Examen de Compilation
Seconde session 2021

Exercice 1. Les questions suivantes sont notées multiplicativement, c'est à dire que que chaque réponse fausse ou laissée vide vous enlèvera 1/4 de vos points :

- Donnez l'ordre d'exécutions des étapes suivantes de compilation : choix des instructions, analyse lexicale, optimisation haut niveau (sur l'AST).
- De quoi LL_1, LR_0, LALR, SLR et LR_1 sont-ils des exemples ? (Plusieurs réponses sont possible, il suffit d'en donner une seule.)
- Écrivez une expression régulière de votre choix utilisant 3 opérateurs différents.
- Qu'est-ce qu'un "stack overflow" ?

Exercice 2. En quelques lignes, Décrivez ce qu'est une continuation.

Exercice 3. Écrivez un fichier de génération de parseur dans le langage de votre choix (lex, ocamllex, javacc, ect...) dont les actions génèrent un AST et qui reconnaisse :

- les expressions composées d'entiers, variables, sommes, produit et appels de fonctions,
- les commandes if_then, while et affectations de variable,
- les déclarations de fonctions.

Il est aussi demandé que l'AST résultant d'une exécution du parseur généré contienne le nombre d'instructions assembleur prévu pour le sous arbre issu du noeud.

On sera très laxiste sur la syntaxe exacte, les bibliothèques et la gestion des erreurs, par contre la structure devra être respectée et les opérateur utilisés doivent exister (avec potentiellement une syntaxe légèrement différente).

Exercice 4. Écrivez un lexeur sous forme d'automate shift-action qui reconnaisse les lexèmes suivants :

- Les Angles en degré minutes et secondes, par exemples `12°15'23"`; attention, on ne peut pas avoir plus de 59 minutes et/ou secondes; de plus on acceptera de ne pas indiquer toutes les informations, ainsi `42°7"` est correcte, de même que `38'`.
- Les Strings délimitées pas des `"`, sans retours à la ligne, et avec les échappements `\"`, `\\` et `\n`,
- Les Entiers,

De plus, les espaces et retours à la ligne sont ajoutés comme séparateurs.
Indiquez les résolution de conflit.
Exécutez le lexeur sur le mot suivant, en utilisant le maximum-munch, et en indiquant les token envoyés (en cas d'erreur indiquez tous ceux envoyés jusque là) :

```
1  57°42’58"
2  57°42’68
3  57°58"42’
4  "57°42’58"12
5  57°42’58"12"
6  67°42’68"12"
```

Exercice 5. Voici un programme dans l'assembleur vu en cours, décompilez-le, c'est à dire retrouvez le programme dont il est issu. Si vous n'arrivez pas à tout traduire, indiquez bien les lignes de début et de fin du code traduit.

```
1  DecVar x
2  DecVar y
3  NewClo 32
4  DecArg x
5  SetVar x
6  CsteNb 4
7  CsteNb 10
8  MultNb
9  SetVar y
10 GetVar y
11 CsteNb 42
12 LeStNb
13 ConJmp 7
14 GetVar x
15 StCall
16 GetVar y
17 SetArg
18 Call
19 SetVar y
20 Jump -11
21 GetVar x
22 StCall
23 GetVar y
24 GetVar x
25 StCall
26 CsteNb 42
27 SetArg
28 Call
29 AddiNb
30 SetArg
31 Call
32 NbToBo
33 ConJmp 1
34 CstNb 42
35 Halt
36 NewClo 5
37 StCall
38 Call
39 GetVar x
40 AddiNb
41 Return
42 GetVar x
43 CsteNb 1
44 AddiNb
45 Return
```

Instruction	sémantique	pile avant	pile après
Log	Print(Pull);	X:pile	X:pile
AddiNb, SubsNb, MultNb, DiviNb	Push(Pop $\odot_f$ Pop); avec $\odot$ représentant, respectivement, $+$, $-$, $*$ et $/$	#n:#n:pile	#n:pile
LoStNb	Push(Pop $<_f$ Pop);	#n:#n:pile	#b:pile
CstNb x	Push(x);	pile	#f:pile
Copy	Push(Pull);	X:pile	X:X:pile
Swap	échange les 2 premières valeurs de la pile	X:Y:pile	Y:X:pile
GetVar n	Push(Get(n))	pile	#v:pile
DclVar n	Insert(n, undefind)	pile	pile
NewClo off	Push(NewCloture{cont := CopyCont, code := PC+off+1})	pile	#l:pile
Jump offset	PC := PC + off + 1;	pile	pile
ConJmp offset	if Pop then PC := PC + 1; else PC := PC + off + 1;	#b:pile	pile
DclArg n	Pull.args.Push(n)	#l:pile	#l:pile
StCal	Pull.setContext(NewContext(CC))	#l:pile	#l:pile
SetArg	v := Pop; clot := Pull; n := clot.args.Pop; clot.cont.Insert(n, v);	#v:#c:pile	#c:pile
Call	clot := Pop Push(NewContinuation{cont := CC, code := PC}) CC := clot.cont; PC := clot.code	#l:pile	#t:pile
Return	res := Pop; continue := Pop; CC := continue.cont; PC := continue.code Push(res)	X:#t:pile	X:pile
Halt	Arrête la machine		