Examen de rattrapage

Année Universitaire : 2008 - 2009
Filière : Ingénieur
Semestre : S3
Période : P2

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

Date : 20/03/2009
Durée : 1H00

Nom :……
Prénom :……

Consignes aux élèves ingénieurs :
- *Aucun document n'est autorisé !!*
- *Le barème est donné seulement à titre indicatif !!*

Exercice I : Choix Exclusifs (10 pts)

Pour chaque concept/question, remplissez la case de la colonne des choix uniques correspondante par un choix qui soit le plus adéquat

Concept/Question	Choix unique	Choix possibles
(**1**) Toute grammaire peut être rendue LL(1)		**(a)** Vrai **(b)** Faux
(**2**) Flex peut totalement remplace Bison pour quelques langages particuliers		**(a)** Vrai **(b)** Faux
(**3**) Bison permet de programmer les grammaires attribuées avec héritage et synthèse d'attributs		**(a)** Vrai **(b)** Faux
(**4**) <**INST**> ::= IDF ":=" <**EXPR**> | IF '(' IDF '=' <**EXPR**> ')' THEN <**LISTE_INST**> ELSE <**LISTE_INST**> ENDIF | IF '(' IDF '=' <**EXPR**> ')' THEN <**LISTE_INST**> ENDIF | PRINT IDF ;		**(a)** est Ambiguë **(b)** n'est pas ambiguë
(**5**) <**ADDMIN**> ::= <**ADDMIN**> '+' IDF | <**ADDMIN**> '-' IDF | IDF		**(a)** est LL(1) **(b)** n'est pas LL(1)
(**6**) le 1-address code est choisi pour sa		**(a)**rapidité **(b)**taille du code **(c)**portabilité
(**7**) la fonction de hashage $h_1(s \in \Sigma^*) = \Sigma_{i=1..\|s\|}\ s_i$ répartit …… les identifiants s que la fonction $h_1(s \in \Sigma^*) = \Sigma_{i=1..\|s\|}\ s_i$ dans la table des symboles		**(a)** mieux **(b)** moins bien **(c)** similairement
(**8**) un automate NFA est analogue à une grammaire		**(a)** ambiguë **(b)** avec des règle à ε **(c)** non LL(1)
(**9**) Le langage $L = \{a^n b^n c^m d^m, n \geq 1,\ m \geq 1\} \cup \{a^n b^m c^m d^n, n \geq 1,\ m \geq 1\}$ est programmable en bison		**(a)** oui **(b)** non **(c)** avec des adaptations
(**10**) La grammaire <**INST1**> ::= IF '('<**EXPR**>')' THEN <**INST1**> ELSE <**INST1**> | IF '('<**EXPR**>**')' THEN** <**INST1**> et la grammaire <**INST2**> ::= IF '('<**EXPR**>')' THEN <**INST2**> ELSE <**INST**> ENDIF | IF '('<**EXPR**>')' THEN <**INST2**> ENDIF donne ….		**(a)** le même langage **(b)** différents langages

#define N 200
#define NBS 100
int NBVAR=0;
typedef enum {false=0, true=1} boolean;
typedef struct {
char *name; int nbdecl; int order;
} varvalueType;
varvalueType TS[NBS];
 	
Compléter La fonction de recherche d'un identifiant dans la tableau des symboles
boolean inTS(char * varname, int * rangvar){
int i =0;
(**11**) while ((i < …….)
&& (strcmp(TS[i].name, varname) != 0)) i++;
		(a) sizeof(TS) **(b)** N **(c)** NBS **(d)** NBVAR

(**12**) if (i == …..) return false;		**(a)** sizeof(TS) **(b)** N **(c)** NBS **(d)** NBVAR
(**13**) else { ……. = i; return true;} }		**(a)** TS[i].order **(b)** rangvar **(c)** &rangvar **(d)** *rangvar

(**14**) Que réalise la fonction process sur les mots du langage des alpha-numériques
typedef char * langage1 ;
void process(langage1 s){
int c, i, j;
for (i = 0, j = strlen(s)-1; i < j; i++, j--) {
c = s[i]; s[i] = s[j]; s[j] = c;
}
}
		(a) décale les mots à droite
		
		(b) décale les mots à gauche
		
		(c) renverse les mots
		
		(d) tranforme le mot en son palyndrôme

(**15**) Que réalise la fonction apply sur le langage des numériques
typedef int langage2;
langage1 apply (langage2 X){
int i = 0;
char langange1 s[100];
langage1 result;
do s[i++] = X % 10 + '0'; while ((X /= 10) > 0);
s[i] = '\0';
process(s);
result = (langage1) malloc(strlen(s) + 1);
strcpy(result, s);
return result;
}
		(a) calcule la chaîne décimale du mot binaire
		
		(b) calcule la chaîne binaire du mot décimale
		
		(c) renvoie l'image numérique du texte représentant le mot
		
		(d) renvoie l'image textuelle du mot

Exercice I : Réseau de concepts (10 pts)

Pour chaque concept/question, remplissez la case de la colonne des choix uniques correspondante par un choix qui soit le plus adéquat

Concept/Question	Choix unique	Choix possibles
(**1**) Bytecode Java	(**B**)	(**A**) démontrer qu' « une grammaire est ambiguë » est décidable mais l'inverse est non décidable
(**2**) ADDOP REG1, REG2		(**B**) Représentation
(**3**) Nombre de registres nécessaire pour un expression arithmétique		(**C**) Représentation
(**4**) Acorn RISC Machine-ARM		(**D**) Erreur Lexicale
(**5**) AST	(**C**)	(**E**) Attribut nécessaire à la génération de pseudo-code
(**6**) Commentaire C non fermé (/* sans */)		(**F**) two-address code
(**7**) Grammaire Ambiguë		(**G**) three-address code
(**8**) Récursivité Gauche		(**H**) Tri et tri inverse des feuilles
(**9**) Grammaire non LL		(**I**) Analyse floue
(**10**) Grammaire algébrique		(**J**) Analyse descendante non optimale
(**11**) *(null).suivant		(**K**)Erreur Syntaxique
(**12**) Select Pilote.Nom from * ;		(**L**) Analyse sans fin
(**13**) (n{m)n}m		(**M**) Analyse ascendante
(**14**) Grammaire LR		(**N**) Erreur Sémantique
(**15**) Dérivation droite et gauche		(**O**) Analyse impossible
(**16**) semi-décidabilité	(**A**) résolue	(**P**) Equation linéaire