Login: Nom:

CCMP2 – Construction des Compilateurs, partie 2

PARTIEL_CCMP2_ING1_JUIN_2011

EPITA – Promo 2013
Avec formulaire de QCM, sans documents ni machine

Juin 2011 (1h30)

Bien lire les questions, chaque mot est important.

Répondre sur les formulaires de QCM ; aucune réponse manuscrite ne sera corrigée. Renseigner les champs d'identité. Il y a exactement une et une seule réponse juste pour ces questions. Si plusieurs réponses sont valides, sélectionner la plus restrictive. Par exemple s'il est demandé si 0 est *nul*, *non nul*, *positif*, ou *négatif*, cocher *nul* qui est plus restrictif que *positif* et *négatif*, tous deux vrais. Répondre incorrectement est plus pénalisé que de ne pas répondre.

1 Incontournables

Chaque erreur ou non réponse aux trois questions suivantes retire 1/6 de la note finale. Avoir tout faux divise donc la note par 2.

Q.1 `sizeof` est une fonction de la bibliothèque standard du langage C.

a. vrai b. faux

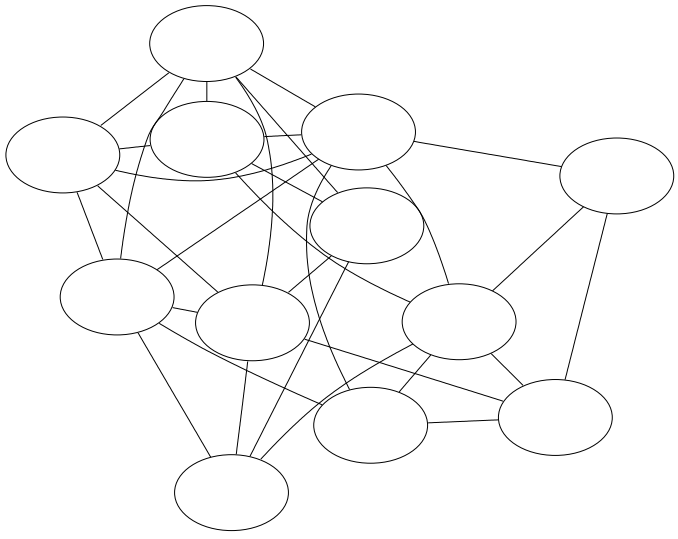
Q.2 Tout graphe peut-être coloré en quatre couleurs.

a. vrai b. faux

Q.3 Boost est. . .

a. Un ensemble de bibliothèques C++ b. Une technique d'optimisation du C++

2 Construction des Compilateurs : Allocation des Registres



Colorer ce graphe d'exclusion mutuelle en 4 registres.

Annoter chaque nœud d'un 1, 2, 3 ou 4 et rendre le sujet.

Donner votre identité en haut.

3 Contrôle

Q.4 Le mot-clé `auto` est très ancien (dès le C Kernighan et Ritchie) mais particulièrement peu utile : comme `static` ou `register` il exprime le type de « stockage » : la pile (ou un register si possible) pour le cas d'`auto`. Mais comme il est la classe de stockage par défaut, peu de programmeurs (aucun ?) prenaient la peine d'écrire `auto int i` là où `int i` signifie la même chose.

La nouvelle norme du C++ introduit un nouvel usage pour le mot-clé `auto` : autoriser l'« inférence de type » pour les déclarations de variables (c'est en quelque sorte le type qui devient optionnel). Ainsi, au lieu de

```
std::pair< std::pair < int, int >, int>::first_type::second_type two =
  std::make_pair(std::make_pair(1, 2), 3).first.second;
```

on peut écrire

```
auto two = std::make_pair(std::make_pair(1, 2), 3).first.second;
```

Cette addition

a. est un ajout trivial dans un compilateur C++-98
b. pose des problèmes nouveaux au scanner du compilateur C++
c. complexifie notablement le front-end du compilateur C++
d. complexifie notablement le back-end du compilateur C++
e. complexifie notablement les front- et back-end du compilateur C++

Q.5 Si une variable locale est passée par référence à d'autres fonctions, elle résidera (s'il y reste de la place)...

a. dans un registre b. sur la pile c. sur le tas d. dans le swap

Q.6 Si une variable locale a des utilisations non-locales, elle résidera (s'il y reste de la place)...

a. dans un registre b. sur la pile c. sur le tas d. dans le swap

Q.7 Si une variable locale C++ contient une structure, elle résidera (s'il y reste de la place)...

a. dans un registre b. sur la pile c. sur le tas d. dans le swap

Q.8 Les variables non-locales sont :

a. des variables définies dans `_main` (variables globales)
b. des variables locales d'une fonction `f` passée par référence à une autre fonction `g`
c. des variables de sous-fonctions (i.e., des fonctions définies syntaxiquement à l'intérieur d'une autre fonction) utilisées par la fonction englobante
d. des variables des fonctions englobantes utilisées par des sous-fonctions
e. des variables d'un autre thread

Q.9 Le support des variables non-locales est typique des langages

a. fonctionnels b. objets c. impératifs d. parallèles e. distribués

Q.10 Dans la gestion des variables non-locales par static-link, le nombre de sauts (déréférencement) des static-links est :

a. n, la profondeur d'imbrication syntaxique des fonctions
b. m, la profondeur des activations de fonctions (i.e., le nombre de cadres de pile qui nous séparent du cadre hébergeant la variable)
c. $n + m$
d. m/n
e. 1

Pour chacun des programmes HIR ci-dessous, sélectionnez le programme « canonisé » (i.e., LIR) qui lui correspond. Par convention les termes '`s*`' désignent des instructions (*statement*), '`e*`' des expressions, '`l*`' des étiquettes (*label*) et '`t*`' des temporaires. Nous utilisons les notations du projet Tiger de l'EPITA : `sxp` au lieu de `exp` chez Andrew Appel, `seq` avec un nombre d'éléments arbitraires, etc.

Q.11 `seq (s1, seq (s2, s3), s4)`

a. `seq (s1, s2, s3, s4)`
b. `seq (seq (se q(s1, s2), s3), s4)`
c. `eseq (seq (s1, s2, s3), s4)`
d. `seq (s1, s2, s4)`

Q.12 `mem (eseq (s, e))`

a. `eseq (move (temp t, e), mem (e))`
b. `seq (s, sxp (mem (e)))`
c. `eseq (s, mem (e))`
d. `seq (s, sxp (e))`

Q.13 `sxp (eseq (s, e))`

a. `eseq (s, e)`
b. `seq (s, sxp (e))`
c. `seq (s, e)`
d. `seq (sxp (s), sxp (e))`

Q.14 `sxp (call (eseq (s, temp t1), temp t2))`

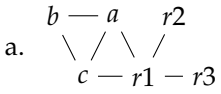
a. `sxp (s, call (temp t1, temp t2))`
b. `sxp (eseq (s, call (temp t1, temp t2)))`
c. `seq (s, sxp (call (temp t1, temp t2)))`
d. `seq (s, call (temp t1, temp t2))`

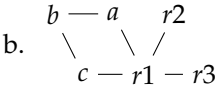
Q.15 `sxp (call (label l, e1, eseq (s, temp t))`

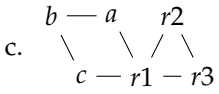
a. `seq (s, move (temp t1, e1), sxp (call (label l, temp t1, temp t)))`
b. `seq (s, sxp (call (label l1, e1, temp t)))`
c. `seq (move (temp t1, e1), s, sxp (call (label l, temp t1, temp t)))`
d. `seq (move (temp t1, e1), move (temp t2, s), sxp (call (label l, temp t1, temp t)))`

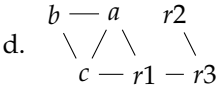
Q.16 Quel est le graphe d'interférence du programme suivant ?

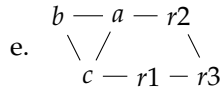
```
r1 := b * b
r2 := a * c
r3 := 4 * r2
r4 := r1 - r3
```

a. 

b. 

c. 

d. 

e. 

Combien suffit-il d'avoir de registres pour ne pas avoir besoin de la pile avec les graphes d'interférence suivants ?

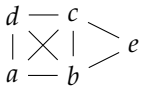
Q.17  a. 2 b. 3 c. 4 d. 5 e. 6

Q.18  a. 2 b. 3 c. 4 d. 5 e. 6

Q.19  a. 2 b. 3 c. 4 d. 5 e. 6

Q.20  a. 2 b. 3 c. 4 d. 5 e. 6

Q.21  a. 2 b. 3 c. 4 d. 5 e. 6

Q.22  a. 2 b. 3 c. 4 d. 5 e. 6

4 À propos de ce cours

Pour terminer cette épreuve, nous vous invitons à répondre à un petit questionnaire. Les renseignements ci-dessous ne seront bien entendu pas utilisés pour noter votre copie. Ils ne sont pas anonymes, car nous souhaitons pouvoir confronter réponses et notes. En échange, quelques points seront attribués pour avoir répondu. Merci d'avance.

Sauf indication contraire, vous pouvez cocher plusieurs réponses par question. Répondez sur la feuille de QCM. N'y passez pas plus de dix minutes.

Cette épreuve

Q.23 Sans compter le temps mis pour remplir ce questionnaire, combien de temps ce partiel vous a-t-il demandé (si vous avez terminé dans les temps), ou combien vous aurait-il demandé (si vous aviez eu un peu plus de temps pour terminer) ?

a. Moins de 30 minutes.
b. Entre 30 et 60 minutes.
c. Entre 60 et 90 minutes.
d. Entre 90 et 120 minutes.
e. Plus de 120 minutes.

Q.24 Ce partiel vous a paru

a. Trop difficile.
b. Assez difficile.
c. D'une difficulté normale.
d. Assez facile.
e. Trop facile.

Le projet Tiger

Q.25 Vous avez contribué au développement du compilateur de votre groupe (une seule réponse attendue) :

a. Presque jamais.
b. Moins que les autres.
c. Équitablement avec vos pairs.
d. Plus que les autres.
e. Pratiquement seul.

Q.26 La charge générale du projet Tiger est

a. Telle que je n'ai pas pu suivre du tout.
b. Lourde (plusieurs jours de travail par semaine).
c. Supportable (plusieurs heures de travail par semaine).
d. Légère (une ou deux heures par semaine).
e. J'ai été dispensé du projet.

Q.27 Y a-t-il de la triche dans le projet Tiger ? (Une seule réponse attendue.)

a. Pas à votre connaissance.
b. Vous connaissez un ou deux groupes concernés.
c. Quelques groupes.
d. Dans la plupart des groupes.
e. Dans tous les groupes.

Questions 28-34 Le projet Tiger vous a-t-il bien formé aux sujets suivants ? Répondre selon la grille qui suit. (Une seule réponse attendue par question.)

a Pas du tout **b** Trop peu **c** Correctement **d** Bien **e** Très bien

Q.28 C++.

Q.29 modélisation orientée objet et *design patterns*.

Q.30 anglais technique.

Q.31 compréhension du fonctionnement des ordinateurs.

Q.32 compréhension du fonctionnement des langages de programmation.

Q.33 travail collaboratif.

Q.34 outils de développement (contrôle de version, systèmes de construction, débogueurs, générateurs de code, etc.)

Questions 35-50 Comment furent les étapes du projet ? Répondre selon la grille suivante. (Une seule réponse attendue par question ; ne pas répondre pour les étapes que vous n'avez pas faites.)

a Trop facile. **b** Facile. **c** Nickel. **d** Difficile. **e** Trop difficile.

Q.35 Rush `.tig` : mini-projet en Tiger (Bistromatig).

Q.36 `TC-0`, Scanner & Parser.

Q.37 `TC-1`, Scanner & Parser, Tâches, Autotools.

Q.38 `TC-2`, Construction de l'AST et pretty-printer.

Q.39 `TC-3`, Liaison des noms et renommage.

Q.40 `TC-4`, Typage et calcul des échappements.

Q.41 `TC-5`, Traduction vers représentation intermédiaire.

Q.42 `TC-6`, Simplification de la représentation intermédiaire.

Q.43 `TC-7`, Sélection des instructions.

Q.44 `TC-8`, Analyse du flot de contrôle.

Q.45 `TC-9`, Allocation de registres.

Q.46 `TC-D (option)`, Désucrage (boucles `for`, comparaisons de chaînes de caractères).

Q.47 `TC-I (option)`, Mise en ligne du corps des fonctions.

Q.48 `TC-B (option)`, Vérification dynamique des bornes de tableaux.

Q.49 `TC-A (option)`, Surcharge des fonctions.

Q.50 `TC-O (option)`, Désucrage des constructions objets.