Final Exam - LO27

2 hours

No document authorized

Supervisor: Nicolas GAUD

"*ADT*" means "Abstract Data Type"

Exercise 1 : Bus Network (5 points ± 1)

Let's consider the list of all bus stops of a public transportation.
A bus line is represented as a list of bus stop. Each stop is characterized by its name and the distance that separates it from the previous stop of the line.

1) Given a bus line, write the recursive algorithm of a subprogram named ***LineLength*** which computes the total length of this line.

2) We consider the whole bus network as a list of bus lines. Each item in this list is characterized by a line number and a list of bus stops.

   a) Write the recursive algorithm of a subprogram named ***shortestLine*** which returns the number of the shortest bus line (the line having the smallest total length), without using a temporary structure to store the length of the lines.

   b) Provide in C language, the complete definition of the **Bus Network** type.

Exercise 2 : Grammars (4 points ± 1)

Lets consider the following alphabet V = {+,=,a}.
Lets consider the grammar G ({S, M}, {+,=,a}, →, S) defined on V according to the following rules:

1. $S \rightarrow aSa$
2. $S \rightarrow a + Ma$
3. $M \rightarrow aMa$
4. $M \rightarrow a = a$

1) Show by derivation that the word aa + aaaa = aaaaaa belongs to the language L(G) (For each derivation step, specify the number of the grammatical rule you are applying).

2) To describe the language, a set (usually infinite) words, we use the following notations:
if t_1 and t_2 are two sets of words:

- $t_1|t_2$ denotes the union of sets t_1 and t_2
- t_1t_2 denotes all the words formed by concatenating a word of t_1 and a word of t_2.
- t_1^* denotes the set of words obtained by concatenating any number of words of t_1 (possibly none).
- t_1^n denotes the set of words obtained by concatenating n words of t_1.

Finally, we note the empty word ε, whose number of characters is zero.

Using this notation, please describe the language L(G) generated by the grammar G.

Exercise 3 : Collatz Conjecture (6 points ± 1)

Let's consider the collatz function ($f : N \rightarrow N$) defined as follows:

- If the number n is even, f(n) = n/2.
- If the number n is odd, f(n) = 3*n + 1.

Given a positive integer C, and the sequence U_n defined as follows:

- $U_0 = C$
- Pour n> 0, $U_n = f(U_{n-1})$

The Collatz conjecture asserts that for any C> 0, it exists an index n such that the sequence reaches 1.

Example

Starting with $n = 6$, the sequence is: 6, 3, 10, 5, 16, 8, 4, 2, 1.

Starting with $n = 11$, the sequence is: 11, 34, 17, 52, 26, 13, 40, 20, 10, 5, 16, 8, 4, 2, 1.

1) Write an algorithm of the subprogram called ***Collatz*** that computes the value of the function f for a given integer.

2) Given a positive integer C, write a recursive algorithm of the subprogram called ***Syracuse*** that builds the list containing the successive values of the sequence U_n ($U_0, U_1, \ldots, U_k=1$) until it reaches the value 1 for the first time.
For C = 6, ***Syracuse*** builds the following list: 6, 3, 10, 5, 16, 8, 4, 2, 1.

3) Write a recursive algorithm of a subprogram called ***StoppingTime*** that computes the minimal number of iterations of the Collatz sequence U_n required to reach 1 for the first time.
For C = 6, ***StoppingTime*** returns 9.

Exercise 4 : Tennis Tournament (5 points ± 1)

We want to represent the results of a tennis tournament using a binary tree. Each tree node contains the name of the winner (string), or the identifier of the match (an integer) between the players within those child nodes. For leaf nodes, the winner corresponds to the player's name and the identifier of the match is -1.

To compare strings, you can use (without defining it) the function ***stringEquality(ch1, ch2)*** which returns true if the two specified strings ch1 and ch2 are equal, false otherwise.

1) Write a recursive algorithm of a subprogram called **winnerMatchs** that builds the list containing all identifiers of the matches won by the winner of the tournament. The returned list must be sorted by order of played matches.

Examen Final – LO27

2 heures

Aucun document autorisé

Enseignant responsable: Nicolas GAUD

Exercice 1 : Lignes de Bus (5 points ± 1)

On considère la liste de tous les arrêts de bus d'un réseau de transport en commun.
Une ligne de bus est une liste d'arrêt. Chaque arrêt est caractérisé par son nom et la distance qui le sépare de l'arrêt précédent de la ligne.

1) Étant donnée une ligne de bus, écrire l'algorithme récursif du sous programme nommé ***LineLength*** permettant de calculer la distance totale de toute cette ligne.

2) On considère l'ensemble du réseau de bus comme une liste de lignes de bus. Chaque élément de cette liste est caractérisé par un numéro de la ligne et une liste d'arrêts.

   a) Écrire l'algorithme du sous programme récursif nommé ***shortestLine*** qui donne le numéro de la ligne la plus courte (la ligne ayant la plus petite distance totale), sans utiliser de structure temporaire pour stocker la longueur des lignes.

   b) Donner en langage C la déclaration complète du type réseau de bus.

Exercice 2 : Grammaires (4 points ± 1)

Considérons l'alphabet V suivant {+,=,a} et la grammaire G suivante ({S, M}, {+,=,a}, →, S) définie sur V avec les règles suivantes:

1. S → aSa
2. S → a + Ma
3. M → aMa
4. M → a = a

1) Monter par dérivation que le mot aa + aaaa = aaaaaa appartient au langage L(G) (Pour chaque étape de la dérivation, veuillez préciser le numéro de la règle grammatical utilisée).

2) Afin de décrire un langage, c'est à dire un ensemble (généralement infini) de mots, on utilise la notation suivante:
Si t_1 et t_2 sont deux ensembles de mots:

- $t_1|t_2$ désigne l'union des ensemble t_1 et t_2
- t_1t_2 désigne l'ensemble des mots formés par concaténation d'un mot de t_1 et d'un mot de t_2.
- t_1^* désigne l'ensemble des mots obtenus par concaténation d'un nombre quelconque de mot de t_1 (éventuellement aucun).
- t_1^n désigne l'ensemble des mots obtenus par concaténation de n mots de t_1.

Enfin, on note ε le mot vide, dont le nombre de caractère est nul.

Utiliser cette notation, pour décrire le langage L(G) généré par la grammaire G.

Exercice 3 : Conjecture de Syracuse (6 points ± 1)

Considérons la fonction de Collatz f : N → N définie comme suit:

- Si n est pair, f(n) = n/2.
- Si n est impair, f(n) = 3*n + 1.

Étant donné un entier positif C et la suite U_n définie de comme suit :

- $U_0 = C$
- Pour n> 0, $U_n = f(U_{n-1})$

La conjecture affirme que, pour tout C > 0, il existe un indice n tel que la suite atteigne 1.

Exemples
En partant de *C* = 6, la suite est la suivante: 6, 3, 10, 5, 16, 8, 4, 2, 1.
En partant de *C* = 11, la suite est la suivante: 11, 34, 17, 52, 26, 13, 40, 20, 10, 5, 16, 8, 4, 2, 1.

1) Écrire l'algorithme d'un sous-programme nommé ***Collatz*** qui calcule la valeur de la fonction f pour un entier donné.

2) Étant donné un entier positif C, écrire l'algorithme récursif du sous-programme nommé ***Syracuse*** qui construit la liste des valeurs successives de la suite U_n (U_0, U_1,…, U_k=1) jusqu'à ce qu'elle atteigne la valeur 1 pour la première fois.
Par exemple, pour C=6, ***Syracuse*** construit la liste suivante:6, 3, 10, 5, 16, 8, 4, 2, 1.

3) Écrire l'algorithme récursif d'un sous-programme nommé ***stoppingTime*** qui calcule le nombre minimal d'itérations nécessaire à la suite U_n pour atteindre 1 pour la première fois.

Par exemple, pour C=6, ***stoppingTime*** revoie la valeur 9.

Exercice 4 : Tournoi de tennis (5 points ± 1)

On souhaite représenter les résultats d'un tournois de tennis à l'aide d'un arbre binaire. Chaque nœud de l'arbre contient le nom du vainqueur (chaine de caractères), l'identifiant ou numéro du match (un entier) entre les joueurs se trouvant dans ces nœuds fils. Pour les nœuds feuilles, le nom du vainqueur correspond au nom d'un joueur et l'identifiant du match est –1.

Pour comparer des chaînes de caractères, vous pouvez utiliser (sans la définir) l'opération ***stringEquality(ch1,ch2)*** qui donne vraie si les deux chaînes ch1 et ch2 sont égales, faux sinon.

1) Écrire l'algorithme récursif du sous programme nommé **winnerMatchs** construisant une liste de tous les identifiants des matchs gagnés par le vainqueur du tournois. La liste retournée doit être triée par ordre des matchs joués.