Devoir de compilation	**module LU3IN018**	**Sorbonne Université**
Individuel, à rendre avant le 28.5.20	**COMPILATION**	**Année 2019-2020**

Ce devoir de programmation du cours Compilation va consister à effectuer plusieurs extensions au compilateur du langage microJS et à la machine virtuelle associée. Toutes les ressources se trouvent à l'adresse suivante :
`http://www-licence.ufr-info-p6.jussieu.fr/lmd/licence/2019/ue/LU3IN018-2020fev/`
Les extensions demandées sont décrites par la suite. Il est possible de ne traiter qu'une partie des extensions sachant que la note tiendra compte de l'ensemble des extensions demandées. Il sera demandé de rendre le code, les tests que vous utilisez et un petit rapport expliquant le travail effectué. Les modalités de rendu sont précisées en bas de page.

1. Extensions à réaliser

Pour chaque extension demandée, vous indiquerez bien dans votre rapport où s'effectuent vos différentes modifications. De même pour chaque extension, vous fournirez et décrirez bien les jeux d'essai utilisés pour montrer le bon fonctionnement de votre code.

1.1 Ajout de nouvelles primitives d'entrées-sorties

On cherche ici à ajouter des primitives pour les entrées/sorties.
- primitive **readInt()** : cette primitive ne prend pas d'argument et retourne l'entier lu au clavier, elle retourne la valeur de l'entier saisi ou 0 si ce n'est pas un entier ;
- primitive **print(expr)** : cette primitive d'arité 1 affiche n'importe quelle valeur du langage passée en argument : c'est-à-dire les entiers, les booléens, . . ., pour les valeurs fonctionnelles, elle affiche seulement la chaîne « FUN ».

1.2. Ajout d'une nouvelle structure de données

On cherche ici à étendre les valeurs manipulées par notre microJS : les paires. Les paires possèdent un constructeur **cons(expr1, expr2)** qui prend deux expressions et en construit la paire de ces deux valeurs et deux accesseurs **car(p)** et **cdr(p)** retournant respectivement le premier élément et le deuxième élément de la paire p passée en argument. Ceux-ci vérifient les propriétés suivantes : $car(cons(a, b)) \Rightarrow a$ et $cdr(cons(a, b)) \Rightarrow b$.
Les noms des constructeurs et des accesseurs sont déjà implantés sous-forme de primitives dans la VM.
Pour simplifier l'écriture des paires, on accepte aussi la syntaxe `[e1, e2]` pour $cons(e1, e2)$. Il faudra l'intégrer dans les analyses lexicale et syntaxique. Modifier la primitive **print** pour tenir compte des paires.

Bonus : vérifier que cette nouvelle structure est bien gérée dans le gestionnaire automatique de mémoire de la machine virtuelle, et modifier le si besoin.

1.3. Ajout de multi-déclarations pour simplifier le code

Pour simplifier le code de l'utilisateur, on cherche à pouvoir initialiser plusieurs variables en une instruction de la manière suivante : `var [a,b] = e1` où e_1 est une paire dont le premier champ sera affecté à a et le second à b. Il en sera de même pour les déclarations locales avec `let`.

Bonus : étendre ce mécanisme pour pouvoir nommer des sous-parties de paires, en utilisant le caractère `_` (souligné) quand vous ne désirez pas nommer une sous-partie comme dans : `let [[a, b], [_, d]] = [[33, True], [44,12]]` à la manière du filtrage par motifs d'OCaml.

1.4 Ajout de la valeur `nil` pour la construction de listes

On demande d'ajouter une valeur spécifique, appelée `nil`, permettant de construire des listes à partir des paires. Une liste peut alors être considérée comme :
- soit une liste vide valant la valeur `nil`
- soit une liste non vide dont la tête correspond au premier élément d'une paire et sa suite au deuxième élément de la paire.

La fonction suivante calcule la longueur d'une liste, l'appel de `length(maliste)` retournera 3 :

```
var maliste = cons(1, cons(2, cons(4, nil))) ; // ou var maliste = [1, [2, [ 4, nil]]] ;
function length(l) {if (l == nil) {return 0;} else {return (1 + (length(cdr(l))));}}
length(maliste) ;
```

Bonus : définir les principaux itérateurs sur les listes : **map, filter, reduce** et tester-les.

2. Description du rendu

Il vous est demandé de rendre avant le 28.05.20 une archive contenant le nouveau compilateur, la nouvelle machine virtuelle, vos jeux d'essai et votre rapport. Ce rapport doit présenter comment vous avez réalisé les extensions demandées et comment les avez-vous testées. Il doit faire au maximum une dizaine de pages et être au format PDF. Vous enverrez ce rendu par courrier électronique adressé à `Emmanuel.Chailloux@lip6.fr,choun-tong.lieu@sorbonne-universite.fr`.

N'hésitez pas à nous contacter si vous avez des questions ou si vous avez besoin de précisions sur ce sujet.