Université du Maine
Faculté des Sciences et techniques

Jeudi 10 mai 2012
13h30-15h30

Langage et compilation(2)
Examen première session

Barème: 20 points
Documents: Tous supports électroniques et tous documents interdits en dehors du présent sujet
Durée: 2 h

1 Analyse lexicale

1.1. Donnez une chaîne lexicale qui décrit l'instruction d'assertion, commentaire compris, qui figure dans la fonction ci-dessous (cette ligne contient des erreurs, mais tous les programmeurs en font).
1.2. Donnez un AEF opérationnel qui reconnaît un identificateur en langage C, ou l'un des mots-clefs cités dans l'exemple ci-dessous (il y en a trois, à savoir « int », « static », et « return »).
1.3. Donnez 3 messages d'erreur relatifs à l'analyse lexicale des identificateurs et mots-clefs du 1.2.
1.4. Tabuler l'AEF opérationnel qui reconnaît identificateur et mots-clefs et qui détecte les 3 types d'erreurs du 1.3.

2 Analyse syntaxique

2.1. Donnez une grammaire algébrique G1 qui décrit la structure de la fonction ci-dessous, en ignorant toutefois les lignes 3, 5 et 7, nettement plus complexes.
2.2. Donnez trois messages d'erreurs syntaxiques pertinents pour l'exemple ci-dessous.

3 Analyse sémantique

3.1. Donner trois variantes de cette fonction qui mettent en évidence au moins trois types d'erreurs de nature sémantique, erreurs cantonnées à l'en-tête de routine et/ou au retour de fonction.
3.2. Ajouter à G1 des points de génération et des attributs qui contrôlent les points sémantiques en question.

4 Génération du code exécutable

n'existe pas

4.1. ~~Dessinez les segments relatifs au code exécutable, aux donnée statiques, à la pile et au tas et représentez les informations qui décrivent l'état mémoire lors d'un appel de la fonction ci-dessous.~~
4.2. Ajouter à G1 des points de génération et des attributs qui engendre le code exécutable pour la seule ligne numéro 6 (celle qui contient un « et logique »). Pour faire simple, on supposera que le code en question est destiné à une machine à pile.

```
1       int bAlexIcIdentificateuR(int eD, int bMiseAuPoint)){
2           int bId;
3            int * pId=malloc(sizeof(int));
4           static int nbAppel=0; nbAppel++;
5            Assert(eD>=0 & eD<(nLexèmeEnTouT)); //soyons prudent//
6            bId = eD>=0 && eD<(nLexemeEnTouT);
7            bId = bId&&(atPilE[eD].classe=IcIdentificateuR);
8            return bId;
9       }
```

.....