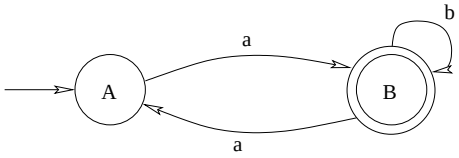
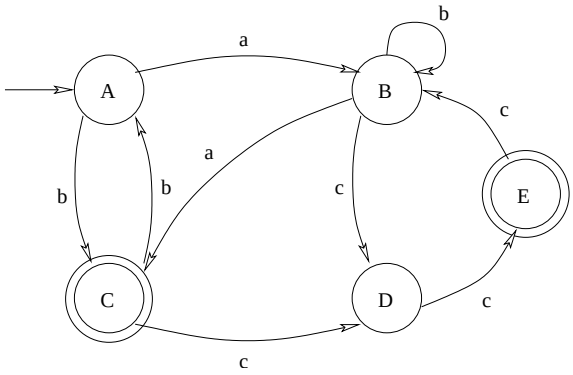
Algorithmique avancée et compilation
Devoir en temps limité
Durée : 2 heures.
Documents autorisés

1. On suppose qu'on décrit des automates **déterministes** sous la forme de suites d'"instructions" comme illustré par les deux exemples suivants :

```
init A ; A--a->B ; B--a->A ; B--b->B ; final B.
```




```
A--a->B ;
A--b->C ;
B--a->C ;
B--b->B ;
B--c->D ;
C--b->A ;
C--c->D ;
D--c->E ;
E--c->B ;
init A ;
final C, E.
```



   Donner une grammaire pour ce langage (la grammaire n'a pas pour fonction de vérifier qu'il s'agit d'un automate déterministe)

   Ecrire un couple *parser/lexer* pour reconnaître les mots de ce langage.

2. Définir une structure de données pour stocker un automate fini déterministe. Pour simplifier la tâche, on supposera qu'on a au plus 26 états correspondants aux lettres majuscules, et au plus 26 transitions correspondant aux lettres minuscules.

   Associer au parser précédent les actions permettant de construire en mémoire l'automate décrit. On suppose que l'automate est syntaxiquement et sémantiquement bien formé.

3. Ajouter dans l'interface `yacc` la possibilité de proposer un mot au parser (par exemple `abbcc?`) pour vérifier que ce mot est reconnu par l'automate (il faut donc implémenter une fonction de parcours de l'automate).

4. [Bonus] Ajouter au parser du contrôle d'erreur pour ne pas construire en mémoire un automate qui serait non déterministe.

Renvoyer par mail en fin d'épreuve un fichier (`tgz` de préférence) contenant : le code `yacc`, le code `lex`, le code C additionnel, le makefile, et un fichier texte contenant une trace d'exécution.