

BACCALAURÉAT GÉNÉRAL

ÉPREUVE D'ENSEIGNEMENT DE SPÉCIALITÉ

SESSION 2022

NUMÉRIQUE ET SCIENCES INFORMATIQUES

Mercredi 11 mai 2022

Durée de l'épreuve : **3 heures 30**

L'usage de la calculatrice n'est pas autorisé.

Dès que ce sujet vous est remis, assurez-vous qu'il est complet.

Ce sujet comporte 14 pages numérotées de 1/14 à 14 /14.

Le candidat traite au choix 3 exercices parmi les 5 exercices proposés

Chaque exercice est noté sur 4 points.

EXERCICE 1 (4 points)

Cet exercice composé de deux parties A et B, porte sur les structures de données.

Partie A : Expression correctement parenthésée

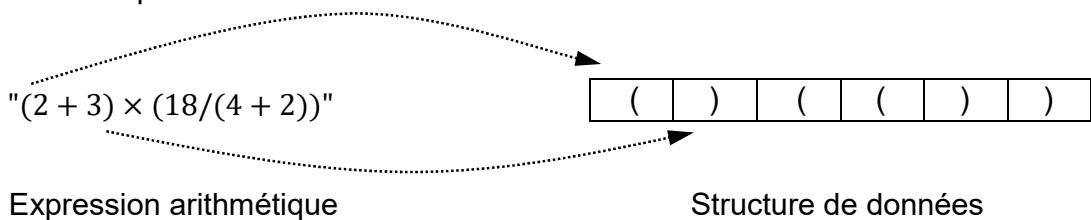
On veut déterminer si une expression arithmétique est correctement parenthésée.

Pour chaque parenthèse fermante ")" correspond une parenthèse précédemment ouverte "(".

Exemples :

- L'expression arithmétique " $(2 + 3) \times (18/(4 + 2))$ " est correctement parenthésée.
- L'expression arithmétique " $(2 + 3) \times (18/(4 + 2$ " est non correctement parenthésée.

Pour simplifier les expressions arithmétiques, on enregistre, dans une structure de données, uniquement les parenthèses dans leur ordre d'apparition. On appelle expression simplifiée cette structure.



1. Indiquer si la phrase « les éléments sont maintenant retirés (pour être lus) de cette structure de données dans le même ordre qu'ils y ont été ajoutés lors de l'enregistrement » décrit le comportement d'une file ou d'une pile. Justifier.

Pour vérifier le parenthésage, on peut utiliser une variable `controleur` qui :

- est un nombre entier égal à 0 en début d'analyse de l'expression simplifiée ;
- augmente de 1 si l'on rencontre une parenthèse ouvrante "(" ;
- diminue de 1 si l'on rencontre une parenthèse fermante ")".

Exemple : On considère l' expression simplifiée A : " $()()()$ "

Lors de l'analyse de l'expression A, `controleur` (initialement égal à 0) prend successivement pour valeur 1, 0, 1, 2, 1, 0. Le parenthésage est correct.

2. Écrire, pour chacune des 2 expressions simplifiées B et C suivantes, les valeurs successives prises par la variable `controleur` lors de leur analyse.

Expression simplifiée B : " $(()())$ "

Expression simplifiée C : " $(())()$ "

3. L'expression simplifiée B précédente est mal parenthésée (parenthèses fermantes manquantes) car le contrôleur est différent de zéro en fin d'analyse. L'expression simplifiée C précédente est également mal parenthésée (parenthèse fermante sans parenthèse ouvrante) car le contrôleur prend une valeur négative pendant l'analyse.

Recopier et compléter uniquement les lignes 13 et 16 du code ci-dessous pour que la fonction `parenthesage_correct` réponde à sa description.

```

1 def parenthesage_correct(expression):
2     ''' fonction renvoyant True si l'expression arithmétique
3     simplifiée (str) est correctement parenthésée, False
4     sinon.
5     Condition: expression ne contient que des parenthèses
6     ouvrantes et fermantes '''
7
8     contrôleur = 0
9     for parenthèse in expression: #pour chaque parenthèse
10        if parenthèse == '(':
11            contrôleur = contrôleur + 1
12        else:# parenthèse == ')'
13            contrôleur = contrôleur - 1
14            if contrôleur ... : # test 1 (à recopier et compléter)
15                #parenthèse fermante sans parenthèse ouvrante
16                return False
17            if contrôleur ... : # test 2 (à recopier et compléter)
18                return True #le parenthésage est correct
19        else:
20            return False #parenthèse(s) fermante(s) manquante(s)

```

Partie B : Texte correctement balisé

On peut faire l'analogie entre le texte simplifié des fichiers HTML (uniquement constitué de balises ouvrantes `<nom>` et fermantes `</nom>`) et les expressions parenthésées :

Par exemple, l'expression HTML simplifiée :

`"<p></p>"` est correctement balisée.

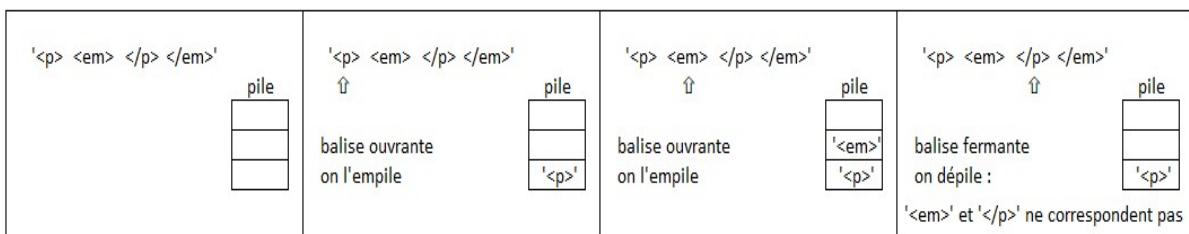
On ne tiendra pas compte dans cette partie des balises ne comportant pas de fermeture comme `
` ou ``.

Afin de vérifier qu'une expression HTML simplifiée est correctement balisée, on peut utiliser une pile (initialement vide) selon l'algorithme suivant :

On parcourt successivement chaque balise de l'expression :

- lorsque l'on rencontre une balise ouvrante, on l'empile ;
- lorsque l'on rencontre une balise fermante :
 - si la pile est vide, alors l'analyse s'arrête : le balisage est incorrect ,
 - sinon, on dépile et on vérifie que les deux balises (la balise fermante rencontrée et la balise ouvrante dépilerée) correspondent (c'est-à-dire ont le même nom) si ce n'est pas le cas, l'analyse s'arrête (balisage incorrect).

Exemple : État de la pile lors du déroulement de cet algorithme pour l'expression simplifiée "`<p></p>`" qui n'est pas correctement balisée.



État de la pile lors du déroulement de l'algorithme

4. Cette question traite de l'état de la pile lors du déroulement de l'algorithme.
 - Représenter la pile à chaque étape du déroulement de cet algorithme pour l'expression "`<p></p>`" (balisage correct).
 - Indiquer quelle condition simple (sur le contenu de la pile) permet alors de dire que le balisage est correct lorsque toute l'expression HTML simplifiée a été entièrement parcourue, sans que l'analyse ne s'arrête.
5. Une expression HTML correctement balisée contient 12 balises.
Indiquer le nombre d'éléments que pourrait contenir au maximum la pile lors de son analyse.