"La ciencia de programar" abridged

1. Una _____ es una asignación de valores de verdad a cada una de las letras proposicionales en una fórmula proposicional. Por ejemplo, dada la fórmula proposicional $\theta = p \vee q \wedge r \rightarrow r$, una _____ de θ es:

   $$\{\xi(p) = \bot, \xi(q) = \bot, \xi(r) = \top\}$$

   Con los valores de verdad que nos da una _____ podemos hallar el valor de verdad de nuestra fórmula proposicional, en nuestro ejemplo:

   $$\bot \vee \bot \wedge \top \rightarrow \top \quad \Rightarrow \quad \bot \wedge \top \rightarrow \top \quad \Rightarrow \quad \bot \rightarrow \top \quad \Rightarrow \quad \top$$

   (Nota: recuerda que $\top$ es verdadero y $\bot$ falso)

   ¿Cuál de las siguientes interpretaciones hace falsa a la fórmula proposicional $(p \wedge r) \vee q \leftrightarrow p \wedge (r \vee q)$?:

   $$\{\xi(p) = \top, \xi(q) = \bot, \xi(r) = \bot\}$$
   $$\{\xi(p) = \bot, \xi(q) = \top, \xi(r) = \top\}$$
   $$\{\xi(p) = \bot, \xi(q) = \bot, \xi(r) = \bot\}$$
   $$\{\xi(p) = \top, \xi(q) = \top, \xi(r) = \bot\}$$

2. Una fórmula bien formada (*fbf*) es una formula proposicional que se puede construir a partir de las siguientes reglas:

   a. si p es una variable proposicional, entonces p es una fbf
   b. si ♠ y ♣ son fbf, y ⊘ es un _____, entonces (♠ ⊘ ♣) es un fbf
   c. si ✠ es una fbf, entonces $\neg$(✠) es una fbf

3. La _____ de los operadores nos dice en que orden debemos realizar las operaciones. A continuación las _____ de algunos de los operadores más comunes en C++ (de mayor a menor precedencia):

   1. `(`, `)` (No son en realidad operadores en C++, pero nos indican que sentencia[1] evaluar primero)
   2. `!`, `(int)` (ó `(double)`, ó cualquier otro *casting*)
   3. `*`, `/`, `%`
   4. `+`, `-`
   5. `<`, `>`, `<=`, `>=`
   6. `==`, `!=`
   7. `&&`
   8. `||`

4. La _____ de los conectivos lógicos nos dice en que orden debemos realizar las operaciones en una fórmula proposicional para obtener su valor de verdad. La _____ de los operadores lógicos es, en orden, de mayor a menor:

   1. (,) (No realmente un conectivo lógico, pero nos indican la expresión que debe ser evaluada antes del resto)
   2. $\neg$
   3. $\wedge$, $\vee$
   4. $\rightarrow$, $\leftrightarrow$

   Fijate en cómo $\wedge$ y $\vee$ tienen la misma precendencia cómo connectivos lógicos, pero esto no sucede en C++ dónde `&&` tiene mayor precedencia que `||`

5. ¿Cuál de las siguientes sentencias en C++ es la mejor traducción de la fórmula lógica $p \vee q \wedge r \wedge r$ a C++?

   a. `p && q || r || r`
   b. `((p || q) && r) && r`
   c. `p || (q && (r && r))`
   d. `(p || q) && (r && r)`

[1] no pude encontrar una mejor traducción para "sentence" :S

6. Dos fórmulas proposicionales son equivalentes si tienen las mismas tablas de verdad, es decir, si θ_1 y θ_2 son fórmulas proposicionales, ambas son equivalentes si y sólo si sus tablas de verdad son las mismas.

   Determine cuál de las siguientes fórmulas proposicionales es equivalente a $p \to q \wedge p$, para esto escriba las tablas de verdad de cada una de las proposiciones:

p	q	$p \to q \wedge p$	$p \leftrightarrow q \wedge p$	$\neg p \wedge q \vee p$	$(\neg p \vee q) \wedge p$	$\neg p \vee (q \wedge p)$
$\top$	$\top$					
$\top$	$\bot$					
$\bot$	$\top$					
$\bot$	$\bot$					

   Otra forma de determinar si dos fórmulas lógicas son equivalentes es mostrando una serie de equivalencias conocidas entre ellas, por ejemplo, la fórmula lógica $p \vee \neg(p \vee r)$ es equivalente a $p \vee \neg r$ porque:

$$
\begin{aligned}
p \vee \neg(p \vee r) &\equiv p \vee (\neg p \wedge \neg r) && \text{Por ley de De Morgan } \neg(\triangle \vee \square) \equiv \neg \triangle \wedge \neg \square \\
&\equiv (p \vee \neg p) \wedge (p \vee \neg r) && \text{Propiedad distributiva de } \vee \text{ en } \wedge \\
&\equiv \top \wedge (p \vee \neg r) && \triangle \vee \neg \triangle \equiv \top \\
&\equiv p \vee \neg r && \text{Dominancia en } \wedge \quad \top \wedge \triangle \equiv \triangle
\end{aligned}
$$

   Donde cada una de las equivalencias mencionadas a la izquierda son bien conocidas y poseen nombre propio (aunque no es necesario aprenderse el nombre, es siempre buena idea saberse algunas equivalencias de memoria).

7. Algunas equivalencias son:

Equivalencia	**Nombre**
$\triangle \to \square \equiv \neg \triangle \vee \square$	
$\triangle \vee \bot \equiv \triangle \qquad \triangle \wedge \top \equiv \triangle$	Dominancia
$\neg(\triangle \vee \square) \equiv \neg \triangle \wedge \neg \square \qquad \neg(\triangle \wedge \square) \equiv \neg \triangle \vee \neg \square$	Leyes de De Morgan
$\triangle \vee (\square \wedge \Diamond) \equiv (\triangle \vee \square) \wedge (\triangle \vee \Diamond)$	Propiedad distributiva de $\vee$ en $\wedge$
$\triangle \wedge (\square \vee \Diamond) \equiv (\triangle \wedge \square) \vee (\triangle \wedge \Diamond)$	Propiedad distributiva de $\wedge$ en $\vee$
$\triangle \wedge \neg \triangle \equiv \bot$	Contradicción
$\triangle \vee \neg \triangle \equiv \top$	Tercio excluído
$\neg\neg \triangle \equiv \triangle$	Doble negación
$\triangle \vee \square \equiv \square \vee \triangle \qquad \triangle \wedge \square \equiv \square \wedge \triangle \qquad \triangle \leftrightarrow \square \equiv \square \leftrightarrow \triangle$	Conmutatividad(es)
$\square \leftrightarrow \triangle \equiv (\square \to \triangle) \wedge (\triangle \to \square)$	Material

   ¿Qué equivalencia se puede utilizar para transformar la fórmula lógica $p \to q \wedge p$ (punto anterior) para hallar otra fórmula lógica equivalente a esta?

8. Los _____ que uno usualmente usa para modelar un problema son $\mathbb{N}$, $\mathbb{R}$ y $\mathbb{B}$. Estos _____ no se encuentran disponibles en C++, pero nos aproximamos a estos con los tipos de datos: `int`, `double` y `bool`, respectivamente.

   Podemos definir un conjunto más, el conjunto $\mathbb{ASCII}$, el cual corresponde al tipo de datos `char` disponible en C++.

   En la siguiente tabla se encuentran todos los tipos de datos de C++ que nos interesan, por el momento, y sus simíles con conjuntos matemáticos:

Conjunto	Tipo de datos
$\mathbb{N} = \{0, 1, 2, 3, 4, ...\}$	`___`
$\mathbb{R}$	`double`
$\mathbb{B} = \{\top, \bot\}$	`____`
_____ $= \{$`'A','a','B','b',...,'!','#','/','\n',...`$\}$	`char`

9. Para convertir una función matemática en su equivalente en C++, lo primero que uno requiere es saber el tipo del dominio de la función así como también el tipo del rango de la función, por ejemplo, dada la función $par : \mathbb{N} \to \mathbb{B}$, uno sabe que su declaración en C++ será: `bool par(int n)`.

   Dada la función $descuento : \mathbb{N} \times \mathbb{B} \times \mathbb{R} \to \mathbb{B}$, escribir su declaración en C++.

10. ¿Qué conectivo lógico es necesario para que la siguiente expresión sea una tautología?:

$$((p \vee q) \wedge (p \to r) \wedge (q \to r)) \ \_ \ r$$

11. La **traza** de un programa es cómo cada variable cambia a medida que se ejecuta cada sentencia de un programa. Por ejemplo, dada la siguiente porción de programa (y suponiendo que todas las variables ya están declaradas, con `r = 4`):

```
l = r + 3;
m = 23*r;
n = m + r;
o = n*l;
l = n+o%m;
```

   La traza de la porción de código será:

t	r	l	m	n	o
0	4	—	—	—	—
1	4	7	—	—	—
2	4	7	92	—	—
3	4	7	92	96	—
4	4	7	92	96	672
5	4	124	92	96	672

   ¿Cuál es la traza del siguiente programa? (suponiendo `mm = -2`):

```
rri = 32;
var = -mm * rri;
ll = var % 4;
var = (ll % 3)*11;
mm = var - ll + rri;
```

12. Nos piden escribir un programa que sea capaz de llevar la contabilidad de una organización, para esto nos dan una lista de requerimientos. Uno de los requerimientos es: "El día jueves hay promoción de alimentos lácteos sólo para nuestros clientes frecuentes y para aquellos que lleven más de 200.000 pesos en comida".

    En el sistema decidimos declarar una variable que determine el día de la semana (dia como un natural, dónde cero es lunes), otra que determine si un cliente es frecuente ($frecuente$ como un booleano), y una última que determine el total de lo que el cliente a de comprar ($total\_compra$ como un real). Una vez determinados las variables, podemos escribir una expresión que determine si se le va a hacer descuento por los lácteos o no:

    $dia = 3 \wedge (frecuente \vee total\_compra > 200000)$

    Esta expresión la podemos convertir fácilmente a C++, la cual sería:

---

13. Cuáles de los siguientes son nombres válidos para una variable o función, y cúales no (¿por qué?):

    `inta`, `variable`, `veryfreakinglongnameof0avariable`, `double`, `3nums`, `num__`, `num_03`, `23_a`, `3.4e-2`, `aeo.02e`, `werid_etr09_2`

PD: Esta guía esta construida para ser un resumen del libro, sin embargo no contiene todo lo que este mismo dice, un par de temas más que **hay** que conocer son: Operaciones entre conjuntos y predicados (una extensión del cálculo proposicional).