PRÁCTICO 2

TUPAR, TUDAI E INGENIERÍA

➢ Actividades de pre-lectura

1. Haga una lista de palabras transparentes.

2. **RELEA** la lista y formule por escrito una hipótesis sobre el tema que tratará el artículo.

➢ Actividades de lectura

A. several	1. ambos
B. latter	2. últimos
C. carriers	3 .dar
D. both	4. elemento de primera necesidad
E. fail	5. transmisores
F. staple	6. varios
G. provide	7. fracasar

3. **RELACIONE** las palabras con sus significados en Castellano.

4. ¿A qué palabras del texto hacen referencia los vocablos que en el artículo aparecen marcados por un círculo?

➢ Gramática

Oraciones condicionales

5. RECONOZCA en el texto dos ejemplos de oraciones condicionales.

¿De cuántas partes constan? Identifíquelas. ¿Qué forma verbal puede reconocer en cada una de sus partes?

6. Siguiendo el sentido del texto, **COMPLETE** los condicionales con el verbo entre paréntesis

If a standard software design book.....(design) it.....(be) easier to choose among design alternatives. (referencia pasada)

Unless routine designs .....carefully..... (record) solutions ..... (not come out) easily.

7. EXPLIQUE con sus palabras qué entiende por la expresión idiomática al final del primer párrafo "the latter are the bread and butter of engineering"

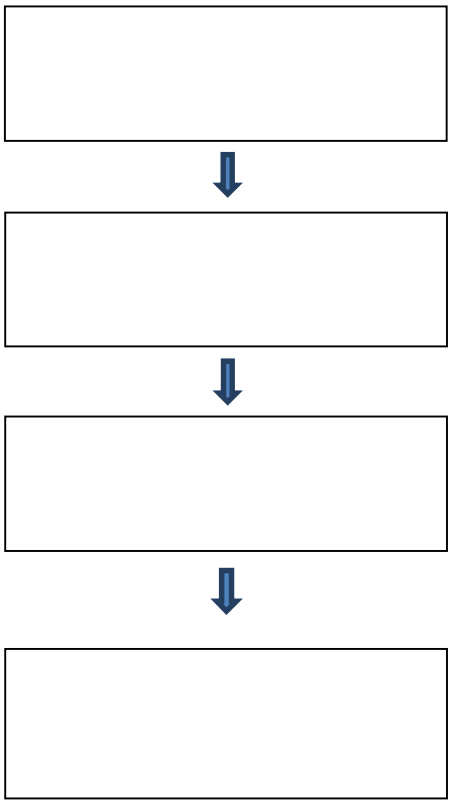
➢ Verificando la comprensión

8. ESCRIBA verdadero o falso.

a) La mayor parte del trabajo de diseño está dedicada a diseños de rutina.
b) La ingeniería del software capta, organiza y comparte el conocimiento de diseño para hacer el diseño de rutina más simple
c) El diseño de software termina casi siempre siendo innovador gracias a la recopilación de información acerca de los diseños de rutina.
d) La programación se ha visto beneficiada durante mucho tiempo por la codificación y almacenamiento de diseños en bibliotecas de subrutinas.

9. Diagrama de bloque

COMPLETE los casilleros con una frase que sintetice la idea de cada párrafo y le permita obtener un resumen de lo leído



➢ ACTIVIDADES DE ESCRITURA

- **UTILICE** el comienzo sugerido y las preguntas orientativas para redactar un párrafo.

En ingeniería del diseño, existen dos tipos de tareas típicas: las de rutina y las innovadoras. ¿Cómo caracteriza el autor a cada una?, ¿Cuál es el problema que se plantea?, ¿Cómo podría solucionarse?

.....
.....
.....
.....
.....
.....
.....

Routine and Innovative Design

Engineering design tasks are of several kinds; one of the most significant distinctions separates routine from innovative design. Routine design involves solving familiar problems, reusing large portions of prior solutions. Innovative design, on the other hand, involves finding novel solutions to unfamiliar problems. Original designs are much more rarely needed than routine designs, so the latter is the bread and butter of engineering.

Most engineering disciplines capture, organize, and share design knowledge in order to make routine design simpler. Handbooks and manuals are often the carriers of this organized information [Marks 87, Perry 84]. But current notations for software designs are not adequate for the task of both recording and communicating designs, so they fail to provide a suitable representation for such handbooks. Software in most application domains is treated more often as original than routine— certainly more so than would be necessary if we captured and organized what we already know. One path to increased productivity is identifying applications that could be routine and developing appropriate support. The current focus on reuse emphasizes capturing and organizing existing knowledge of a particular kind: knowledge expressed in the form of code. Indeed, subroutine libraries—especially of system calls and general-purpose mathematical routines—have been a staple of programming for decades. But this knowledge cannot be useful if programmers do not know about it or are not

encouraged to use it. Further, library components require more care in design, implementation, and documentation than similar components that are simply embedded in systems. Practitioners recognize the need for mechanisms to share experience with good designs. This cry from the wilderness appeared on a Software Engineering news groups:

"In Chem E, when I needed to design a heat exchanger, I used a set of references that told me what the constants were ... and the standard design equations ...

"In general, unless I, or someone else in my engineering group, has read or remembers and makes known a solution to a past problem, I'm doomed to recreate the solution. ... I guess ... the critical difference is the ability to put together little pieces of the problem that are relatively well known, without having to generate a custom solution for every application...

"I want to make it clear that I am aware of algorithm and code libraries, but they are incomplete solutions to what I am describing. (There is no Perry's Handbook for Software Engineering.)"

This former chemical engineer is complaining that software lacks the institutionalized mechanisms of a mature engineering discipline for recording and disseminating demonstrably good designs and ways to choose among design alternatives. Perry's handbook is the standard design handbook for chemical engineering; it is about 4 inches thick x 8-1/2" x 11", printed in tiny type on tissue paper [Perry 84].

Fuente:

Shaw, M. (1990) *Prospects for an Engineering Discipline of Software*. pp. 2-3