

Algorithmique et bases de la programmation

Durée : 1h30

Documents autorisés : **AUCUN**

Remarques :

- Veuillez lire attentivement les questions avant de répondre.
- Le barème donné est un barème indicatif qui pourra évoluer lors de la correction.
- Rendez une copie propre.
- N'utilisez pas de crayon à papier sur votre copie.

1 QCM (5 points)

Répondez au qcm ci joint (à rendre avec votre copie). La note de chaque question peut varier de -1 à $+1$. La note à l'exercice (0 au minimum) est la moyenne des notes de chaque question multipliée par le nombre de points de l'exercice.

Soit B le nombre de bonnes réponses à une question et soit M le nombre de mauvaises réponses à cette même question ($B+M$ est égale au nombre de réponses à la question). Chaque bonne réponse cochée rapporte $1/B$ points et chaque mauvaise réponse $-1/M$ points.

2 Anagramme (4 points)

Deux mots sont des anagrammes si l'un est une permutation des lettres de l'autre. Par exemple les mots suivants sont des anagrammes :

- aimer et maire
- chien et niche
- ...

Dans cet exercice nous représentons les mots par des chaînes de caractères et par définition on considère que deux chaînes vides sont des anagrammes.

On considère que l'on possède les fonctions suivantes :

- **fonction** longueur (chaîne : **Chaîne de caracteres**) : **Naturel**
retourne 0 si chaîne=""
- **fonction** iemeCaractere (chaîne : **Chaîne de caracteres**, indice : **Naturel**) : **Caractere**
|précondition(s) $1 \leq \text{indice}$ et $\text{indice} \leq \text{longueur}(\text{chaîne})$
- **fonction** positionCaractere (chaîne : **Chaîne de caracteres**, car : **Caractere**) : **Naturel**
retourne 0 si car non présent dans chaîne
- **fonction** supprimerCaractere (chaîne : **Chaîne de caracteres**, position : **Naturel**) : **Chaîne de caracteres**
|précondition(s) $1 \leq \text{position}$ et $\text{position} \leq \text{longueur}(\text{chaîne})$

Proposez le corps de la fonction récursive suivante qui permet de savoir si deux mots sont des anagrammes :

fonction sontDesAnagrammes (mot1,mot2 : **Chaîne de caracteres**) : **Booleen**

3 Problème : calculer la valeur d'une expression booléenne (11 points)

3.1 TAD Dictionnaire (1 point)

Rappelez le TAD Dictionnaire vu en cours (sans les parties axiomes et sémantiques).

3.2 Évaluation d'une expression booléenne (10 points)

On considère qu'une expression booléenne est soit une opération unaire (opérateur **non**) soit une opération binaire (opérateur **ou** et **et**). L'opérande d'une opération booléenne est soit une constante (valeur **vrai** ou **faux**), soit une variable, soit une expression booléenne entre parenthèses.

On suppose que :

- la valeur booléenne d'une variable est donnée par un dictionnaire dont la clé est l'identifiant de la variable ;
- l'on possède les types énumérés :
 - **Type** OperateurUnaire = {not}
 - **Type** OperateurBinaire = {and,or}

L'objectif est de concevoir l'opération `evaluerExpressionBooléenne` qui calcule la valeur d'une expression booléenne représentée par une chaîne de caractères (par exemple "a et (c ou (vrai et d))"). Un booléen permet de savoir si cette chaîne représente ou pas une expression booléenne valide. L'opération `evaluerExpressionBooléenne` est insensible à la casse. Les constantes, opérateurs, variables sont séparés par des espaces. Enfin :

- les constantes sont identifiées par les chaînes de caractères "vrai" et "faux";
- les opérateurs booléens sont identifiés par les chaînes de caractères "non", "et" et "ou";
- les variables sont identifiées par une suite de caractères parmi les caractères suivants : 'a' à 'z', '0' à '9' et '_'. Il y a au moins un caractère et le premier sera obligatoirement une lettre ou un '_'.

3.2.1 Analyse (2 points)

Complétez l'analyse descendante donnée par la figure 1. Pourquoi y a-t-il un lien de dépendance entre l'opération `reconnaîtreOperande` et `reconnaîtreExpressionBooléenne`?

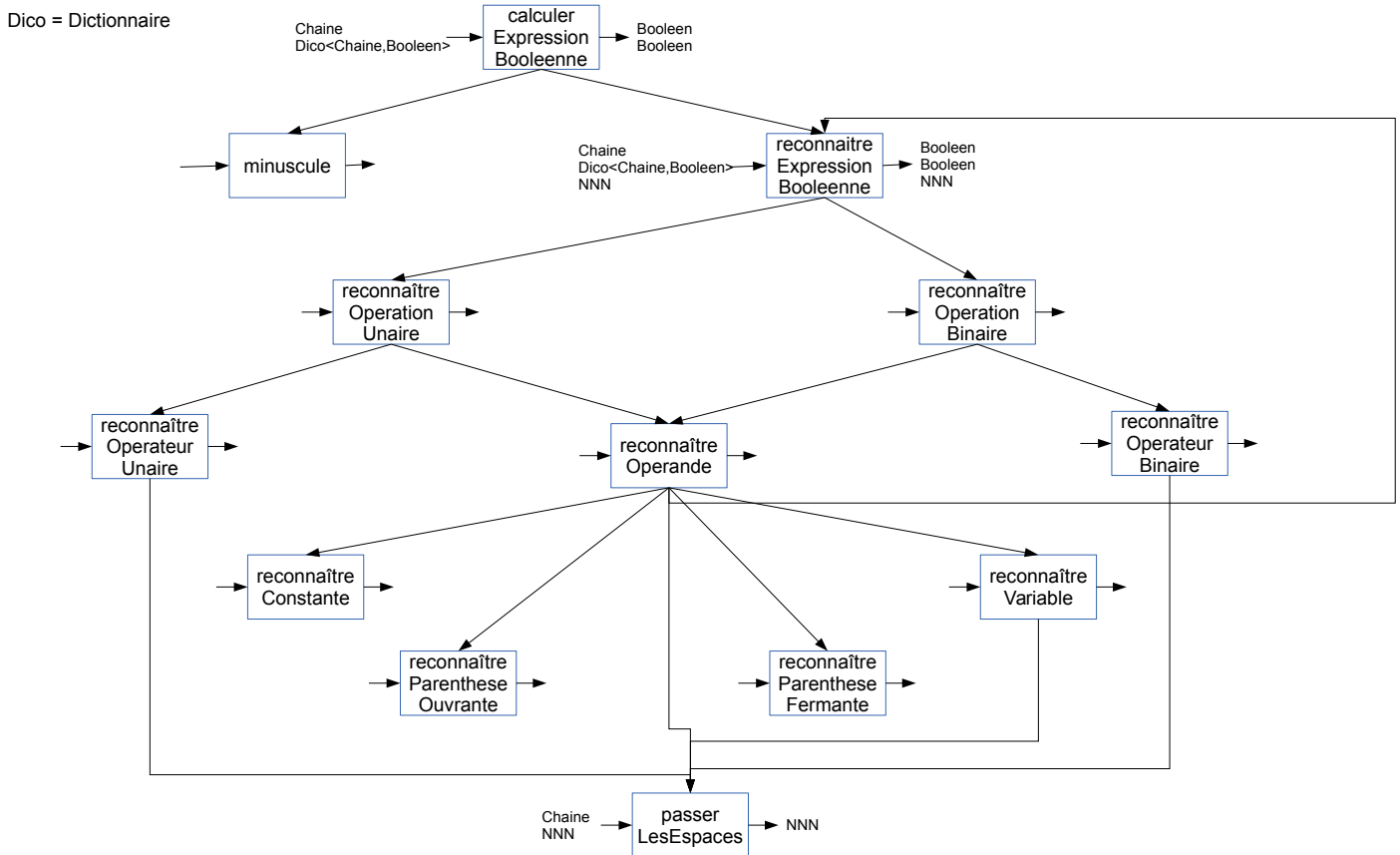


FIGURE 1 – Analyse descendante

3.2.2 Conception préliminaire (1 point)

Donnez les signatures des procédures ou fonctions des opérations `reconnaîtreOperationBinaire`, `reconnaîtreOperande`, `reconnaîtreConstante`, `reconnaîtreParentheseOuvrante`, `reconnaîtreParentheseFermante`, `reconnaîtreVariable`, `passerLesEspaces`.

3.2.3 Conception détaillée (7 points)

Donnez les algorithmes des procédures ou fonctions des opérations :

1. `reconnaîtreOperationBinaire`
2. `reconnaîtreOperande`
3. `reconnaîtreVariable`