Exercice 1. Système complet logique
On définit les opérateurs NAND, NOR, XOR par leurs tables de vérité :

x	y	$x\ NAND\ y$	$x\ NOR\ y$	$x\ XOR\ y$
0	0	1	1	0
0	1	1	0	1
1	0	1	0	1
1	1	0	0	0

On dit qu'un ensemble S d'opérateurs logiques est complet si toute formule logique est équivalente à une formule qui n'utilise que des opérateurs dans S.

1. Exprimer NAND, NOR, XOR, à l'aide de $\vee$, $\wedge$, $\neg$.
2. Montrer que $\{\wedge, \neg\}$ est complet.
3. Montrer que $\{NAND\}$ est complet. (c'est pour cette raison que le NAND est très utilisé en électronique)
4. Montrer que $\{NOR\}$ est complet.
5. Montrer que $\{XOR\}$ n'est pas complet.
6. Montrer que $\{XOR, \neg\}$ n'est pas complet.

Exercice 2. Extrait Mines-Pont 2010

Étant donnée une formule logique f sous forme normale conjonctive, on note dans ce problème $\max(f)$ le nombre maximum de clauses de f pouvant être satisfaites par une même valuation.
En notant m le nombre de clauses de f, on remarque que f est satisfiable si et seulement si $\max(f) = m$.

On considère la formule f_1 (sous forme normale conjonctive) dépendant des variables x, y, z :

$$f_1 = (x \vee y \vee z) \wedge (\neg x \vee \neg y \vee \neg z) \wedge (\neg x \vee \neg y) \wedge (\neg x \vee \neg z) \wedge (x \vee \neg y \vee z)$$

1. Indiquer si f_1 est satisfiable ou non et, si elle est satisfiable, donner l'ensemble des solutions de f_1.

Une instance de 3-SAT est une formule logique sous forme normale conjonctive dont toutes les clauses contiennent 3 littéraux.

2. Déterminer une instance f_2 de 3-SAT non satisfiable et possédant exactement 8 clauses; indiquer $\max(f_2)$ en justifiant la réponse.

On considère une instance f de 3-SAT définie sur n variables. On note V l'ensemble des 2^n valuations des variables de f.
Soit val une valuation des n variables. Si C est une clause, on note $\varphi(C, val)$ la valeur de C pour la valuation val et on note $\psi(f, val)$ le nombre de clauses de f qui valent 1 pour la valuation val.
On a : $\psi(f, val) = \sum_{C \text{ clause de } f} \varphi(C, val)$ et $\max(f) = \max_{val \in V} \psi(f, val)$.

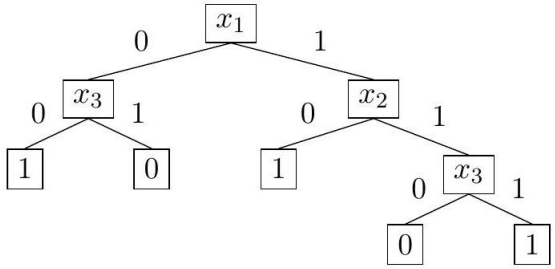
3. Soit C une clause de f. Donner une expression simple de $\sum_{val \in V} \varphi(C, val)$, en fonction de n.
4. Soit m le nombre de clauses dont f est la conjonction.
   En considérant la somme $\sum_{C \text{ clause de } f} \sum_{val \in V} \varphi(C, val)$, donner en fonction de m un minorant de $\max(f)$.
5. Donner le nombre minimum de clauses d'une instance de 3-SAT non satisfiable.

Exercice 3. Arbre de décision (Oral ENS info)

On considère un ensemble de variables propositionnelles $\mathcal{X} = \{x_1, \ldots, x_n\}$ muni de l'ordre total où $x_i < x_j$ si et seulement si $i < j$. Un arbre de décision sur $\mathcal{X}$ est un arbre binaire dont les feuilles sont étiquetées par 0 ou par 1 , et dont les nœuds internes sont étiquetés par une variable de $\mathcal{X}$ et ont deux enfants appelés enfant 0 et enfant 1 . On impose que si un nœud interne n étiqueté par x_i a un descendant n' qui est un nœud interne étiqueté par x_j alors $x_i < x_j$.

Un arbre de décision T sur $\mathcal{X}$ décrit une fonction booléenne Φ_T qui à toute valuation $\nu : \mathcal{X} \to \{0,1\}$ associe une valeur de vérité calculée comme suit : si T consiste exclusivement d'une feuille étiquetée $b \in \{0,1\}$ alors la fonction Φ_T s'évalue à b quelle que soit ν. Sinon, on considère le nœud racine n de T et la variable x_i qui l'étiquette, on regarde la valeur $\nu(x_i) \in \{0,1\}$ que ν donne à x_i, et le résultat de l'évaluation de Φ_T sous ν est celui de l'évaluation de $\Phi_{T'}$ sous ν, où T' est le sous-arbre de T enraciné en l'enfant b de n.

1. On considère l'arbre de décision T_0 suivant et la fonction Φ_{T_0} qu'il définit. Évaluer cette fonction pour la valuation donnant à x_1, x_2, x_3 respectivement les valeurs $0, 1, 0$. Donner un exemple de valuation sous laquelle cette formule s'évalue en 0 .



2. On considère la formule de la logique propositionnelle $(x_1 \wedge x_2) \vee \neg(x_1 \wedge \neg x_3)$. Construire un arbre de décision sur les variables $x_1 < x_2 < x_3$ qui représente la même fonction.
3. Quels arbres de décision représentent des tautologies? des fonctions satisfiables?
4. Étant donné un arbre de décision représentant une fonction Φ, expliquer comment on pourrait construire un arbre de décision représentant sa négation $\neg\Phi$.

On définit un type pour les formules logiques :

```
type formule =
    | Var of int
    | Et of formule * formule
    | Ou of formule * formule
    | Non of formule
```

On définit aussi un type pour les arbres de décision (où le sous-arbre gauche est celui de valeur 0 et le sous-arbre droit celui de valeur 1) :

```
type arbre = F of int | N of int * arbre * arbre
```

5. Écrire une fonction `arb_to_for : arbre -> formule` permettant de convertir un arbre de décision en formule logique. Analyser sa complexité en temps et en espace.
6. Écrire une fonction `and : arbre -> arbre -> arbre` telle que, si `a1` et `a2` sont des arbres représentant des formules `f1` et `f2`, `and a1 a2` renvoie un arbre représentant la formule `Et (f1, f2)`.
Quelle est sa complexité en temps et la taille de l'arbre obtenu ?
7. Étant donné un arbre de décision et la séquence de variables $x_1, \ldots, x_n$, donner le pseudocode d'un algorithme qui calcule combien de valuations satisfont la fonction booléenne qu'il capture. Quelle est sa complexité en temps?
8. Peut-on efficacement récrire une formule quelconque de la logique propositionnelle en arbre de décision?