de reglas, expresadas en un lenguaje específico, para realizar alguna tarea en general, es decir, un conjunto de pasos, procedimientos o acciones que permiten alcanzar un resultado o resolver un problema.

Estas reglas o pasos pueden ser aplicados un número ilimitado de veces sobre una situación particular.

Un algoritmo es la parte más importante y durable de las ciencias de la computación debido a que éste puede ser creado de manera independiente tanto del lenguaje como de las características físicas del equipo que lo va a ejecutar.

Las principales características con las que debe cumplir un algoritmo son:

• Preciso: Debe indicar el orden de realización de paso y no puede tener ambigüedad

• Definido: Si se sigue dos veces o más se obtiene el mismo resultado.

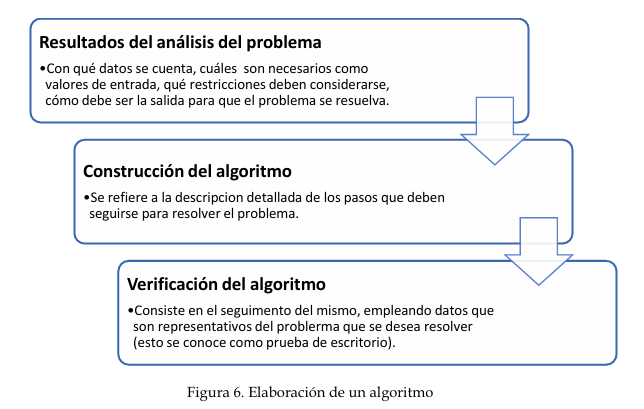
• Finito: Tiene fin, es decir tiene un número determinado de pasos.

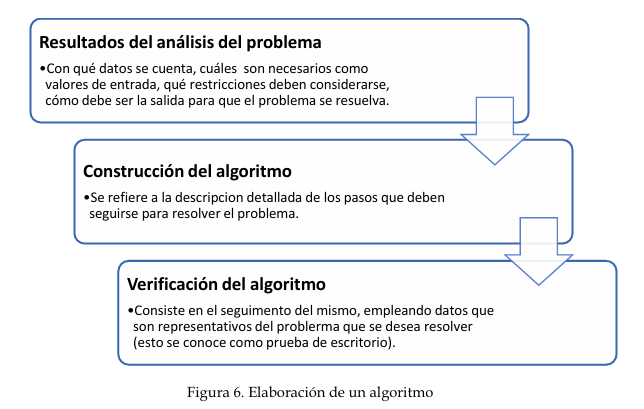
• Correcto: Cumplir con el objetivo.

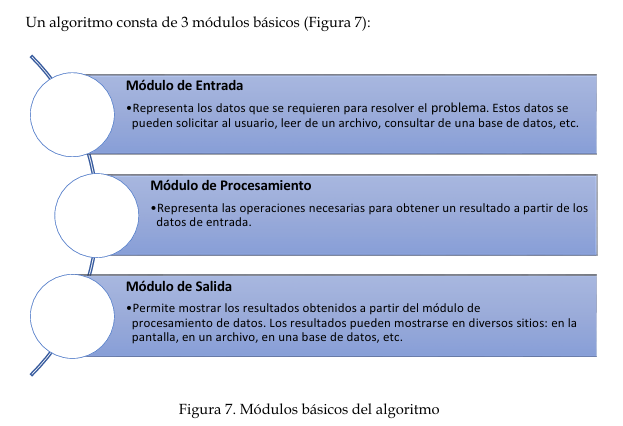
• Debe tener al menos una salida y ésta debe de ser perceptible

• Debe ser sencillo y legible

• Eficiente: Realizarlo en el menor tiempo posible

• Eficaz: Que produzca el efecto esperado

Por tanto, un buen algoritmo debe ser correcto (cumplir con el objetivo) y eficiente (realizarlo en el menor tiempo posible), además de ser entendible para cualquier persona. Las actividades a realizar en la elaboración de un algoritmo para obtener una solución a un problema de forma correcta y eficiente se muestran en la Figura 6. 



**Variables**

Un algoritmo requiere del uso de variables porque guardan el valor (numérico o no numérico) de los datos de entrada; también suelen ser utilizadas para almacenar datos generados en el proceso y datos de salida. Es a través del valor de dichas variables que el algoritmo puede fluir en la secuencia de pasos a seguir.

**Prueba de escritorio**

Para validar que la creación de un algoritmo para la solución de un problema es la correcta, se suele realizar mediante una prueba de escritorio. Una prueba de escritorio es una matriz formada por los valores que van adquiriendo cada una de las variables del algoritmo en cada iteración. Una iteración es el número de veces que se ejecuta una parte del algoritmo y permite ver los valores que van adquiriendo las variables en cada repetición. Para el ejemplo en cuestión, la prueba de escritorio quedaría de la siguiente manera (considerando a X como la variable que almacena el número solicitado)

***Ejercicio 1***

PROBLEMA: Seguir el algoritmo para obtener una figura

ENTRADA: Hoja tamaño carta en limpio, regla y lápiz.

SALIDA: Figura correcta.

Algoritmo

1. Dibuja una V invertida. Empieza desde el lado izquierdo, sube, y baja hacia el

lado derecho, no levantes el lápiz.

2. Ahora dibuja una línea en ángulo ascendente hacia la izquierda. Debe cruzar la

primera línea más o menos a 1/3 de la altura. Todavía no levantes el lápiz del

papel.

3. Ahora, dibuja una línea horizontal hacia la derecha. Debe cruzar la V invertida

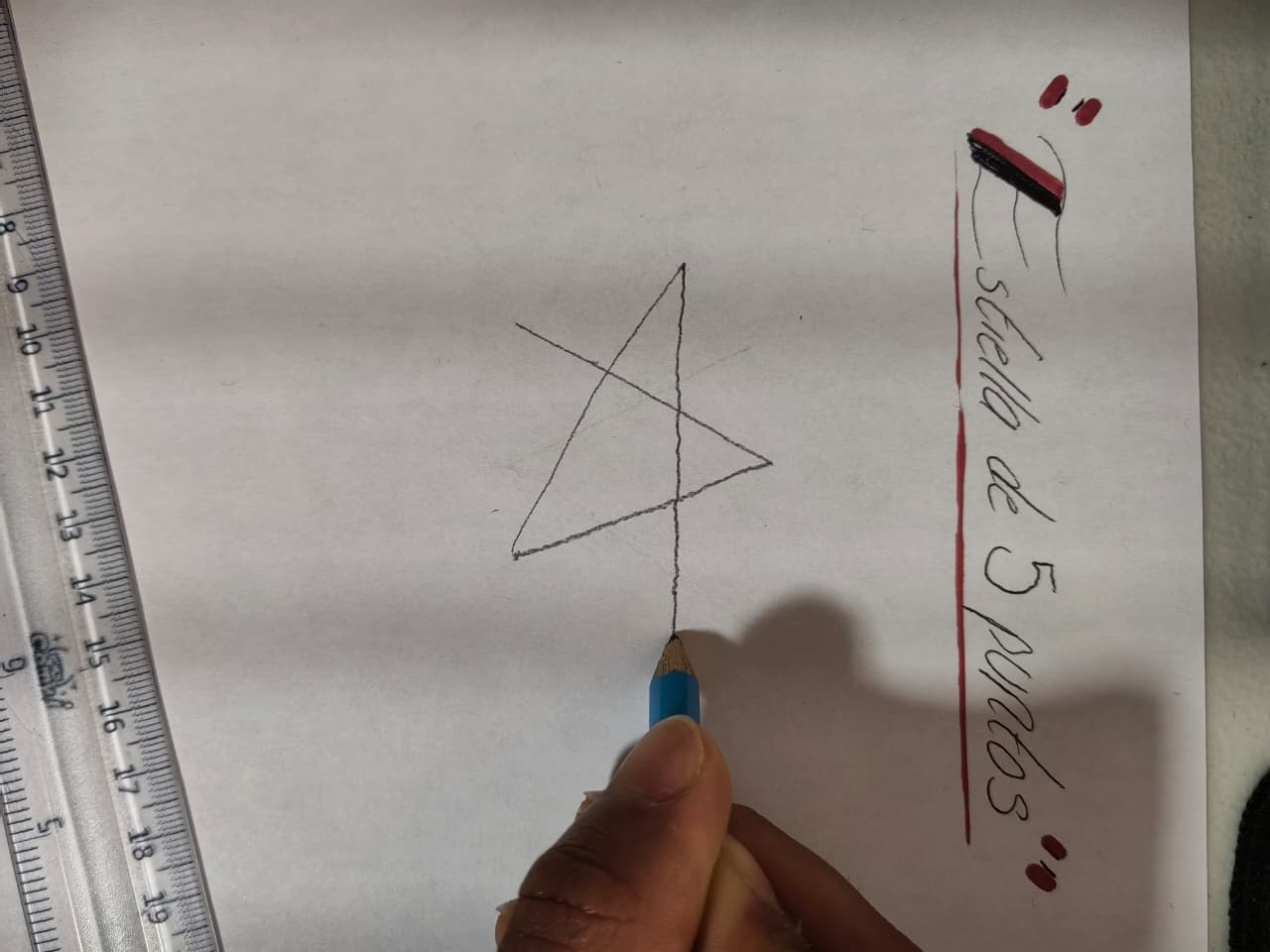
más o menos a 2/3 de la altura total. Sigue sin levantar el lápiz.

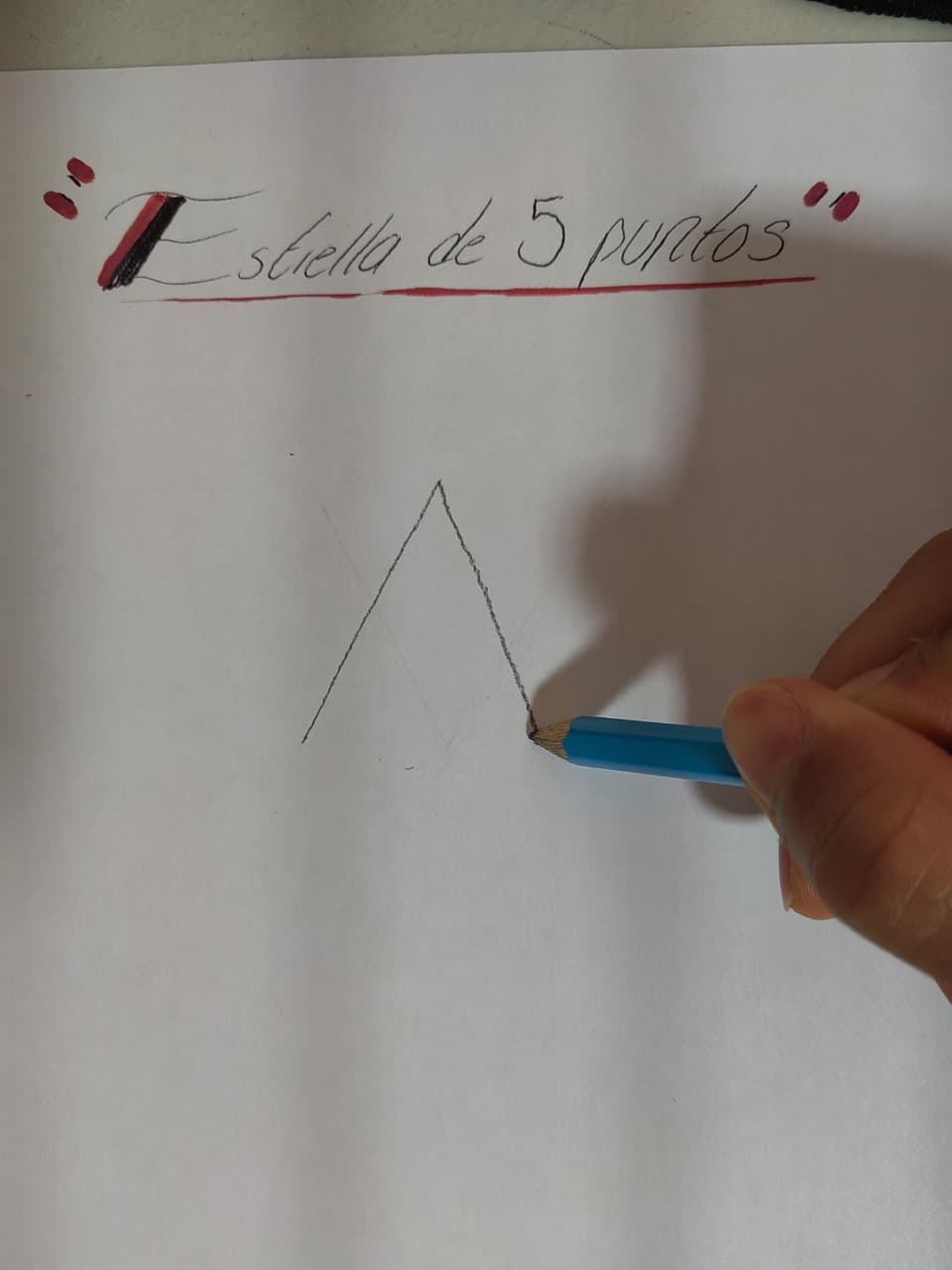
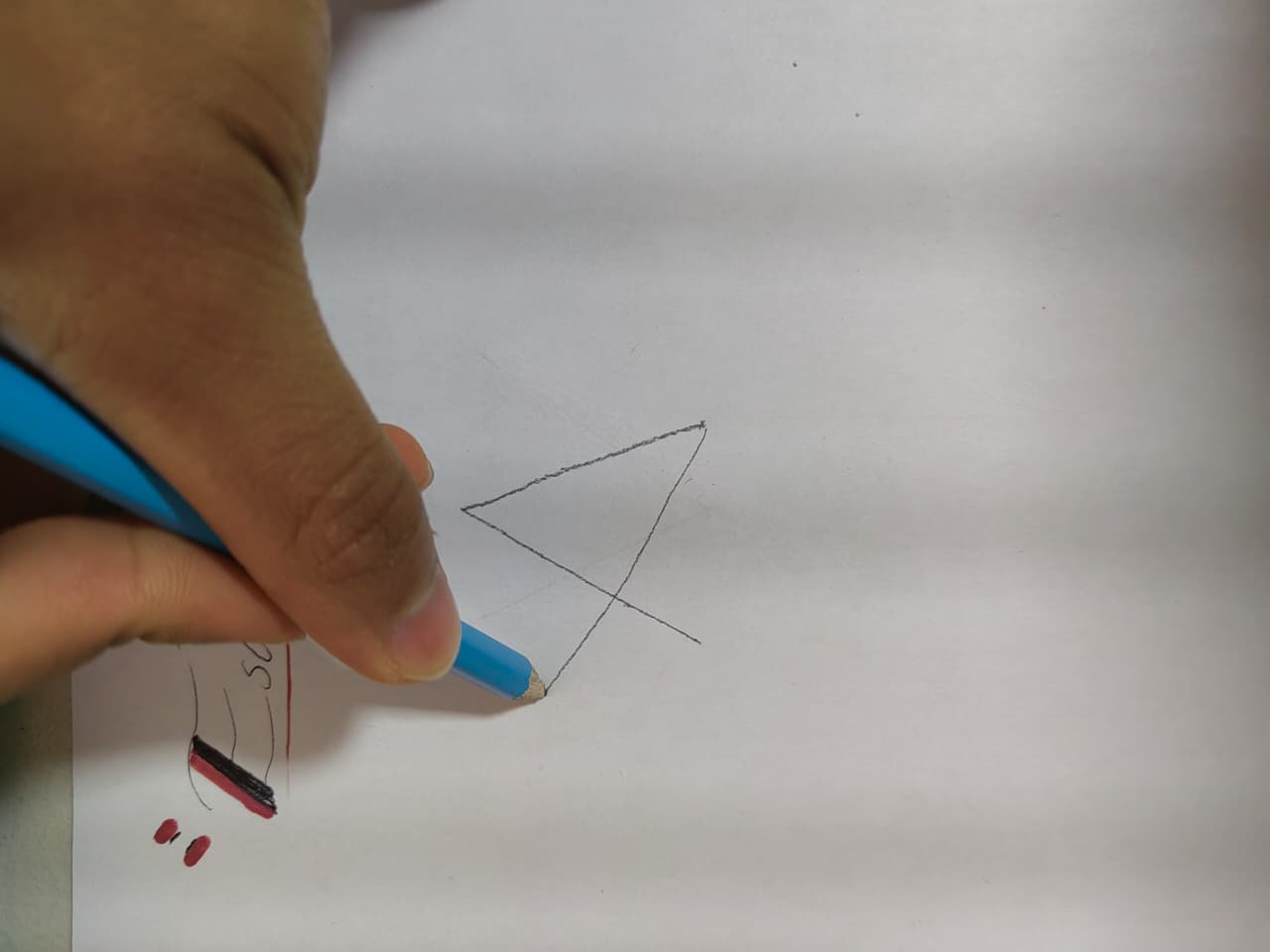
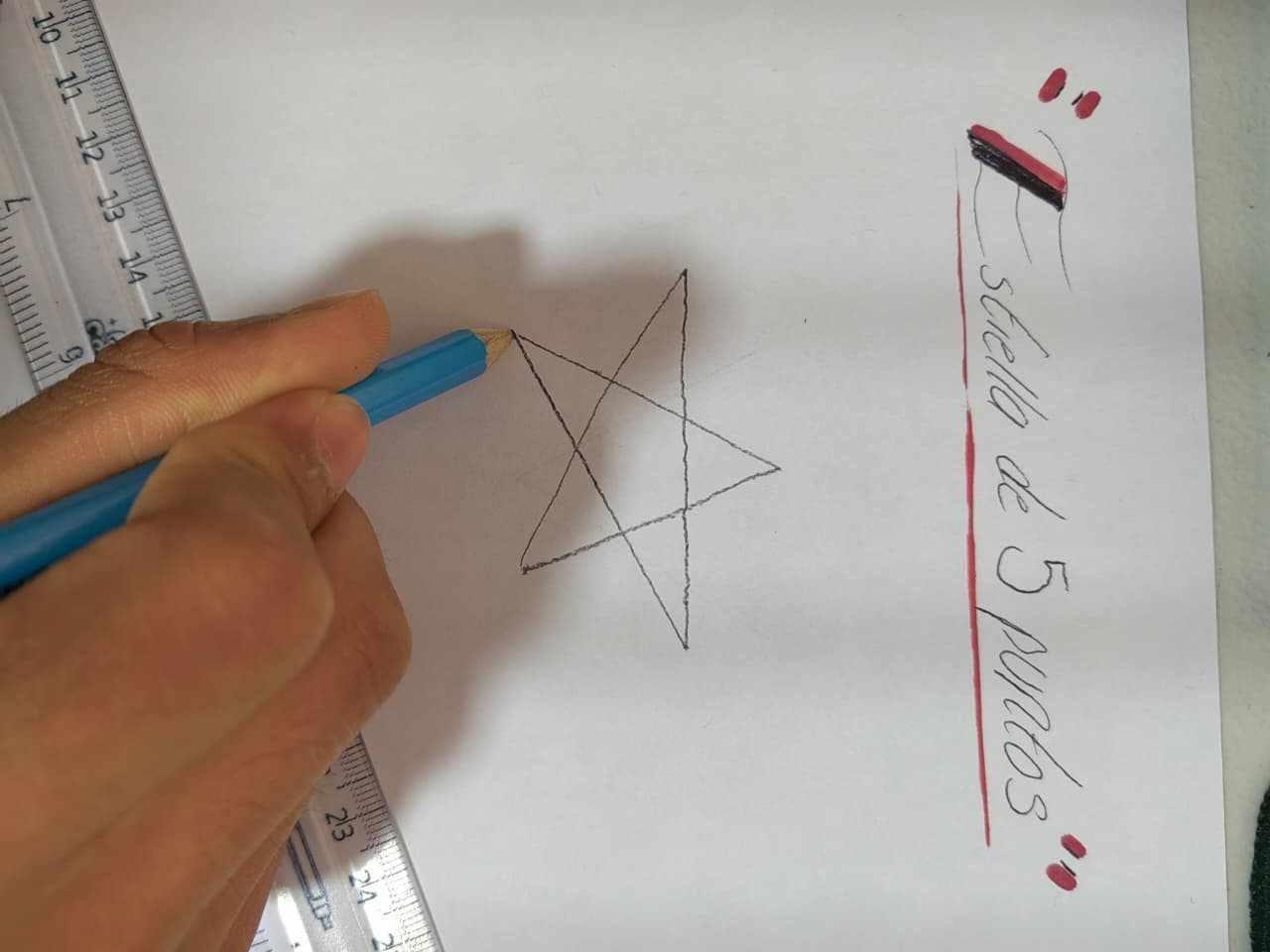
4. Dibuja una línea en un ángulo descendente hasta el punto de inicio. Las líneas

deben unirse.

5. Ahora ya puedes levantar el lápiz del papel. Has terminado la estrella de 5

puntas.





***Ejercicio 2***

PROBLEMA: Seguir el algoritmo para obtener una figura

ENTRADA: Hoja tamaño carta en limpio, regla y lápiz.

SALIDA: Figura correcta.

Algoritmo

1. Empieza dibujando un círculo con un compás. Coloca un lápiz en el compás.

Coloca la punta del compás en el centro de una hoja de papel.

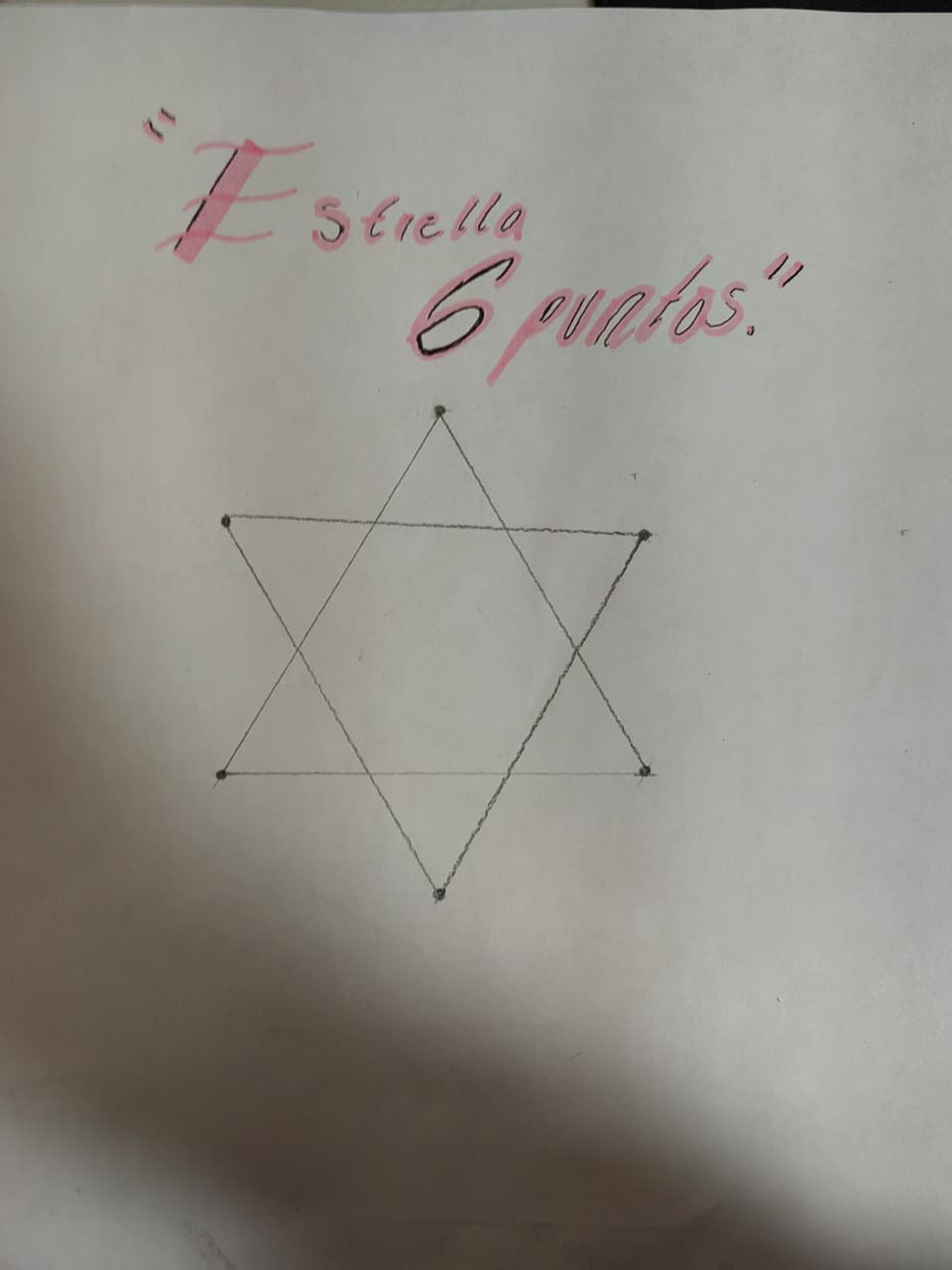
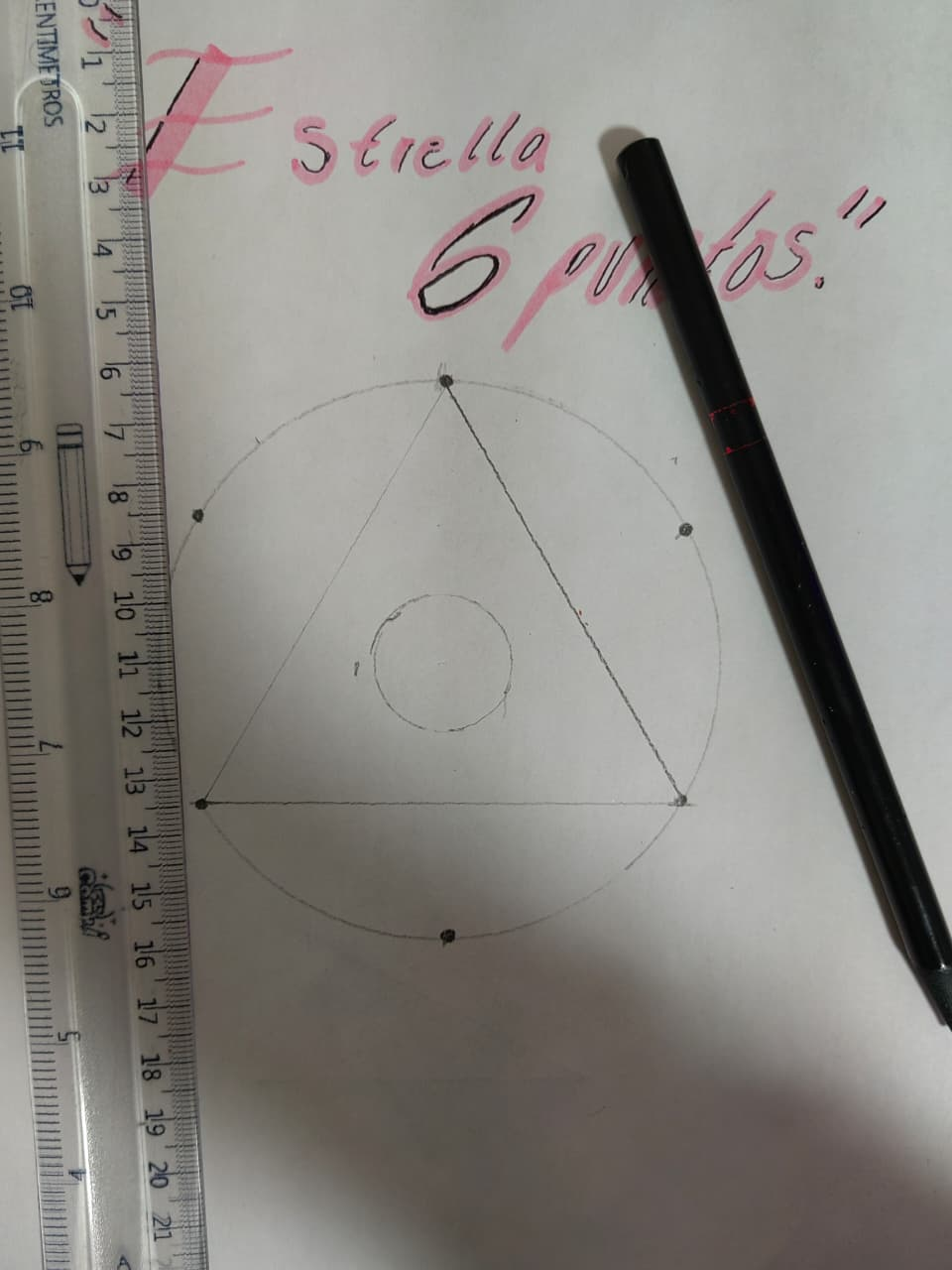
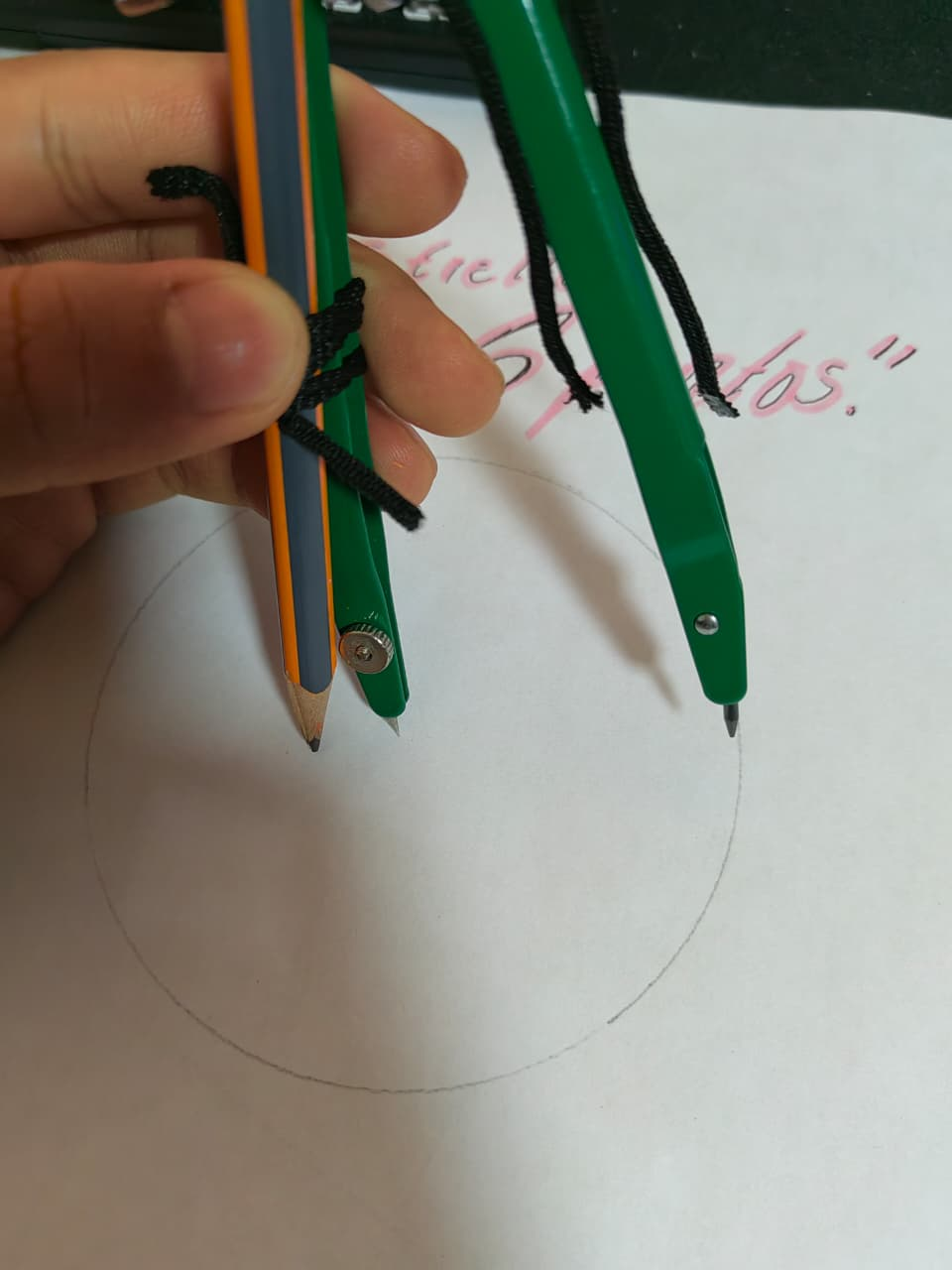
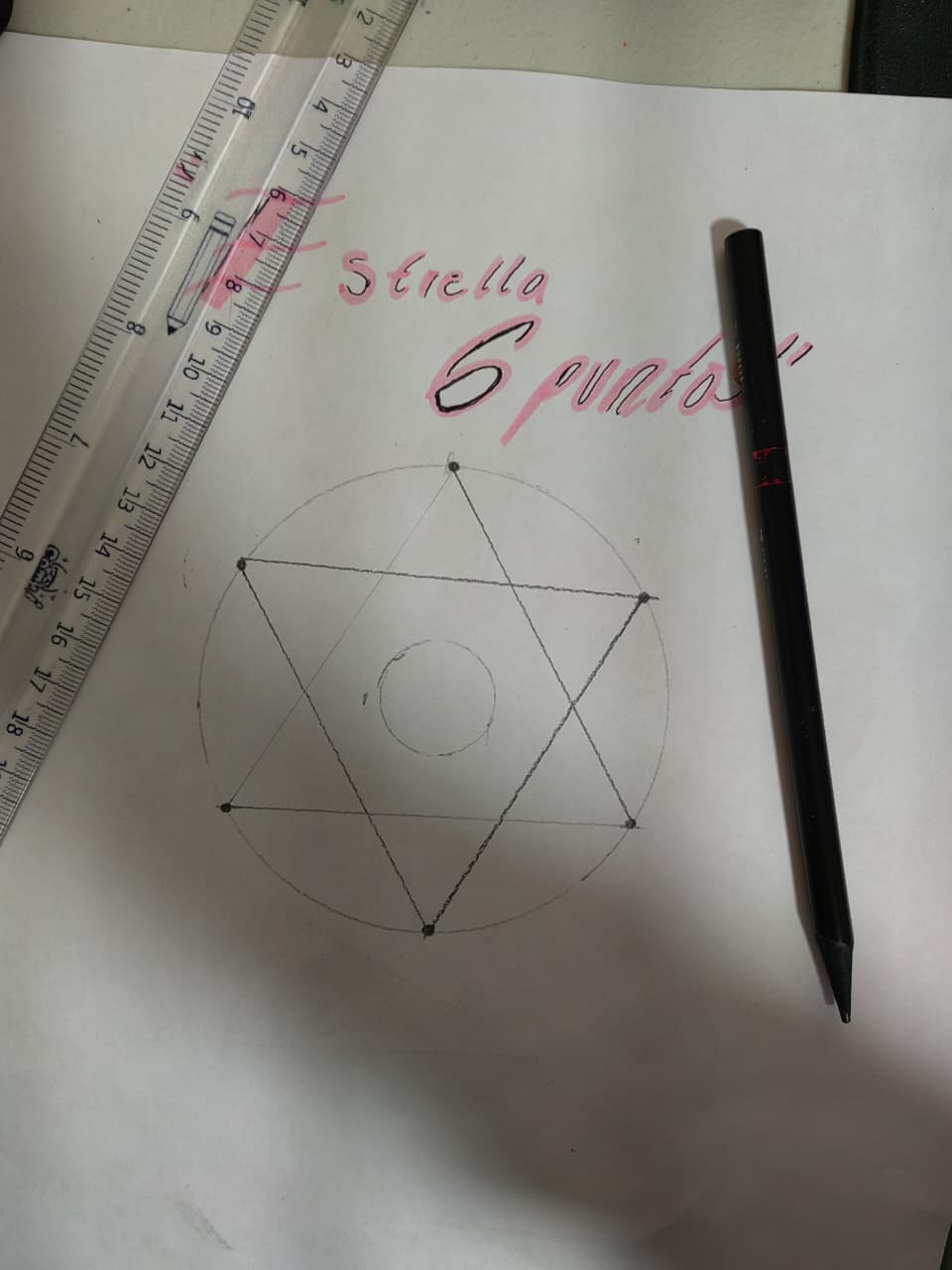
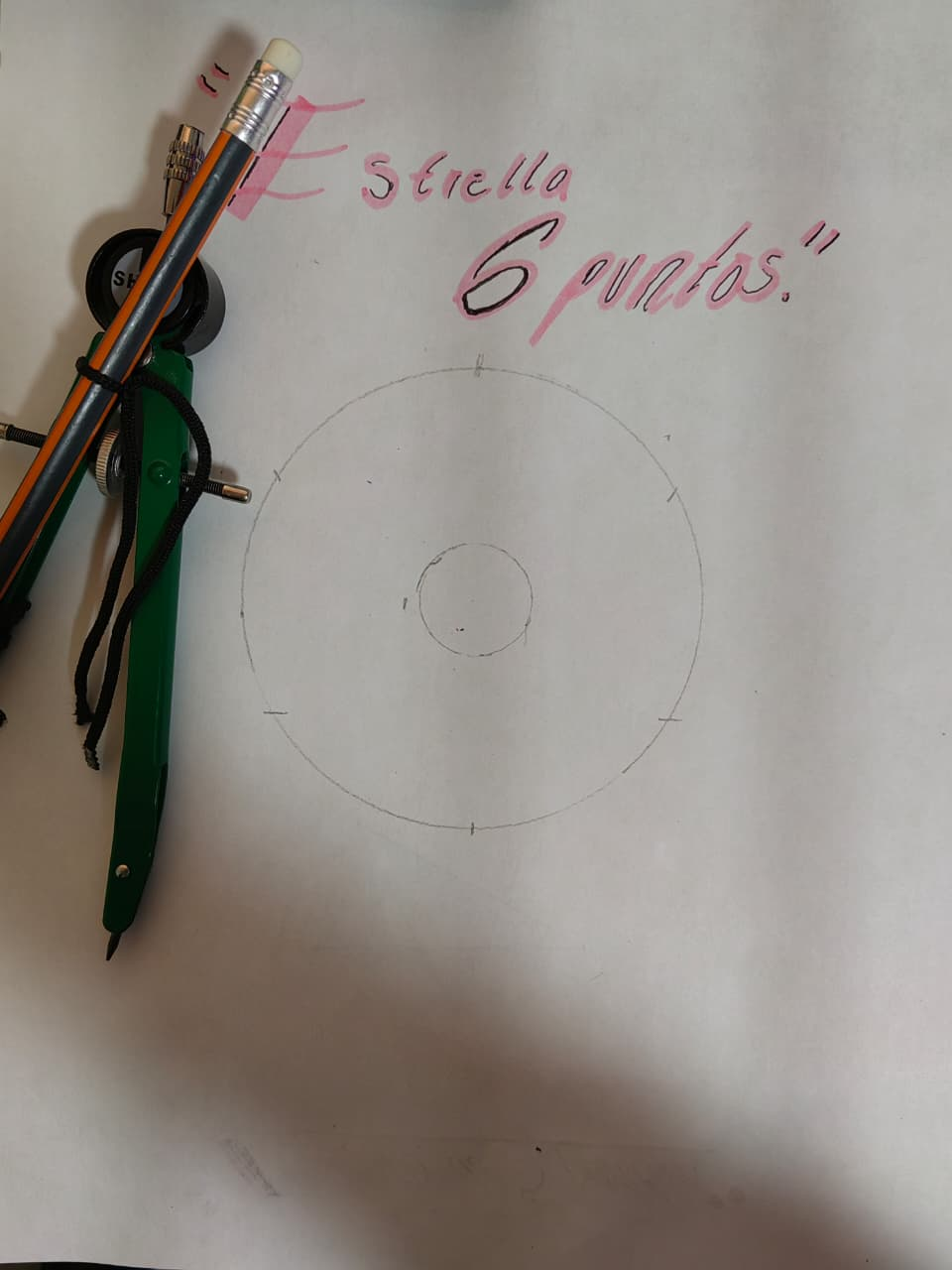
2. Ahora gira el compás, mientras mantienes la punta apoyada en el papel. El lápiz

dibujará un círculo perfecto alrededor de la punta del compás.

3. Marca un punto en la parte superior del círculo con el lápiz. Ahora, coloca la

punta del compás en la marca. No cambies el radio del compás con que hiciste el

círculo.



***Ejercicio 3***

Repetir ejercicio 3 de desarrollo para 5 alumnos. (Solicitar 3 calificaciones, obtener el promedio, si es mayor igual a 6 mostrar "Aprobado", en caso contrario mostrar "Reprobado”)

Solución:

1. Solicitar primer número y almacenarlo en una variable (calif1).

2. Solicitar el segundo número y almacenarlo en una variable (calif2).

3. Solicitar tercer número y almacenarlo en una variable (calif3).

4. Calcular la suma de los tres números (calif1 + calif2 + calif3 = SUMA)

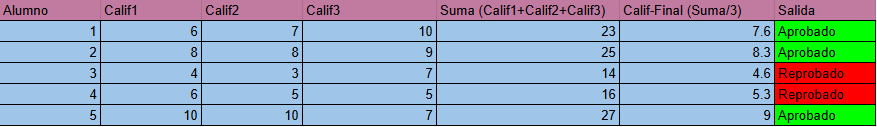
.

5. Dividir SUMA entre 3 (SUMA / 3 = calif-Final).

6. Si calif-Final ≥ 6, se realiza lo siguiente:

6.1. Mostrar “APROBADO”.

6.2. En caso contrario (calif-Final < 6), mostrar “REPROBADO”.



**Conclusión:**

La verdad está práctica me gusto mucho, ya que desde un inicio aprendí mucho sobre bastantes cosas como lo fueron los algoritmos, ciclo de vida del software, solución de problemas, variables y pruebas de escritorio. Todo esto para poder llevar a cabo un proceso y que cumpla una función.

El primer ejercicio sinceramente me sorprendió porque era bastante sencillo, pero si te ponías a pensar era una forma muy fácil y completa para entender los algoritmos.

El segundo ya subía la dificultad,ya que te pedía hacer pasos más específicos para poder llegar al resultado final, que en este caso fue una estrella de seis puntos.

Para el último ejercicio ya no era algo tan práctico como lo había hecho anteriormente, aquí era algo más informático y con números.

Sin duda alguna me gusto mucho lo aprendido, también en cómo te introducen de la forma más sencilla posible para que puedas entender lo que viene más adelante, mi favorito fue hacer estrellas porque demuestra que de un simple dibujo se puede llegar al objetivo que en este caso fue entender y practicar lo aprendido.

**Fuentes:**

Facultad de Ingeniería. (2025). Manual de prácticas del laboratorio de Fundamentos

de Programación. Laboratorio de computación. Salas A y B. Universidad Nacional

Autónoma de México. pp. 36-51. Recuperado el 13 de septiembre de 2025 de

<http://lcp02.fi-b.unam.mx/>