Implémentation de l'algorithme ID3

On s'intéresse à l'implémentation en OCaml de l'algorithme ID3 dans le cas où tous les attributs sont booléens (même l'attribut cible).

Un exemple d'apprentissage sera donc un tableau de booléens où la case i contient la valeur vrai/faux pour l'attribut numéro i. L'ensemble des exemples d'apprentissage sera une liste de tableaux de booléens.

Les arbres de décision seront codés par le type :

```
type abd =
    | Feuille of bool
    | Noeud of int * abd * abd
;;
```

On suppose déjà écrite certaines fonctions :

```
(* calcule le log en base 2 *)
log2 : float -> float

(* retourne une nouvelle liste où toutes les occurrences de x ont été
 * enlevées *)
enleve : 'a list -> 'a -> 'a list
```

Vous trouverez ces implémentations dans le fichier `id3_eleve.ml` fourni.

Question 1

Écrire une fonction
`classer : abd -> bool array -> bool`
qui prend en entrée un arbre de décision et un exemple non étiqueté et qui retourne l'étiquette prédite.

Question 2

Écrire une fonction
`touslesmemes : bool array list -> int -> bool`
qui prend en entrée une liste d'exemples, et un numéro d'attribut et qui retourne *true* si cet attribut a même valeur pour tous les exemples.

Question 3

Écrire une fonction
`compte_vrai_faux : bool array list -> int -> (int * int)`
prenant en en entrée une liste d'exemples et un numéro d'attribut et qui retourne un couple (nf, nv) où nv est le nombre de fois où l'attribut vaut vrai et nf le nombre de fois où il vaut faux.

Question 4

Écrire une fonction
`majoritaire : bool array list -> int -> bool`
prenant en en entrée une liste d'exemples et un numéro d'attribut et qui retourne cette la valeur majoritaire vrai ou faux observée pour l'attribut donné.

Question 5

Écrire une fonction
`entropie : bool array list -> int -> float`
prenant en entrée une liste d'exemples et un numéro d'attribut cible et qui retourne l'entropie de cet ensemble d'exemples.

Question 6

Écrire une fcontion
`gain : bool array list -> int -> int -> float`
telle que `gain exemples k cible` retourne le gain d'information de l'attribut k par rapport au numéro d'attribut cible.

Question 7

En déduire une fonction
`plus_discriminant : bool array list -> int list -> int -> int`
telle que `plus_discriminant exemples attributs cible` retourne le numéro d'attribut parmi la liste d'attributs proposée qui correspond au plus grand gain d'information.

Question 8

Écrire une fonction `separe` prenant en entrée une liste d'exemples l, un numéro d'attribut k et qui retourne un couple listes `(l1, l2)` où `l1` contient tous les exemples pour lesquels l'attribut `k` vaut vrai et `l2` les autres.

Question 9

Écrire, à l'aide de tout ce qui précède, la fonction récursive :
`id3 : bool array list -> int list -> int -> abd`
construisant un arbre de décision binaire à partir d'une liste d'exemples, d'une liste d'attributs d'apprentissage et d'un attribut cible.