T.P. B

I bis – Formules arithmétiques

Dans ce TP nous allons interpréter les programmes dans un langage très simple : les expressions arithmétiques. Le but est de lire une expression arithmétique sous forme d'une chaîne de caractères et d'en calculer la valeur.

Nous nous restreindrons au cas de valeurs entières positives (le cas des entiers négatifs est compliqué par le double sens du symbole "-") et des opérations "+", "-", "*" et "/".

Rappels

- Une chaîne de caractères est encadrée par des guillemets doubles : "Bonjour".
- Sa longueur est fournie par String.length ch.
- Le caractère d'indice i d'une chaîne est obtenu par ch.[i].
- Un caractère est encadré par des guillemets simples.
- Char.code permet de renvoyer le code ascii d'un caractère.
- pour calculer le chiffre représenté par un caractère entre '0' et '9', on peut écrire Char.code c − Char.code '0'. '5' donnera le chiffre 5.
- List.rev permet de retourner une liste.

1 Lexer

La première étape consiste à lire la chaîne de caractères et à la séparer en ses objets de bases appelés lexèmes : ce sera les entiers, les opérateurs et les parenthèses.

On utilisera le type suivant :

```
type lexeme =  Nb of int (* les entiers *)
             |Plus |Moins |Fois |Divise (* les opérateurs *)
             |ParO |ParF;; (* les parenthèses *)
```

Question 1.1 **(Solution page 16)**
Écrire une fonction lecture : string −> lexeme list qui reçoit une chaîne de caractères et qui renvoie la liste des lexèmes associée. Les caractères qui ne correspondent pas à un lexème seront ignorés. lecture "5∗(41 + 3) a" doit renvoyer

```
[Nb 5; Fois; ParO; Nb 41; Plus; Nb 3; ParF]
```

2 Notation polonaise inversée

En 1920 le mathématicien polonais Jan Lukasiewicz propose la notation pré-fixé (ou polonaise) qui consiste à considérer les opérations comme des opérateurs à 2 variables, $7 + 11$ s'écrit $+\ 7\ 11$. L'avantage est que les parenthèses deviennent inutiles :+++
$(7 - 2).\sin(2x + \pi/3)$ devient $*\ -\ 7\ 2\ \sin\ +\ *\ 2\ x\ /\ \pi\ 3$.++ La notation polonaise inverse ou notation post-fixée a été proposée par le philosophe et inf
Elle a été diffusée dans le public comme interface utilisateur avec les calculatrices de bureau de Hewlett-Packard (HP-9100), puis avec la calculatrice scientifique HP-35 en 1972.
Elle consiste simplement à placer l'opérateur **après** les deux opérandes, "7 + 11" s'écrit "7 11 +". L'expression "32 − 2∗((7 / 2)∗3 − 12)" peut s'écrire "32 2 7 2 / 3 ∗ 12 − ∗ −"
L'avantage est que, combinée à une pile, cette notation permet d'effectuer les calculs sans faire référence à une quelconque adresse mémoire.

- On lit l'expression terme-à-terme :
- si on lit une valeur numérique, elle est empilée,
- si on lit une opération ♣,
  on dépile les deux derniers opérandes a et b
  et on empile le résultat du calcul a ♣ b.

Un exemple On part de l'expression "32 2 7 2 / 3 ∗ 12 − ∗ −" :

Lecture	Action	Pile			
	On crée une pile vide				
32	On empile	32			
2	On empile	32	2		
7	On empile	32	2	7	
2	On empile	32	2	7	2
/	On dépile 2 éléments	3	2		
	On empile 7/2	32	2	3	
3	On empile	32	2	3	3
∗	On dépile 2 éléments	32	2		
	On empile 3 ∗ 3	32	2	9	
12	On empile	32	2	9	12
-	On dépile 2 éléments	32	2		
	On empile 9 − 12	32	2		-3
∗	On dépile 2 éléments	32			
	On empile 2 ∗ (−3)	32	-6		
−	On dépile 2 éléments				
	On empile 32 − (−6)	38			
	On dépile le résultat : 38				

Question 2.1 (Solution page 16)
Définir un type pile pour une pile **impérative** et écrire les fonctions créerPile, estrPileVide, empiler, voir et depiler. depiler devra enlever l'élément au sommet de la pile **et** le renvoyer en même temps.

Question 2.2 (Solution page 17)
Écrire une fonction calculNPI : string −> int qui calcule la valeur d'une expression **écrite au format NPI** dans une liste de lexèmes et qui calcule sa valeur.

3 Cas général

Le procédé ci-dessus demande d'écrire la formule mathématique en notation N.P.I. ce qui n'est pas pratique. Nous allons maintenant traduire une formule "normale" en une formule N.P.I.
Pour cela on aura à gérer la priorité des opérations :
"3 + 4 ∗ 5" est traduit en "3 4 5 ∗ +" alors que "3 ∗ 4 + 5" est traduit en "3 4 ∗ 5 +"
On donne la priorité 1 aux opérations + et − et la priorité 2 aux opérations ∗ et /.

Pour traduire une formule on effectue les opérations suivantes.

- On lit la formule terme-à-terme :
- on crée une pile vide :
- si on lit une valeur numérique, elle est ajoutée à la sortie,
- si on lit une opération ♣, on dépile les opérations de priorité supérieure ou égale à celle de ♣ et on les ajoute à la sortie, on empile ensuite ♣,
- si on lit une parenthèse ouvrante on l'empile,
- si on lit une parenthèse fermante, on dépile et on ajoute à la sortie toutes les opérations jusqu'à ce qu'on trouve une parenthèse ouvrante, on dépile cette parenthèse ouvrante.
- Quand on a lu tous les lexèmes, on dépile et on ajoute à la sortie toutes les opérations resytantes.
- La sortie est une liste que l'on doit renverser pour obtenir la formule N.P.I.

Un exemple On part de l'expression "2∗(11−2+3∗2)" qui a été transcrite en

```
[Nb 2; Fois; ParO; Nb 11; Moins; Nb 2; Plus;
                                 Nb 3; Fois; Nb 2; ParF]
```

Pour des raisons de place on écrit les lexèmes sous forme "naturelle" dans la pile et la sortie.

Lecture	Pile	Sortie
Nb 2		[2]
Fois	(∗, 2)	[2]
ParO	(∗, 2) (, 0)	[2]
Nb 11	(∗, 2) (, 0)	[11; 2]
Moins	(∗, 2) (, 0) (-, 1)	[11; 2]
Nb 2	(∗, 2) (, 0) (-, 1)	[2; 11; 2]
Plus	(∗, 2) (, 0) (+, 1)	[-; 2; 11; 2]
Nb 3	(∗, 2) (, 0) (+, 1)	[3; -; 2; 11; 2]
Fois	(∗, 2) (, 0) (+, 1) (∗, 2)	[3; -; 2; 11; 2]
Nb 2	(∗, 2) (, 0) (+, 1) (∗, 2)	[2; 3; -; 2; 11; 2]
ParF	(∗, 2)	[+; ∗; 2; 3; -; 2; 11; 2]
		[∗; +; ∗; 2; 3; -; 2; 11; 2]

La notation N.P.I. est donc

```
[Nb 2; Nb 11; Nb 2; Moins; Nb 3; Nb 2; Fois; Plus; Fois]
```

Question 3.1 (Solution page 17)
Écrire une fonction convertirNPI : lexeme list −> lexeme list qui convertit une liste de lexèmes qui exprime une expression sous forme naturelle en une liste de lexèmes qui exprime une expression sous forme N.P.I.

Question 3.2 (Solution page 18)
En déduire une fonction calcul : string −> int qui calcule la valeur d'une expression naturelle dans une chaîne de caractères et qui calcule sa valeur.

4 Solutions

Correction de la question 1.1 page 13

```
let c0 = Char.code '0';;
let add k liste =
    if k = 0
    then liste
    else (Nb k) :: liste;;

let lecture texte =
   let n = String.length texte in
   let rec lire i k fait =
      if i = n
      then add k fait
      else match texte.[i] with
           |c when Char.code c >= c0 && Char.code c <= c0 + 9
                -> lire (i+1) (10*k + Char.code c - c0) fait
           |'+' -> lire (i+1) 0 (Plus   :: (add k fait))
           |'-' -> lire (i+1) 0 (Moins  :: (add k fait))
           |'*' -> lire (i+1) 0 (Fois   :: (add k fait))
           |'/' -> lire (i+1) 0 (Divise :: (add k fait))
           |'(' -> lire (i+1) 0 (ParO   :: (add k fait))
           |')' -> lire (i+1) 0 (ParF   :: (add k fait))
           |_   -> lire (i+1) 0 (add k fait) in
   List.rev (lire 0 0 []);;
```

Correction de la question 2.1 page 14

```
type 'a pile = {contenu = 'a list};;

let creerPile () = {contenu = []};;

let estPileVide pile =
   pile.contenu = [];;

let empiler x pile =
   pile.contenu  <- x :: pile.contenu;;

let depiler pile =
   match pile.contenu with
   |[] -> failwith "La pile est vide"
   |t::q -> pile.contenu <- q;
            t;;

let voir pile = List.hd pile.contenu;;
```

Correction de la question 2.2 page 14

```
let depiler2 pile =
    let b = depiler pile in
    let a = depiler pile in
    a, b;;

let calculNPI liste =
   let pile = creerPile () in
   let rec traiter liste =
      match liste with
      |[] -> depiler pile
      |(Nb k)::q -> empiler k pile;
                    traiter q
      |Plus::q -> let a, b = depiler2 pile in
                  empiler (a+b) pile;
                  traiter q
      |Moins::q -> let a, b = depiler2 pile in
                   empiler (a-b) pile;
                   traiter q
      |Fois::q -> let a, b = depiler2 pile in
                  empiler (a*b) pile;
                  traiter q
      |Divise::q -> let a, b = depiler2 pile in
                    empiler (a/b) pile;
                    traiter q
       |_::q -> traiter q  in
   traiter liste;;
```

Correction de la question 3.1 page 15

```
let sortir prio pile out =
    while not (estPileVide pile) && snd (voir pile) >= prio do
       empiler (fst (depiler pile)) out done;;

let vider pile =
   let rec transfert pile out =
      if estPileVide pile
      then out
      else let x = depiler pile in transfert pile (x::out)
   in transfert pile [];;
```

```
let convertirNPI liste =
   let ope = creerPile () in
   let out = creerPile () in
   let rec traiter liste =
      match liste with
      |(Nb k)::reste -> empiler (Nb k) out;
                        traiter reste
      |Plus::reste -> sortir 1 ope out;
                      empiler (Plus, 1) ope;
                      traiter reste
      |Moins::reste -> sortir 1 ope out;
                       empiler (Moins, 1) ope;
                       traiter reste
      |Fois::reste -> sortir 2 ope out;
                      empiler (Fois, 2) ope;
                      traiter reste
      |Divise::reste -> sortir 2 ope out;
                        empiler (Divise, 1) ope;
                        traiter reste
      |ParO::reste -> empiler (ParO, 0) ope;
                      traiter reste
      |ParF::reste -> sortir 1 ope out;
                      let _ = depiler ope in traiter reste
      |[] -> sortir 1 ope out;
             vider out
   in traiter liste;;
```

Correction de la question 3.2 page 15

```
let calcul ch =
   calculNPI (convertirNPI (lecture ch));;
```