OPTION INFORMATIQUE
Devoir surveillé 3

1 Cherchez les Clés du Paradis (d'après CCP 2018)

1.1 Préparation

Dans la suite on note $\mathbb{B} = \{F, V\}$ l'ensemble des booléens, avec F désignant le faux et V le vrai.
On utilisera la notation $\overline{\phi}$ pour désigner $\neg\phi$, où ϕ est une formule.
On introduit dans la syntaxe du calcul propositionnel les constantes $\bot$ et $\top$, $\bot$ étant toujours interprétée par F, et $\top$ par V.

1. Pour A une variable propositionnelle, simplifier les formules :
   $A \vee \bot \quad A \vee \top \quad A \wedge \bot \quad A \wedge \top \quad A \vee \overline{A} \quad A \wedge \overline{A} \quad A \to \bot \quad A \to \top \quad \bot \to A \quad \top \to A$

On introduit également le constructeur binaire $\oplus$ (lire xor), dont la sémantique est définie ainsi : une valuation v satisfait $\phi \oplus \psi$ si et seulement si v satisfait exactement une formule parmi ϕ et ψ.
On considérera dans la suite que c'est ce connecteur logique que désigne un « ou bien » en français.

1. Pour A, B deux variables propositionnelles, donner une forme normale conjonctive de $A \oplus B$, puis une forme normale disjonctive.

1.2 Les épreuves

Vous avez été sélectionné(e) pour participer au jeu "Cherchez les Clés du Paradis (CCP)".

Première épreuve

Jean-Pierre Pendule, le célèbre animateur, vous accueille pour la première épreuve. Il vous explique la règle du jeu. Devant vous, deux boîtes et sur chacune d'entre elles une inscription. Chacune des boîtes contient soit une clé verte, soit une clé rouge. Vous devez choisir l'une des boîtes : si le résultat est une clé rouge, alors vous quittez le jeu, si c'est une clé verte vous remportez l'épreuve. Jean-Pierre Pendule dévoile les inscriptions sur chacune des boîtes et vous affirme qu'elles sont soit vraies toutes les deux, soit fausses toutes les deux :

- sur la boîte 1, il est écrit "Une au moins des deux boîtes contient une clé verte" ;
- sur la boîte 2, il est écrit "Il y a une clé rouge dans l'autre boîte".

Dans toute cette partie, on note P_i la proposition affirmant qu'il y a une clé verte dans la boîte i.

1. Donner une formule représentant la phrase écrite sur la boîte 1. Donner une seconde formule pour la boîte 2.
2. Donner une formule représentant l'affirmation de l'animateur. Simplifier cette formule de sorte à n'obtenir qu'une seule occurrence de chaque P_i.
3. Quel choix devez-vous faire pour continuer le jeu à coup sûr ?

Deuxième épreuve

Bravo, vous avez obtenu la première clé verte, et parvenez à la deuxième épreuve. Avec les mêmes règles du jeu, l'animateur vous propose alors deux nouvelles boîtes portant les inscriptions suivantes :

- sur la boîte 1, il est écrit "Il y a une clé rouge dans cette boîte, ou bien il y a une clé verte dans la boîte 2" ;
- sur la boîte 2, il est écrit "Il y a une clé verte dans la boîte 1".

1. Donner une formule de la logique des propositions pour chaque affirmation.
2. Dresser la table de vérité de ces formules et en déduire l'ensemble des valuations satisfaisant les règles du jeu. En déduire la boîte que vous devez choisir.

Troisième épreuve

Sous les acclamations du public, vous parvenez à l'épreuve finale. Jean-Pierre Pendule sourit et vous demande simplement :

1. Montrer que $\{\oplus, \to\}$ est un système complet de connecteurs.

2 Pile avec oubli

On s'intéresse ici à une structure de données de pile avec oubli, similaire à une pile avec capacité, mais dans laquelle un empilement est toujours possible. Dans le cas où on empile une nouvelle valeur alors que la capacité est déjà atteinte, la valeur en fond de pile est retirée en compensation. Cette structure de données est par exemple utilisée par les éditeurs de documents pour stocker les versions d'un fichier et permettre d'annuler les dernières modifications.
Par souci de simplicité, on considère des piles d'entiers. On suppose que la capacité `c` est une variable globale. Les opérations disponibles sont :

- `creer_pile ()` : renvoie une pile avec oubli de capacité c,
- `est_vide p` : détermine si la pile `p` est vide,
- `empiler p v` : modifie la pile `p` en ajoutant la valeur `v` au sommet. Si la capacité est dépassée, la valeur en fond de pile est retirée.
- `depiler p` : modifie la pile `p` en retirant le sommet, et renvoie cette valeur. Si la pile est vide, une erreur est provoquée à la place.

Implémenter ces opérations en Caml à l'aide de tableaux. Chaque opération devra avoir une complexité en $O(1)$. On expliquera sur un exemple la représentation utilisée.

3 Arbres binaires de recherche

Dans cette partie, on considère des arbres binaires définis par le type Caml

```
type 'a arbre = V | N of 'a arbre * int * 'a * 'a arbre;;
```

Pour un noeud $N(g, c, x, d)$, g correspond au fils gauche de ce noeud, c à la clé associée au noeud, x à la valeur associée au noeud, et d au fils droit de ce noeud.

Si a est un arbre et n un entier, on notera $a < n$ si toutes les clés de a sont strictement inférieures à n. On définit de même $a \leq n, a > n, a \geq n$.

1. En utilisant ces notations, donner une définition inductive de la propriété pour un arbre binaire d'être un arbre binaire de recherche (on utilisera des inégalités strictes sur les clés).
2. (a) Écrire une fonction `valeur_associee` prenant en argument un arbre binaire de recherche et une clé entière, et renvoyant sa valeur associée. Une erreur sera levée si la clé n'apparaît pas dans l'arbre.
   (b) Déterminer la complexité de cette fonction.
3. (a) Écrire une fonction `maptree` prenant en argument une fonction `f : int -> int` et un arbre binaire a et renvoyant l'arbre obtenu à partir de a en appliquant f à chaque clé.
   (b) Donner une condition nécessaire et suffisante sur f pour que `maptree f` transforme tout arbre binaire de recherche en arbre binaire de recherche.
   (c) Démontrer rigoureusement le caractère nécessaire et suffisant de la condition précédente.
4. (a) Écrire une fonction récursive `scission` prenant en argument un arbre binaire de recherche a et un entier n et renvoyant un couple d'arbres binaires de recherche a_1, a_2 contenant ensemble les mêmes couples (clé,valeur) que a et vérifiant $a_1 \leq n < a_2$.
   (b) Déterminer la complexité de la fonction précédente.
5. En déduire une fonction récursive `fusion` prenant en argument deux arbres binaires de recherche a_1 et a_2 et renvoyant un arbre binaire de recherche contenant l'ensemble de leurs couples (clé,valeur). On supposera que a_1 et a_2 ne contiennent aucune clé en commun.