Tarea 1
Lógica Proposicional

Ejercicio 1 — 24 Pt

En este ejercicio le vamos a pedir que escriba un programa que dada una fórmula de la lógica proposicional en forma normal conjuntiva, determine si es satisfactible o no. Para describir el ejercicio usaremos Python, pero también podrá utilizar Java o C++.

Las fórmulas van a estar codificadas a través de listas de listas de enteros (no nulos). Cada entero va a representar un literal, siendo positivo si corresponde a una variable proposicional y negativo si corresponde a su negación. Por ejemplo la lista [[1, 2, -3], [-1, -2, 4], [3, 4]] corresponde a la fórmula $(p_1 \vee p_2 \vee \neg p_3) \wedge (\neg p_1 \vee \neg p_2 \vee p_4) \wedge (p_3 \vee p_4)$. Asumiremos que ninguna cláusula contendrá simultáneamente una variable proposicional y su negación.

(a) [8 Pt] En la primer parte del ejercicio le pediremos que implemente la función `SetLiteral`. Dicha función tomará una fórmula y un literal, y devolverá la fórmula simplificada que se obtiene al asignarle *true* al literal. Por ejemplo

```
>>> SetLiteral([[1, 2, -3], [-1, -2, 4], [3, 4]], 1)
[[-2, 4], [3, 4]]
```

La cláusula $p_1 \vee p_2 \vee \neg p_3$ fue eliminada de la fórmula porque se vuelve verdadera cuando $p_1 = true$. De manera similar, la cláusula $\neg p_1 \vee \neg p_2 \vee p_4$ fue simplificada a $\neg p_2 \vee p_4$. Éstos son los (únicos) dos tipos de "simplificaciones" que llevará a cabo la función. Considere un último ejemplo:

```
>>> SetLiteral([[1, 2, -3], [-1, -2, 4], [3, 4]], -1)
[[2, -3], [3, 4]]
```

(b) [8 Pt] Ahora le pediremos que implemente la función `IsSatisfiable`. Dicha función toma una fórmula y devuelve *true* o *false* señalando si la fórmula es satisfactible o no. Por ejemplo

```
>>> IsSatisfiable([[1, 2, -3], [-1, -2, 4], [3, 4]])
True
```

```
>>> IsSatisfiable([[1,2], [-1,-2], [-1,2],[1,-2]])
False
```

Para determinar si una fórmula es satisfactible, la función `IsSatisfiable` va a operar recursivamente. Primero va a elegir un literal arbitrario de la fórmula. Luego va a setear el literal a *true*, va a simplificar la fórmula (usando `SetLiteral`) y va a determinar (recursivamente) si la formula así obtenida es satisfactible. Si ése no es el caso, va a repetir el proceso anterior (usando el mismo literal), pero esta vez seteandolo a *false*. Al implementar la función, piense con detenimiento cuál(es) debería(n) ser el(los) caso(s) base(s).

(c) [8 Pt] Por último le pediremos que implemente la función `BuildModel`. Dicha función tomará una fórmula y devolverá un par. La primer componente del par es un Booleano indicando si la fórmula es satisfactible o no. Si la fórmula es satisfactible, la segunda componente del par debe ser una valuación que hace verdadera la fórmula. Si por el contrario la fórmula es insatisfactible, la segunda componente del par debe ser una valuación vacía. Por ejemplo,

```
>>> BuildModel([[-2, 4], [1], [-4,-1]])
(True, {1: True, 2: False, 4: False})
```

```
>>> BuildModel([[1,2], [-1,-2], [-1,2],[1,-2]])
(False, {})
```

Al exhibir las valuaciones debe mostrar el valor de verdad de las variables proposicionales (por ejemplo `1:True`) y no el valor de sus negaciones (por ejemplo `-1:False`). Por último, se permite exhibir una valuación parcial cuando la fórmula es verdadera cualquiera sea el valor de verdad de las variables no exhibidas. Por ejemplo

```
>>> BuildModel([[1, 2, -3], [-1, -2, 4], [3, 4]])
(True, {1: True, 2: False, 3: True})
```

Para resolver este ejercicio complete el código del archivo T1.py, T1.cpp o T1.java y entréguelo dentro de un *zip* conteniendo la resolución de los demás ejercicios. ¡Incluya sólo uno de estos tres archivos! Si el *zip* contiene más de uno, al ejercicio se le otorgará 0 puntos.

Ejercicio 2 — 6 Pt

Una fórmula de la lógica proposicional formada a partir de una única variable proposicional admite 4 posibles tablas de verdad:

p	**0**	**1**	**id**	**neg**
0	0	1	0	1
1	0	1	1	0

Dicho de otra manera, existen 4 operadores unarios: el operador constante 0, el operador constante 1, el operador identidad y el operador negación. En general, ¿cuántos operadores n-arios existen para $n \in \mathbb{N}$? Justifique su respuesta.

Ejercicio 3 — 14 Pt

Sea P un conjunto (no vacío) de variables proposicionales y sean, además, $\alpha, \varphi_1, \ldots, \varphi_n$ fórmulas de la lógica proposicional en $\mathcal{L}(P)$. Demuestre las siguientes proposiciones:

(a) [6 Pt] $\{\varphi_1, \ldots, \varphi_n\} \models \alpha$ si y sólo si $\varphi_1 \wedge \ldots \wedge \varphi_n \to \alpha$ es una tautología.

(b) [4 Pt] El conjunto vacío es satisfacible.

(c) [4 Pt] El conjunto $\mathcal{L}(P)$ (de todas las fórmulas de lógica proposicional sobre P) es insatisfactible.

Ejercicio 4 — 16 Pt

Supongamos que usted está cursando 3 ramos: "Discretas", "Algoritmos" y "Gráficas". Para modelar en qué bloques tiene clases de cada ramo introducimos el siguiente conjunto de variables proposicionales:

$$\begin{aligned} P \quad := \quad & \{D_{ij} \mid 1 \leq i \leq 5, 1 \leq j \leq 6 \} \cup \\ & \{A_{ij} \mid 1 \leq i \leq 5, 1 \leq j \leq 6 \} \cup \\ & \{G_{ij} \mid 1 \leq i \leq 5, 1 \leq j \leq 6 \} \end{aligned}$$

Aquí, D, A y G denotan el ramo, el subíndice i denota el día (1 significa lunes, 5 significa viernes) y el subíndice j denota el bloque (1 representa el bloque 8:30-10:00, 6 representa el bloque 18:00-19:30).

Así, por ejemplo, si D_{23} es verdadero significa que usted tiene clases de Discretas el martes entre las 12:00 y las 13:30.

Utilizando este conjunto de variables proposicionales, se pide que exprese cada una de las siguientes propiedades a través de una fórmula de la lógica proposicional:

(a) [5 Pt] Que su horario no tenga ninguna clase en el horario de 8:30 a 10:00.

(b) [5 Pt] Que su horario no tenga ningún choque.

(c) [6 Pt] Que su horario no tenga ventanas. (En otras palabras, que todas las clases del día estén en bloques consecutivos.)