Université Mohammed V – Souissi
École Nationale Supérieure d'Informatique et d'Analyse des Systèmes

Examen

Année Universitaire : 2010 - 2011
Filière : Ingénieur
Semestre : S3
Période : P2

Module : **M3.4** - Compilation
Élément de Module : **M3.4.1** - Compilation
Professeur : Karim BAÏNA

- **Date : 12/01/2011**
- **Durée : 2H00**

Consignes aux élèves ingénieurs :

- *Le barème est donné seulement à titre indicatif !!*
- *Les **réponses directes** et **synthétiques** seront appréciées*
- *Soignez votre **présentation** et **écriture** !!*

Soit la grammaire du langage SQL simplifié :

<SELECT> ::= **select** <PROJECT> <FROM>
<PROJECT> ::= '*****' | <COLUMNS>
<FROM> ::= **from** <TABS> <FROMAUX>
<COLUMNS> ::= <COLUMN> <COLUMNAUX>
<COLUMNAUX> ::= '**,**' <COLUMNS> | **ε**
<COLUMN> ::= **idf** <POINTEDCOLUMN>
<POINTEDCOLUMN> ::= '**.**' **idf** | **ε**
<TABS> ::= **idf** | **idf** ',' <TABS>
<FROMAUX> ::= <WHERE> ';' | <ORDERBY> ';' | ';'
<WHERE> ::= **where** <EXPBOOL>
<EXPBOOL> ::= **not** <EXPBOOL>
| <EXPBOOL> **and** <EXPBOOL>
| <EXPBOOL> **or** <EXPBOOL>
| <COLUMN> <OP> <COLUMN>
<OP> ::= **lower** | **loweroreq** | **greater** | **greateroreq** | **eq** | **neq**
<ORDERBY> ::= **orderby** <COLUMNS>

1) Désambiguïser la grammaire en réécrivant les règles correspondant au non-terminal ou aux non-terminaux ambigus avec les priorités habituelles (2pts)

<EXPBOOL> ::= <OR>
<OR> ::= .....
<NOT> ::= .....
<AND> ::= .....
<AUX> ::= <COLUMN> <OP> <COLUMN>

2) Éliminer la récursivité à gauche (ne donner que les nouvelles règles) (2pts)

<OR> ::= .....
<ORAUX> ::= .....
<NOT> ::= .....
<NOTAUX> ::= .....
<AND> ::= .....
<ANDAUX> ::= .....

3) Rendre la grammaire LL(1) (ne donner que les nouvelles règles) (2pts)

<TABS> ::= .....
<TABSAUX> ::= .....

4) Calculer les directifs First et Follow des NT nullables (2pts)

Non-terminal	Les premiers (*First*)	Les suivants (*Follow*)
<COLUMNAUX>	.....	.....
<POINTEDCOLUMN>	.....	.....
<OR>	.....	.....
<NOTAUX>	.....	.....
<ANDAUX>	.....	.....
<TABSAUX>	.....	.....

5) Programmer en C le prédicat pointedcolumn faisant partie de l'analyseur syntaxique LL(1) (2pts)

.....
.....
.....
.....
.....
.....
.....
.....

6) Lister 4 erreurs sémantiques possibles (2pts)

erreur 1 - .....
erreur 2 - .....
erreur 3 - .....
erreur 4 - .....

7) Questions sur le code étudié en TP : (8pts)

7.1) Soit le type INST vu en TP, améliorer le pour prendre en compte l'instruction for. (2pts)

```
typedef struct INST {

  Type_INST typeinst;                    .....
  union  {                               .....
    // PRINT idftoprint                  .....
    struct  {                            .....
      int rangvar;                       .....
    } printnode;                         .....
    // left := right                     .....
    struct  {                            .....
      int rangvar;                       .....
      AST right;                         .....
    } assignnode;                        .....
    // IF ... THEN                       .....
    struct  {                            .....
      int rangvar;
      AST right;
      struct LIST_INST * thenlinst;
      struct LIST_INST * elselinst;
    } ifnode;
  } node;
} instvalueType;
```

7.2) Qu'est ce qui joue le rôle du tas dans la programmation de la mémoire virtuelle étudiée en TP ? **(2pts)**
.....
.....

7.3) Qu'est ce qui joue le rôle de la mémoire statique dans la programmation de cette mémoire virtuelle ? **(2pts)**
.....
.....

7.4) Donner deux limitations à la fonction interpreter_pseudo_code vue en TP (en justifiant) : **(2pts)**

```
void interpreter_pseudo_code(pseudocode pc){
 char ** next_label_name = (char **) malloc(sizeof (char*));
 if (pc != NULL){
    interpreter_pseudo_instruction(pc->first, next_label_name);
    if (*next_label_name == NULL) interpreter_pseudo_code(pc->next); // Il n y a pas de branchement !!
    else{ // JNE ou JMP ==> effectuer un branchement
            struct pseudocodenode * compteur_ordinal = pc->next;
            while ( (compteur_ordinal->first.codop != LABEL) ||
                        (strcmp(compteur_ordinal->first.param.label_name, *next_label_name) != 0) ) {
                        // (compteur_ordinal ne peut jamais == NULL) après JMP/JNE dans le code (par construction)
                        compteur_ordinal = compteur_ordinal->next;
            }
            interpreter_pseudo_code(compteur_ordinal); // branchement
    }
 }
}
```

limitation 1 – .....
.....

limitation 2 – .....
.....