TP automates

1. Objectif

Le but du programme est de construire un automate à partir de sa description puis de faire passer un mot dans l'automate pour voir si ce mot est reconnu. L'automate est défini sur un alphabet $\Sigma = \{a, b, ..., z\}$, ses états sont numérotés à partir de 0 et il est déterministe.

La description de l'automate sera lue suivant ce format (dans un fichier) :

- taille de l'alphabet ;
- nombre d'états ;
- numéro de l'état initial ;
- nombre d'états terminaux suivi des numéros de ces états ;
- listes des transitions de la forme *état de départ, lettre, état d'arrivée* à raison d'une transition par ligne.

Par exemple :

```
2
5
0
2 1 4
0 a 1
0 b 2
1 a 1
1 b 3
2 a 4
2 b 3
3 a 4
3 b 4
4 a 3
4 b 4
```

2. Questions

Vous devez créer 4 fichiers : `automates.h` contenant la déclaration de la structure et des fonctions, `automates.c` contenant le corps des fonctions sur les automates, `main.c` et `Makefile`.

Question 1 *Décrire la structure de données à utiliser pour ces automates et la déclarer dans* `automates.h`. *La structure doit utiliser des listes d'adjacence. Vous pourrez vous inspirer de vos fonctions sur les listes et sur les graphes. Il faudra créer une structure pour les transitions et une pour l'automate.*

Question 2 *Écrire la fonction* `int* creer_estTerminal(int nbE)` *qui initialise tous les états comme non terminaux (tableau à 0) et retourne le tableau des états terminaux.*

Question 3 *Écrire la fonction* `void mettreTerminal(int* estT, etat e)` *permettant de définir l'état* `e` *de l'ensemble* `estT` *comme étant terminal.*

Question 4 *Écrire la fonction* `ptransition* creer_transitions(int nbE)` *qui crée et retourne un tableau de transitions de taille* `nbE` *initialisé à* `NULL`.

Question 5 *Écrire la fonction* `void ajout_transition(ptransition* tabT, etat source, etat dest, char a);` *qui prend en paramètres un état de départ, une lettre et un état d'arrivée et ajoute la transition à l'automate.*

Question 6 *Écrire la fonction* `pautomate allouer_automate()` *qui crée un automate et retourne un pointeur sur une structure d'automate sans l'initaliser.*

Question 7 *Écrire la fonction* `void init_automate(pautomate A, FILE *fauto)` *qui initialise et retourne un automate dont on lit la description dans un fichier (un élément par ligne, comme décrit au début de la fiche.). (Cette fonction fera appel aux précédentes.)*

Question 8 *Écrire la fonction* `pautomate creer_init_automate(char* nomFichier)` *qui retourne un automate dont la description est dans le fichier de nom* `nomFichier` *Cette fonction va donc faire appel aux fonctions de création puis d'initialisation d'automate.*

Question 9 *Écrire la fonction* `affiche_automate` *qui affiche la description de l'automate passé en paramètre. Par exemple avec le format :*

```
 nb d'etats : 5
taille alphabet : 2
etat initial : 0
etats terminaux :
 1  -  4  -
 les transitions :
( 0, b, 2) - ( 0, a, 1) -
( 1, b, 3) - ( 1, a, 1) -
( 2, b, 3) - ( 2, a, 4) -
( 3, b, 4) - ( 3, a, 4) -
( 4, b, 4) - ( 4, a, 3) -
```

Question 10 *Écrire la fonction* `etat destination (pautomate Aut, etat source, char a);` *qui retourne l'état de destination pour un état de départ et une lettre passée en paramètres.*

Question 11 *Écrire la fonction* `int estReconnu(pautomate A, char* mot)` *qui, pour un mot et un automate* `A`*, renvoie 1 si le mot est reconnu et 0 sinon. Le chemin suivi dans l'automate sera affiché.*

Question 12 *Reprendre le problème avec une structure utilisant les matrices de transitions pour représenter l'automate.*