Feuille d'exercices n°13 de l'option d'informatique.

CAML, les piles et les expressions postfixées.

1. Écrire les fonctions **créepile**, **empile** et **dépile** qui permettent de créer une pile, d'ajouter un élément à une pile et de retirer l'élément sur la pile en prenant la définition de pile suivante :

```
type 'a pile = {mutable pile : 'a list},
```

2. Écrire une fonction **vide** qui retourne le booléen **true** si la pile est vide et **false** dans le cas contraire. On définit un ensemble de fonctions et un ensemble d'opérateurs de la manière suivante :

```
type operateur =   add | mult | sous  | div ;;
type fonction = moins | Cos | Sin  | Carré | Rac_carrée;;
```

Les opérateurs `add`, `mult`, `sous`, `div` étant respectivement les opérations d'addition, de multiplication, de soustraction et de division, la fonction `moins` représentant le moins unaire et les fonctions `Cos`, `Sin`, `Carré, Rac_carrée` représentant les fonctions cosinus, sinus, élévation au carré et racine carrée. (Vous pouvez rajouter des fonctions supplémentaires si vous le souhaitez)
On définit ensuite le type `expression`s regroupant les réels et les fonctions sou la forme suivante :

```
type expression =
  reel of float
| op of operateur
| fct of fonction;;
```

Une expression algébrique sera représentée par une liste d'`expression`. Par exemple, l'expression post-fixée `1 cos 3 +` sera représentée par la liste : `[reel 1.;fct Cos; reel 3.;op add]`

3. Écrire une fonction qui évalue une expression postfixée à l'aide d'un pile, en utilisant l'algorithme suivant : on parcourt l'expression post-fixée de gauche à droite, et on effectue les opérations suivantes :

- quand on rencontre un nombre, on l'empile,
- quand on rencontre une fonction f, on dépile le dernier élément x et on empile le résultat de $f(x)$,
- quand on rencontre un opérateur $\otimes$, on dépile deux valeurs y et x et on empile le résultat de $x \otimes y$ (attention aux opérateurs non commutatifs).

En même temps que nous évaluons l'expression nous allons vérifier que c'est bien une expression post-fixée :

- Si au cours du déroulement de l'algorithme précédent la pile se retrouve vide, c'est qu'il a trop d'opérateurs et que l'expression n'est pas correcte,
- S'il reste plus d'un élément dans la pile d'évaluation alors que nous avons parcouru toute l'expression, cela veut dire qu'il y a trop de réels et pas assez d'opérateurs et donc que notre expression post-fixé est incomplète.
- Si en fin de balayage de notre expression, il ne reste plus qu'une valeur dans la pile alors l'expression est bien une expression post-fixée et la dernière valeur de la pile nous donne le résultat de l'évaluation de l'expression post fixée.

Dans le cas où la pile se retrouve vide ou si la pile contient plus d'un élément après avoir balayé l'expression, on retournera un message d'erreur approprié.
L'évaluation de l'expression donnée en exemple devra retourner la valeur de $\cos(1) + 3$.

4. On veut maintenant travailler avec des expressions contenant des variables et déterminer si une expression est bien du type post fixée ou non. Pour cela, on redéfinit le type `expression` sous la forme

```
type expression =
  reel of float
| var of string
| op of operateur
| fct of fonction;;
```

Le but est d'écrire une fonction qui vérifie si une expression est bien du type postfixée. Pour cela, on définit une fonction de poids p sur les expressions par :

- $p(a_i) = 1$ si a_i est une constante ou une variable,
- $p(a_i) = 0$ si a_i est une fonction,
- $p(a_i) = -1$ si a_i est un opérateur binaire.

Nous avons la propriété suivante :

Une expression est postfixée si et seulement si son poids est égal à 1 et le poids de tous ses préfixes sont supérieurs ou égaux à 1.

Écrire une fonction qui retourne le booléen **true** si une expression algébrique est postfixée correcte et **false** sinon en utilisant la propriété ses poids décrite ci-dessus.

5. On définit un type `expr_prefixe` pour représenter des expression préfixée :

```
type expr_prefixe =
  reel_pf of float
| var_pf of string
| Plus of expr_prefixe * expr_prefixe
| Sous of expr_prefixe * expr_prefixe
| Mult of expr_prefixe * expr_prefixe
| Div of expr_prefixe * expr_prefixe
| Sinu of expr_prefixe
| Cosu of expr_prefixe
| Carré of expr_prefixe
| Moins of expr_prefixe
| Racine of expr_prefixe;;
```

Écrire une fonction `prefixe_de_postfixe` qui détermine si une expression algébrique contenant des constantes, des variables, des fonctions et des opérateurs est postfixée et renvoie l'expression préfixée correspondante en utilisant le type `expr_prefixe` donnée dans l'énoncé.
exemple :

```
#prefixe_de_postfixe [reel 1.;fct Cos; reel 3.;op add];;
- : expr_prefixe = Plus (Cosu (reel_pf 1.0), reel_pf 3.0)
```