

### Manejo de partes y componentes

La primera tarea programada consiste en desarrollar en Eiffel un programa que permita almacenar un *catálogo* que dé información sobre qué *partes* se requieren para construir *componentes*. A partir de las definiciones de partes básicas se especifican componentes más complejos que involucran partes básicas o componentes más sencillos en cantidades fijas. Adicionalmente, se deben implementar varias operaciones de consulta algunas de las cuales requieren también de un *inventario* que indique la cantidad de partes y componentes disponibles para construir un grupo de componentes.

El programa debe implementar las siguientes operaciones:

#### Definición de una parte básica

Se introduce una nueva parte en el catálogo; de ahí en adelante se puede especificar dicha parte en la definición de componentes compuestos. Al definir una parte, se debe especificar un código único de identificación, un nombre único y un valor numérico que indique el grado de complejidad de dicha parte.

#### Definir un componente compuesto:

Se introduce en el catálogo un componente que consiste en la agregación de varias partes o *subcomponentes* en las cantidades especificadas. Al definir un componente se debe especificar un código único de identificación, un nombre único, y una lista de las partes o subcomponentes requeridos para construir el componente que se define. En el desglose se incluye además las cantidades respectivas de cada parte o subcomponente. Se debe calcular el valor de complejidad del nuevo componente usando la función de complejidad descrita más adelante.

#### Explosión de materiales:

Usando el catálogo definido dentro del sistema se da una solicitud para construir un grupo de componentes en cantidades varias. El sistema debe calcular cuánta materia prima (partes) se necesita para construir las cantidades pedidas de los componentes especificados. Se puede especificar explícitamente una lista de subcomponentes que no deben ser descompuestos en sus partes.

#### Componentes máximo:

Se da un inventario que especifica las cantidades disponibles de algunos materiales (partes y componentes) definidos en el catálogo. El sistema debe indicar cuáles componentes pueden construirse usando los elementos disponibles en dicho inventario y las cantidades que se pueden construir de cada uno. El sistema debe intentar construir los componentes más complejos posibles. Además se debe producir un inventario con los materiales sobrantes.

#### Calcular faltantes:

Se da un inventario que especifica las cantidades disponibles de algunos materiales

definidos en el catálogo. Para un componente dado, el sistema debe indicar cuáles partes y subcomponentes y en qué cantidades hacen falta para construir el componente dado. El sistema debe indicar el faltante en términos de los subcomponentes más complejos posibles.

Mostrar catálogo:

Lista la información de todas las partes y componentes definidos hasta ese momento.

Si bien en los ejemplos se muestran las operaciones anteriores como funciones independientes, el programa deberá ser desarrollado como un ciclo iterativo que responda a una serie de solicitudes del usuario.

### Función de complejidad

Todo componente y toda parte deben tener asociado un valor que mida la complejidad de dicho componente o parte. Para las partes, la complejidad es un valor dado al ser definidas. Para los componentes se usa la función que se describe a continuación para calcular su complejidad.

Suponer que C es un componente compuesto de las siguientes partes en las cantidades especificadas:  $(C_1, k_1), (C_2, k_2), \dots, (C_n, k_n)$ , eso es,  $C_i$  es el i-ésimo componente (o parte) que forma parte de C y  $k_i$  es la cantidad de instancias de dicho componente. La función de complejidad  $f_C$  se define recursivamente como:

$$f_C(C) = \log_2(1+k_1) * f_C(C_1) + \log_2(1+k_2) * f_C(C_2) + \dots + \log_2(1+k_n) * f_C(C_n)$$

### Consideraciones generales

El programa debe seguir los principios de la programación orientada a objetos.

La interfaz de entrada/salida debe ser independiente de la implementación de las operaciones descritas. En las corridas mostradas al final de este documento se usan unos comandos como **defp, defc, mostrar, expandir, componenteMax, faltante**. La interfaz puede variar un poco el formato de los comandos, pero **NO** debe ser un ciclo de menú que vaya pidiendo los parámetros uno por uno, como se muestra a continuación:

```
>> definirParte
>>> Dar codigo:
tco
>>> Dar nombre:
tornillo corto
>>> Dar complejidad
0.01
>>
```

Una interfaz como la anterior hace muy lentas las pruebas. Debe usar comandos o algún otro mecanismo que permita especificar las operaciones rápidamente.

Incluir en el código aserciones tal como se espera en el “Diseño por Contrato”.

**La fecha de entrega es el lunes 18 de noviembre a las 8am.**

## Ejemplos

1. Crear las siguientes partes como datos base y agregar al catálogo:

Código	Nombre	Complejidad
tco	tornillo-corto	0.01000
tla	tornillo-largo	0.02000
tue	tuerca	0.01000
lam	lamina	0.10000
e3d	esquina3d	0.12000

2. Crear los siguientes componentes y agregar al catálogo:

Código	Nombre	Subcomponente / parte	Cantidad
panel	panel	p002	4
		p003	12
		p004	2
cubo	cubo	p001	24
		p003	24
		p005	8
		p006	6
te	te-cubos	p001	20
		p003	20
		p007	6

Al calcular las complejidades de los componentes anteriores se obtienen los siguientes valores:

panel	0.24194
cubo	1.15248
te	3.32326

3. Se quieren construir dos cubos y una te. Dar las partes básicas necesarias:

expandir cubo 2   te 1

Resultado:

64 e3d esquina3d  
96 lam lamina  
192 tla tornillo-largo  
212 tco tornillo-corto  
788 tue tuerca

4. Se quiere construir dos cubos y una te. No descomponer los cubos:

expandir cubo 2   te 1 - cubo

Resultado:

8 cubo cubo  
20 tco tornillo-corto  
20 tue tuerca

5. Se quiere construir dos cubos y una te. No descomponer los paneles:

expandir cubo 2 te 1 - panel

Resultado:

48 panel panel  
64 e3d esquina3d  
212 tco tornillo-corto  
212 tue tuerca

6. Se tiene el siguiente inventario:

tco	100
tla	100
tue	200
lam	20
e3d	10

Dar los que se puede construir con él

componenteMax tco 100 tla 100 tue 200 lam 20 e3d 10

Resultado:

cubo	1
panel	4

Sobran:

tco	76
tla	60
tue	56
e3d	2

7. Para el mismo inventario anterior se desea construir dos cubos lo siguiente:

faltante cubo 2

Faltan:

lam	4
e3d	6

## Glosario

parte: unidad básica de construcción, su complejidad es dada por el usuario

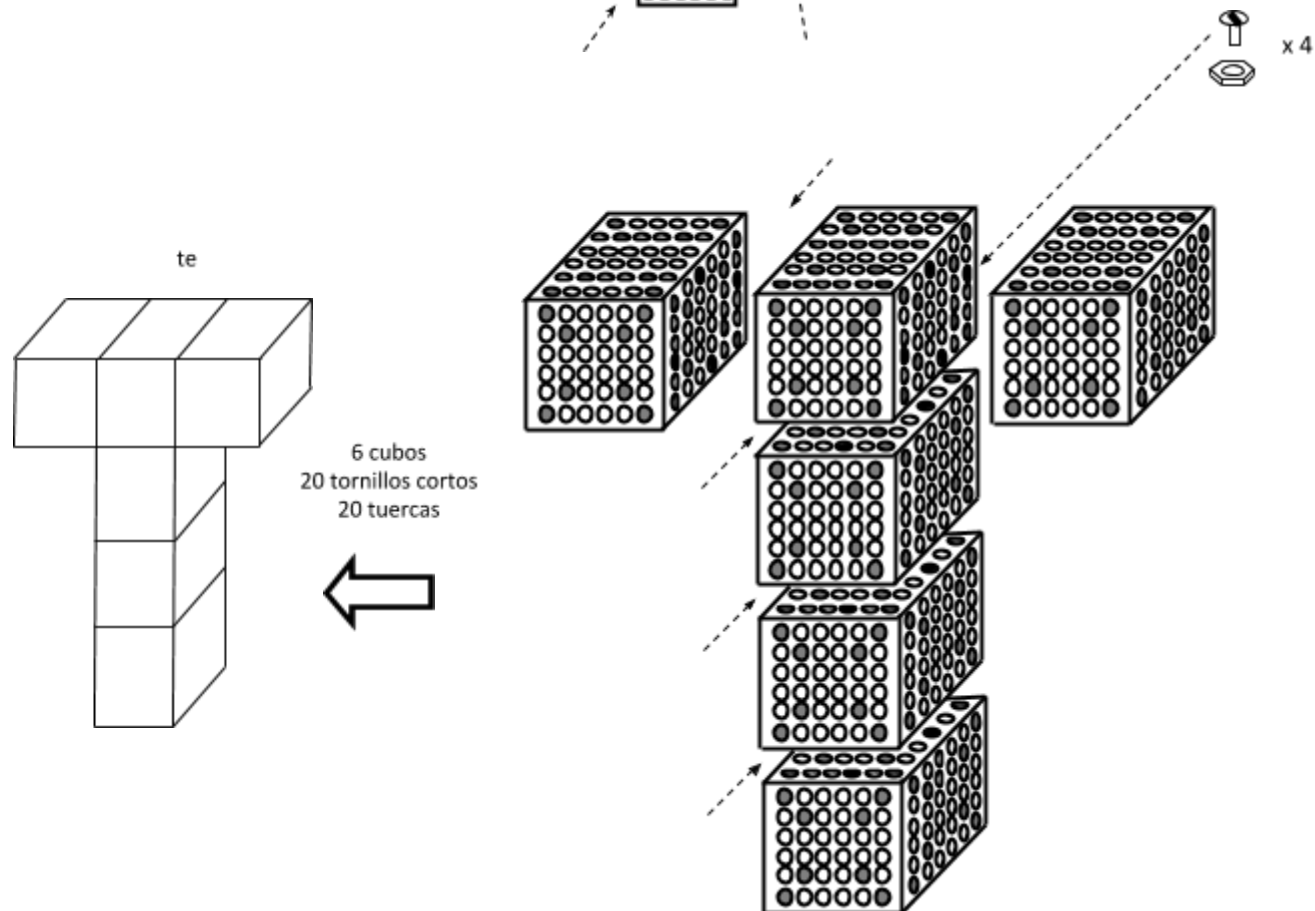
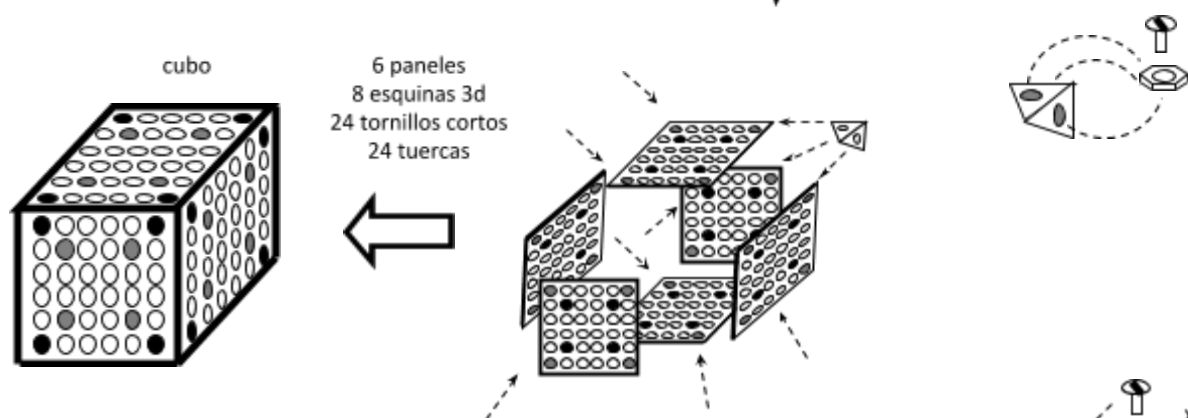
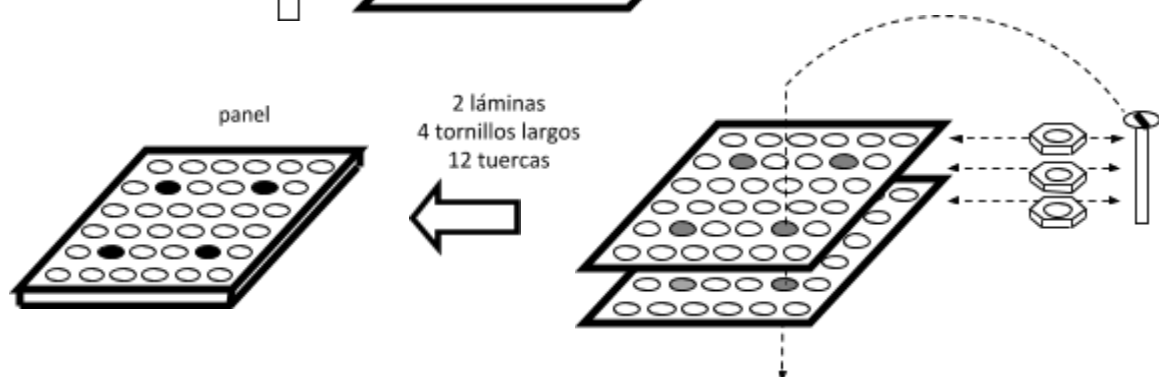
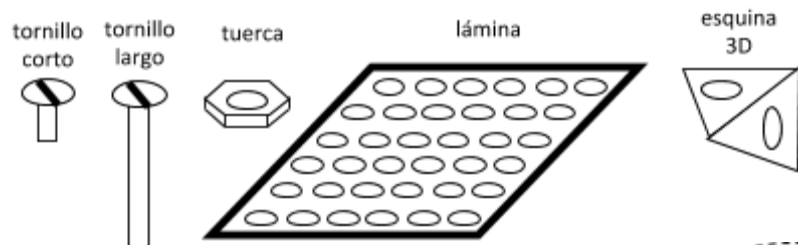
componente: unidad compuesta de partes u otros componentes más sencillos; su complejidad es calculada de acuerdo con la fórmula dada

subcomponente: componente que es parte de otro componente más complejo

catálogo: conjunto de descripciones de partes y componentes

inventario: lista que indica para algunas partes y componentes la cantidad disponible para cada uno de ellos





## Corrida 1

### 1. Definir las siguientes partes (agregar al catálogo):

```
>> defp tco tornillo-corto 0.01
>> defp tla tornillo-largo 0.02
>> defp tue tuerca 0.01
>> defp lam lamina 0.10
>> defp e3d esquina-3d 0.12
```

### 2. Definir los siguientes componentes (agregar al catálogo):

```
>> defc pan panel tla 4 tue 12 lam 2
>> defc cub cubo tco 24 tue 24 e3d 8 pan 6
>> defc te te-cubos tco 20 tue 20 cub 6
```

### 3. Mostrar catálogo

```
>> mostrar
```

#### Resultado

Código	nombre	complejidad
tco	tornillo-corto	0.01000
tla	tornillo-largo	0.02000
tue	tuerca	0.01000
lam	lamina	0.10000
e3d	esquina	0.12000
pan	panel	0.24194
cub	cubo	1.15248
te	te	3.32326

### 4. Construir dos cubos una te

```
>> expandir cub 2 te 1
```

#### Resultado

Cantidad	Código
212	tco
192	tla
788	tue
96	lam
64	e3d

### 5. Construir dos cubos y una te, sin descomponer los cubos:

```
>> expandir cub 2 te 1 - pan
```

#### Resultado

Cantidad	Código
20	tco
20	tue
8	cub

### 6. Dado un inventario dar el componente más complejo que se puede construir:

```
>> componenteMax tco 100 tla 100 tue 200 lam 20 e3d 10
```

#### Resultado

Se pueden construir:

Cantidad	Código
1	cub
4	pan

Sobran:

76	tco
60	tla
56	tue
2	e3d

### 7. Dado un inventario dar lo que falta para construir unos componentes:

```
>> faltante cub 2 - tco 100 tla 100 tue 200 lam 20 e3d 10
```

#### Resultado

Faltan:

Cantidad	Código
4	lam
6	e3d



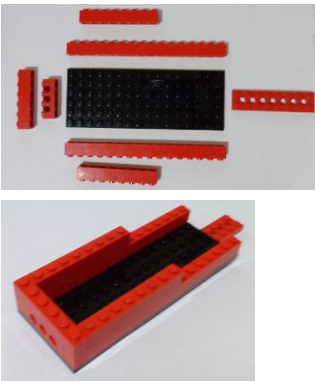
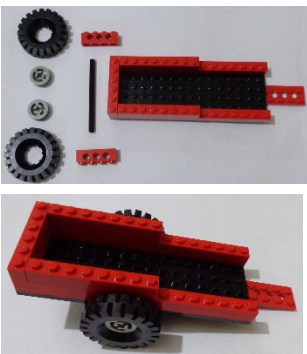


## Corrida 2

### 1. Definir las siguientes partes:

q1 ld6	ladrillo-6		0,15
q2 ld8	ladrillo-8		0,25
q3 ld16	ladrillo-16		0,35
q4 ld4h	ladrillo-4-con-huecos		0,20
q5 la6x16	lamina-6x16		0,30
q6 l2x8h	lamina-2x8-con-huecos		0,40
q7 aro	aro		0,50
q8 llan	llanta		0,45
q9 eje8	eje-8		0,17

### 2. Definir los siguientes componentes

€1 cajon	cajón 	1 ladrillo 4 con huecos 1 ladrillo 6 2 ladrillos 8 2 ladrillos 16 1 lamina 6x16 1 lamina 2x8 con huecos	2,001
€2 carr	carreta 	1 cajón 2 ladrillos 4 con hueco 2 aros 2 llantas 1 eje	5,471

### 3. expandir carr 2 cajon 1

3 ladrillos 6 6 ladrillos 8	3 laminas 6x16 3 laminas 2x8 con huecos
--------------------------------	--------------------------------------------

6 ladrillos 16	4 aros
7 ladrillos 4 con huecos	4 llantas
	2 ejes

4. expandir carr 1 cajon 1 – cajon (resultado erroneo)

<del>4 ladrillos 4 con huecos</del>	<del>4 llantas</del>
<del>4 aros</del>	<del>2 ejes</del>
	<del>3 cajones</del>

5. componenteMax

3 ladrillos 6	3 laminas 6x16
6 ladrillos 8	3 laminas 2x8 con huecos
6 ladrillos 16	4 aros
7 ladrillos 4 con huecos	4 llantas
	1 eje

Resultado

1 carreta

2 cajones

Sobran

2 ladrillos 4 con huecos

2 aros

2 llantas

6. faltante carr 2

3 ladrillos 6	3 laminas 6x16
6 ladrillos 8	3 laminas 2x8 con huecos
6 ladrillos 16	3 aros
4 ladrillos 4 con huecos	4 llantas
	1 eje

Resultado (faltan)

2 ladrillos 4 con huecos	1 aro
	1 eje