Modalités :

- Durée : 2 heures
- Aucun document autorisé, ni machine à calculer, ni téléphone
- Le barème est donné à titre indicatif (±1)

Exercice 1 (6 points)

Un arbre binaire booléen est un arbre binaire dont les nœuds ne peuvent avoir comme valeur que 0 ou 1.

1) Étant donné un arbre binaire booléen, écrire l'algorithme **récursif** du sous-programme *créerArbreBooléen* qui construit un nouvel arbre binaire booléen de même morphologie et dont les valeurs sont organisées telles que (3 points):

- la valeur de chaque nœud feuille est la négation de la valeur du nœud feuille de même position dans l'arbre binaire booléen donné en entrée,
- la valeur de chaque nœud qui n'est pas une feuille est le OU logique entre la valeur du nœud fils gauche et la valeur du nœud fils droit du nœud de même position dans l'arbre donné en entrée.

2) Donner en langage C, la déclaration complète du type arbre binaire booléen (1 point).

3) Traduire complètement l'algorithme du sous-programme de la question 1 en langage C en respectant la déclaration fournie à la question 2 (2 points).

Exercice 2 (10 points)

On souhaite gérer un parc de location de véhicules. On considère le parc de véhicules comme une liste de véhicules. Chaque véhicule est caractérisé par son numéro d'immatriculation (une chaine de caractères), sa puissance fiscale (un entier strictement positif) et son état (disponible ou non). Pour gérer les locations, on dispose d'une liste de locations. Chaque élément de cette liste est caractérisé par le numéro d'immatriculation du véhicule, la date de location (une date), la date de retour (une date), et le nom du client ayant loué le véhicule (une chaine de caractères). On suppose que les deux listes sont triées par ordre croissant sur les numéros d'immatriculation.

1) Étant donnés une demande de location (la puissance fiscale du véhicule souhaité) et un parc de véhicules, écrire l'algorithme **récursif** du sous-programme *estSatisfiable* qui permet de déterminer si la demande de location peut être satisfaite ou non par au moins l'un des véhicules du parc donné (3 points).

2) Étant donnés un parc de véhicules, une liste de locations et une date d, écrire l'algorithme **itératif** du sous-programme *construireNouveauParc* qui permet de construire une nouvelle liste de véhicules contenant les véhicules du parc avec leur état mis à jour en fonction de la liste de locations (4 points). On considère qu'un véhicule de la liste de location dont la date de retour est inférieure à la date d est disponible.

3) Pour optimiser le stockage des données concernant le parc, on introduit un nouveau type de liste *LocationListe*, où la valeur de chaque élément est caractérisée par la puissance fiscale et la liste des véhicules de location ayant cette puissance. Dans cette dernière liste, un véhicule est composé de son numéro d'immatriculation (une chaine de caractères) et de son état (disponible ou non).

Donner en langage C, la déclaration complète du nouveau type *LocationListe* et celle des autres types nécessaires à sa déclaration (3 points).

Exercice 3 (4 points)

Soit la grammaire G= ({S, A, B}, {0, 1}, P, S) avec P l'ensemble des règles suivant :

1. S → 1A | 0S
2. A → 1B | 0S | 1
3. B → 0B | 1B | 0 | 1

1) Montrer que le mot 001011001 appartient au langage L(G) (1 point).

2) Donner un automate à état fini déterministe permettant de reconnaitre les mots du langage L(G) (3 points).

Terms:

- Duration: 2 hours
- No document authorized, nor calculating machine, nor phone
- The rate is indicative (±1)

Exercise 1 (6 points)

A Boolean binary tree is a binary tree whose nodes can only have as value 0 or 1.

1) Given a Boolean binary tree, write the **recursive** algorithm of the subroutine *buildBooleanTree* that builds a new Boolean binary tree with the same morphology and whose values are organized such that (3 points):

- the value of each leaf node is the negation of the value of the leaf node at the same position in the Boolean binary tree given as input.
- the value of each node, which is not a leaf, is the logical OR between the value of the left and the right children of the node in the same position at the Boolean binary tree given as input.

2) Provide in C language, the full declaration of the type Boolean binary tree (1 point).

3) Translate completely in C the algorithm of the subroutine of Question 1 respecting the declaration provided in Question 2 (2 points).

Exercise 2 (10 points)

We want to manage a fleet of rental vehicles. We consider the fleet as a list of vehicles. Each vehicle is characterized by its registration mark (a string), its fiscal power (a positive integer) and its status (available or not). To manage rentals, there is a list of rentals. Each element of this list is characterized by the registration mark of the vehicle, the rental date (a date), the return date (a date), and the name of the customer who rented the vehicle (a string). It is assumed that the two lists are sorted in ascending order on the registration marks.

1) Given a rental application (the fiscal power of the vehicle required) and a fleet of vehicles, write the **recursive** algorithm of the subroutine *isSatisfiable* that determines if the rental application can be satisfied or not by at least one vehicle in the given fleet (3 points).

2) Given a fleet of vehicles, a list of rentals and a date d, write the **iterative** algorithm of the subroutine *buildNewFleet* that builds a new list of vehicle containing the vehicles of the fleet with their updated status according to the list of rentals (4 points). We consider a vehicle of the rental list with a return date which is lesser than the specified date d is available.

3) To optimize data storage for the fleet, we introduce a new type of list *LocationList*, where the value of each element is characterized by the fiscal power and the list of rental vehicles with this power. In this last list, a vehicle is composed of its registration mark (a string) and its status (available or not).

Provide in C language, the full declaration of the new type *LocationList* and the associated required types (3 points).

Exercise 3 (4 points)

Let the grammar G = ({S, A, B}, {0, 1}, P, S) with P the set of rules as follows:

1. S → 1A | 0S
2. A → 1B | 0S | 1
3. B → 0B | 1B | 0 | 1

1) Show that the word 001011001 belongs to the language L(G) (1 point).

2) Provide a deterministic finite state machine to recognize the words of the language L(G) (3 points).