LO21-2010A-FS01-01.

Examen Final – LO21 et LO27

2 heures

Aucun document autorisé

Le barême par exercice est donné à titre indicatif et pourra être modifié durant la correction

Exercice 1: TAD Polynôme (10 points)

Un polynôme est considéré comme une liste de monômes ordonnée par degré croissant.
Un monôme est représenté par son coefficient et son degré.

Liste des fonctions disponibles

- créer_monôme(coefficient : Flottant, degré : Entier) : crée un monôme à partir de son degré et de son coefficient.
- créer_polynôme() : crée un polynôme vide
- somme(p1: Polynome, p2: Polynome) : Polynome : calcule le polynôme somme de p1 et p2
- ajouter_tête(p: Polynome, m: Monome):Polynome
- ajouter_queue(p: Polynome, m: Monome):Polynome
- coefficient(m: Monome) : Flottant
- degré(m: Monome) : Entier
- reste(p: Polynome) : Polynome
- valeur_tête(p: Polynome) : Monome
- est_vide(p: Polynome) : Booléen

Demander à l'enseignant surveillant l'examen avant d'utiliser d'autres fonctions.

Question 1 : Multiplication de polynomes (4 points)

Fournir l'algorithme du sous-programme « *multiplication* » qui calcule le polynôme f.g issu de la multiplication de deux polynômes f et g.
f.g devra être une liste de monômes ordonnée par degré croissant.
Par exemple si pour tout réel x :

$f(x) = -3.x^2 + 4.x - 2$

$g(x) = x^3 - x + 1$

alors f.g est défini par :

$(f.g)(x) = f(x) . g(x)$

$= (-3.x^2 + 4.x - 2) . (x^3 - x + 1)$

$= -3.x^5 + 3.x^3 - 3.x^2 + 4.x^4 - 4.x^2 + 4.x - 2.x^3 + 2.x - 2$

$= -3.x^5 + 4.x^4 - 7.x^2 + 6.x - 2$

De façon plus générale le degré d'un produit de polynômes est la somme des degrés des facteurs.

$d°(f.g) = d°(f) + d°(g)$

Question 2 : $K^{ième}$ dérivée d'un monôme (2 points)

Fournir l'algorithme du sous-programme « *mderivative* » qui calcule la $k^{ième}$ dérivée d'un monôme donné.

On rappelle que la $k^{ième}$ dérivée d'un monôme $M_i(X)$ de degré i est le monome $D_k(M_i(X))$ est défini de la manière suivante :

$D_k(M_i(X)) = 0$, si $k > i$

$D_k(M_i(X)) = i*(i-1)*....*(i-k+1)*a_i*X^{i-k}$, sinon.

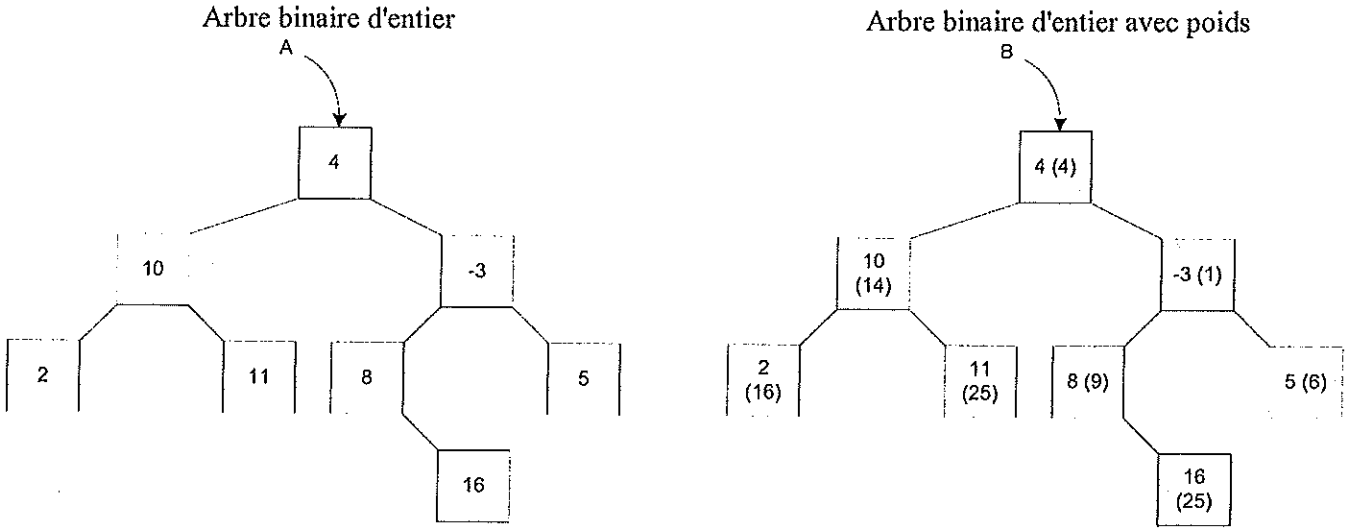
Indice : Trouver la relation de récurrence qui permet de calculer $D_k(M_i(X)$ en fonction de $D_{k-1}(M_i(X))$.

Question 3 : $K^{ième}$ dérivée d'un polynôme (4 points)

Fournir l'algorithme du sous-programme « *pderivative* » qui calcule la $k^{ième}$ dérivée d'un polynôme donné.

Exercice 2 : TAD Arbre binaire (6 points)

On traite des arbres binaire d'entiers. Le poids d'un nœud n d'un arbre t est la somme des entiers étiquetant les nœuds du chemin de la racine de t à ce nœud n.
Dans la figure ci-dessous la poids des nœuds apparaît entre parenthèse.



Question 1 (3 points)

Fournir l'algorithme du sous-programme « *buildweightedtree* » permettant de construire l'arbre binaire de type B à partir d'un arbre binaire d'entier comme A.
On considéra que la valeur des nœuds de l'arbre de type B est un couple d'entier (valeur, poids).

Question 2 (3 points)

Fournir l'algorithme du sous-programme « *removeleaves* » permettant de supprimer toutes les feuilles d'un arbre binaire de type B dont le poids est strictement supérieur à une valeur entière X donnée.

Exercice 3 : Grammaires et automates (4 points)

Considérons l'alphabet A = {0,1,2,3,4,5,6,7,8,9} et la grammaire G = ({S,N}, A, →, S)
définie sur A avec les règles suivantes :

S → S N | N
N → 0 | 1 | 2 | 3 | 4 | 5 | 6 | 7 | 8 | 9

Question 1 (2 points)
Quel est le langage L engendré (défini) par la grammaire G ?

Question 2 (2 points)
Construire (Dessiner) l'automate à état fini permettant de reconnaître un mot du langage L. On supposera que l'automate prends en entrée une chaine de caractère terminant par la caractère * indiquant la fin de la chaine à analyser.

LO2f-20104-FS02-01

Final Exam – LO21 et LO27
2 hours
No document authorized

Exercise 1: ADT Polynomial (10 points)

A polynomial is represented as a List of Monomials sorted in ascendant degree.
A monomial is represented by its coefficient and its degree

List of available functions

- create_monomial(coefficient : Float, degree : Integer) : creates a monomial according to its coefficient and degree
- create_polynomial() : creates a empty polynomial
- sum(p1: Polynomial, p2: Polynomial) : Polynomial: computes the sum of p1 and p2
- add_head(p: Polynomial, m: Monomial):Polynomial
- add_tail(p: Polynomial, m: Monomial):Polynomial
- coefficient(m: Monomial) : Float
- degree(m: Monomial) : Integer
- rest(p: Polynomial) : Polynomial
- head_value(p: Polynomial) : Monomial
- is_empty(p: Polynomial) : Boolean

Ask the teacher supervising the test before using other functions.

Question 1 : Polynomial Multiplication (4 points)

Provide the algorithm of a subroutine « *multiplication* » which computes the polynomial f.g derived by multiplying two polynomials f and g.
f.g has to be represented as a list of monomials sorted in ascendant degree.

For example, if for all real x :
$f(x) = -3.x^2 + 4.x - 2$
$g(x) = x^3 - x + 1$
then f.g is defined by :
$(f.g)(x) = f(x) . g(x)$
$= (-3.x^2 + 4.x - 2) . (x^3 - x + 1)$
$= -3.x^5 + 3.x^3 - 3.x^2 + 4.x^4 - 4.x^2 + 4.x - 2.x^3 + 2.x - 2$
$= -3.x^5 + 4.x^4 - 7.x^2 + 6.x - 2$

More generally the degree of a polynomial product is the sum of the degrees of its factors.
$d°(f.g) = d°(f) + d°(g)$

Question 2 : Kth derivative of a monomial (2 points)

Provide the algorithm of a subroutine « *mderivative* » computing the kth derivative of a given monomial.

Remind that the kth derivative of a monomial $M_i(X)$ of degree i is the monomial $D_k(M_i(X))$ defined as follows
$D_k(M_i(X)) = 0$, si $k > i$
$D_k(M_i(X)) = i*(i-1)*....*(i-k+1)*a_i*X^{i-k}$, sinon.

Tip : Find the recurrence relation to compute $D_k(M_i(X)$ as a function of $D_{k-1}(M_i(X)$

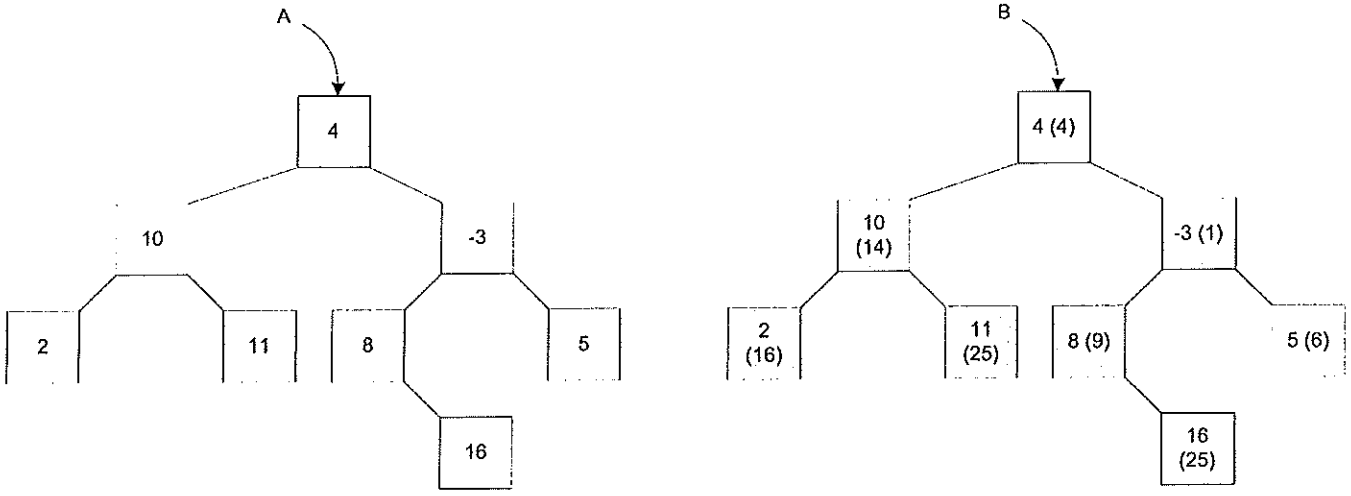
Question 3 : Kth derivative of a polynomial (4 points)

Provide the algorithm of a subroutine « *pderivative* » computing the kth derivative of a given polynomial.

Exercise 2: ADT Binary Tree (6 points)

We are dealing with binary trees of integers. The weight of a node n of a tree t is the sum of the integers labeling the nodes in the path from the root of t to that node n.

In the figure below, the weight of the nodes appears in brackets.



Question 1 (3 points)
Provide the algorithm of a subroutine « *buildweightedtree* » creating a tree of type B from a binary tree of integers like A.
We consider that the value of the nodes of the tree B is a pair of integer (value, weight).

Question 2 (3 points)
Provide the algorithm of a subroutine « *removeleaves* » removing all the leaves of a binary tree B whose weight is strictly greater than a given integer value X.

Exercise 3: Grammars et Automata (4 points)

Let's consider the alphabet A = {0,1,2,3,4,5,6,7,8,9} and the grammar G = ({S,N}, A, →, S)
defined on A with the following rules :
S → S N | N
N → 0 | 1 | 2 | 3 | 4 | 5 | 6 | 7 | 8 | 9

Question 1 (2 points)
What is the language L generated (defined) by the grammar G?

Question 2 (2 points)
Construct (draw) the finite state automaton recognizing a word of L. We assume that the automaton takes as input a string ending with the character * indicating the end of the string to parse.