MALLÉJAC CLÉMENT
Note: 5.8/20 (score total : 17.75/60.75)

+20/1/47+

UE INF1601 **2019**

Théorie des langages et compilation
Contrôle continu numéro 3
(45 minutes)

Nom et prénom : Mallejac Clément

Noircissez les bonnes réponses (cocher ne suffit pas). Les questions faisant apparaître le symbole ♣ peuvent présenter une ou plusieurs bonnes réponses ; les autres ont une seule bonne réponse. Toute absence de réponse équivaut à une réponse fausse. Utilisez le verso des feuilles comme brouillon si nécessaire.

Compilateur

Question 1 La phase d'analyse reconnaît qu'une chaîne de caractères est la description correcte d'un programme.

1/1

- [x] vrai
- [ ] faux

Question 2 ♣ La phase d'analyse comporte les étapes suivantes :

1.25/1.25

- [x] l'analyse lexicale
- [ ] l'analyse de code
- [ ] la génération de code intermédiaire
- [x] l'analyse sémantique
- [x] l'analyse syntaxique

Question 3 La phase d'analyse est dépendante du langage cible

1/1

- [x] faux
- [ ] vrai

Question 4 La phase d'analyse est dépendante du langage source

1/1

- [ ] faux
- [x] vrai

Question 5 ♣ La phase d'analyse produit

1/1

- [ ] un arbre de dérivation
- [x] une table des symbole
- [ ] un programme cible
- [x] un arbre de syntaxe abstrait

Question 6 ♣ La phase de synthèse produit

0.75/0.75

- [ ] un arbre de dérivation
- [x] un programme cible
- [ ] un arbre de syntaxe abstrait

Question 7 Le compilateur PTS produit du code en langage d'assemblage.

1/1

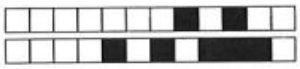
- [ ] faux
- [x] vrai

Analyseur lexical

Question 8 L'analyseur lexical prend en entrée une liste d'expressions régulières.

-1/1

- [x] faux
- [x] vrai



+20/2/46+

Question 9 L'analyseur lexical ne tient pas compte des commentaires.

-1/1 ☒ faux ● vrai

Question 10 ♣ l'analyseur lexical produit :

1/1 ■ des unités lexicales ☐ une chaîne de caractères
☐ un arbre de dérivation ☐ un arbre de syntaxe abstraite

Question 11 Un lexème est une unité lexicale correspondant à un identificateur.

-1/1 ● vrai ☒ faux

Question 12 Le description d'une unité lexicale est un modèle.

-1/1 ☒ vrai ● faux

Question 13 Un modèle peut être donné sous la forme d'une expression régulière.

1/1 ☐ faux ■ vrai

Question 14 L'analyseur lexical fourni un lexème/l'unité lexicale correspondante à un identificateur.

-1/1 ☒ faux ● vrai

Question 15 L'analyseur lexical est appelé par l'analyseur syntaxique.

1/1 ■ vrai ☐ faux

Question 16 Un analyseur lexical met en œuvre un automate à pile.

-1/1 ● vrai ☒ faux

Question 17 Un générateur d'analyseur lexical prend en entrée un ensemble d'expressions régulières.

1/1 ■ vrai ☐ faux

Question 18 Un générateur d'analyseur lexical produit une liste d'unités lexicales.

1/1 ■ faux ☐ vrai

Question 19 JFlex permet de récupérer un lexème de l'analyseur lexical `scan` par :

1.5/1.5 ☐ `scan.yylex()` ■ `scan.yytext()` ☐ `scan.yyline`

Question 20 ♣ Le fichier de spécification d'un générateur d'analyseur lexical contient :

1/1 ■ des expressions régulières ■ des actions
☐ des règles de grammaire ■ des modèles d'unités lexicales

Question 21 Un makefile est un outil qui compile des analyseurs lexicaux.

1/1 ■ faux ☐ vrai

Question 22 Dans le compilateur PTS, l'analyseur lexical est mis en œuvre avec JFlex.

-1/1

☒ faux ● vrai

Analyseur syntaxique

Question 23 L'analyseur syntaxique prend en entrée une liste d'unités lexicales.

1/1

☐ faux ■ vrai

Question 24 ♣ L'analyseur syntaxique produit :

1/1

☐ un arbre de dérivation
☐ une liste d'unités syntaxiques
■ une table des symboles
■ un arbre de syntaxe abstraite

Question 25 ♣ un AST est :

0/1

☐ un arbre de syntaxe concrète
■ un arbre de syntaxe abstraite
☐ un arbre de dérivation
☒ un arbre de dérivation simplifié

Question 26
On considère ci-dessous, à gauche une grammaire (abrégée pour la partie des expressions arithmétiques) et à droite un programme respectant cette grammaire. Dessinez un AST qui pourrait être produit par un analyseur syntaxique pour ce programme.

$Stat \rightarrow IfStat | Aff | Iter$
$IfStat \rightarrow$ **if** $Cond$ **then** $Stat$ **else** $Stat$
$Iter \rightarrow$ **while** $Cond$ **do** $Stat$ **od**
$Aff \rightarrow Id := Expr;$
$Cond \rightarrow Expr\ relop\ Expr$
$Expr \rightarrow \ldots$

```
while (x < n) do
 if (i == 0) then z := 1; else z := 2;
od
```

0/2.5

☐A ☐B ☐C ☐D ☐E ■F *Réservé au correcteur : ne pas cocher !*

Question 27 Un analyseur syntaxique met en œuvre un automate à pile.

1/1

■ vrai ☐ faux

Question 28 La méthode de descente récursive est une mise en œuvre d'un analyseur prédictif itératif.

-1/1

☒ faux ● vrai

Question 29 Un générateur d'analyseur syntaxique produit une table d'analyse prédictive.

-1/1 ☒ faux ● vrai

Question 30 Un générateur d'analyseur syntaxique prend en entrée un fichier de spécification contenant une grammaire et des actions exprimées dans un langage de programmation.

1/1 ■ vrai ☐ faux

Question 31 JavaCC produit un analyseur prédictif récursif.

1/1 ☐ faux ■ vrai

Question 32 JavaCC nécessite une grammaire qui soit LL(k).

-1/1 ● faux ☒ vrai

Question 33 La grammaire de PTS est LL(1).

-1/1 ● vrai ☒ faux

Question 34
Complétez l'algorithme de l'analyseur prédictif itératif suivant (on notera Δ la table d'analyse prédictive et \$ le symbole de fin de mot) :

```
var V_T  symb := nextSymb(w);
empiler(S);
repeter
  top := sommetPile();




tant que non pileVide();
```

☐A ☐B ☐C ☐D ☐E ■F *Réservé au correcteur : ne pas cocher !*

Question 35 L'analyseur syntaxique du compilateur de PTS est mis en œuvre avec JavaCC.

1/1 ☐ faux ■ vrai

Analyseur sémantique

Question 36 ♣ L'analyseur sémantique vérifie :

1.5/1.5
- ■ le typage
- ■ le flot d'exécution
- ■ l'utilisation des identificateurs
- ☐ la syntaxe des identificateurs
- ■ la déclaration des identificateurs
- ☐ la syntaxe des déclarations

Question 37 ♣ L'analyseur sémantique prend en entrée :

0/1

- [x] une table des symboles
- [ ] un arbre de dérivation
- [x] un arbre de syntaxe abstraite
- [ ] une liste de déclaration

Question 38 ♣ L'analyseur sémantique produit :

-2/1

- [x] une liste de déclaration
- [x] un arbre de syntaxe abstraite
- [x] une table des symboles
- [ ] un arbre de dérivation

Question 39 La valeur gauche d'un identificateur désigne son emplacement.

1/1

- [ ] faux
- [x] vrai

Question 40 La valeur droite d'un identificateur désigne sa valeur.

1/1

- [x] vrai
- [ ] faux

Question 41 En Java il n'est pas possible de redéfinir une variable de même nom dans un bloc imbriqué.

1/1

- [ ] faux
- [x] vrai

Question 42 En PTS il n'est pas possible de redéfinir une variable de même nom dans un bloc juxtaposé.

-1/1

- [x] faux
- [x] vrai

Question 43 Le langage Java est un langage à portée dynamique.

1/1

- [x] faux
- [ ] vrai

Question 44 Un attribut hérité est une valeur communiquée par un nœud à ses fils.

-1/1

- [x] vrai
- [x] faux

Question 45 Un attribut synthétisé est une valeur communiquée par un nœud à son père.

1/1

- [x] vrai
- [ ] faux

Question 46 La vérification de type d'un langage dans lequel toutes les déclarations doivent précéder leurs utilisations nécessite un seul parcours de l'AST.

1/1

- [x] vrai
- [ ] faux

Question 47 Les informations de type des identificateurs sont mémorisées dans la table des symboles.

1/1

- [x] vrai
- [ ] faux

Question 48 Dans le compilateur de PTS, l'analyseur sémantique procède en un seul parcours de l'AST.

-1/1

- [x] faux
- [x] vrai

Question 49 Compléter la grammaire d'attribut suivante pour la construction de l'AST d'une expression arithmétique.

$S \to E$	$S.noeud = E.noeud;$
$E \to E_1 + T$	
$E \to T$	
$T \to T_1 \times F$	
$T \to F$	
$F \to (E)$	
$F \to const$	

0/2.5 ☐A ☐B ☐C ☐D ☐E ■F *Réservé au correcteur : ne pas cocher !*

Générateur de code

Question 50 L'entrée du générateur de code intermédiaire est une liste d'instructions.

1/1 ■ faux ☐ vrai

Question 51 Quel est le code intermédiaire généré pour l'expression $x1 := (-b + rac)/(2 * a)$ dans le cas où les variables temporaires renvoyées par $newTmp()$ sont réutilisables ? on considérera que les variables a, b sont de type entier et $x1, rac$ de type réel.

$t_0 = -b$
$t_1 = int(rac)$
$t_0 += t_1$
$t_1 = 2 * a$
$t_2 = t_0$
$t_0 /= t_1$
$t_2 = float(t_0)$

1/2.5 ☐A ☐B ☐C ■D ☐E ☐F *Réservé au correcteur : ne pas cocher !*

Question 52 Le code court-circuit généré pour une expression booléenne permet de n'évaluer que la partie droite ou bien que la partie gauche de l'expression.

-1/1 ☒ faux ● vrai

Question 53 Donnez la grammaire d'attributs permettant de générer le code de l'instruction suivante en considérant que les expressions booléennes sont évaluées avec un code court-circuit.
$I \rightarrow$ **repeter** I_1 **tantque** E.

☐A ☐B ☐C ☐D ☐E ☒F *Réservé au correcteur : ne pas cocher !*

Optimiseur de code

Question 54 L'entrée de l'optimiseur de code intermédiaire est une liste d'instructions.

1/1

☐ faux ☒ vrai

Question 55 Décrivez en quelques items le principe d'un optimiseur à lucarne.

Regarder le programme avec une fenêtre de n instructions
Tant que l'on reconnait des fenêtre d'instruction optimisable :
Appliquer le changement et recommencer la lecture avec la même taille de fenêtre.
Si il n'y a plus de changements après une lecture :
Agrandir la fenêtre ou arreter l'optimiseur.

2/2.5

☐A ☒B ☐C ☐D ☐E ☐F *Réservé au correcteur : ne pas cocher !*

Question 56 ♣ Quelles sont les optimisations caractéristiques de cette technique ?

-0.25/0.75

☒ élimination des instructions inaccessibles
☒ simplification du flot de contrôle
☒ simplifications algébriques