INF203 - Exercices semaine 10
Programmation d'automates

Exercice 1 :

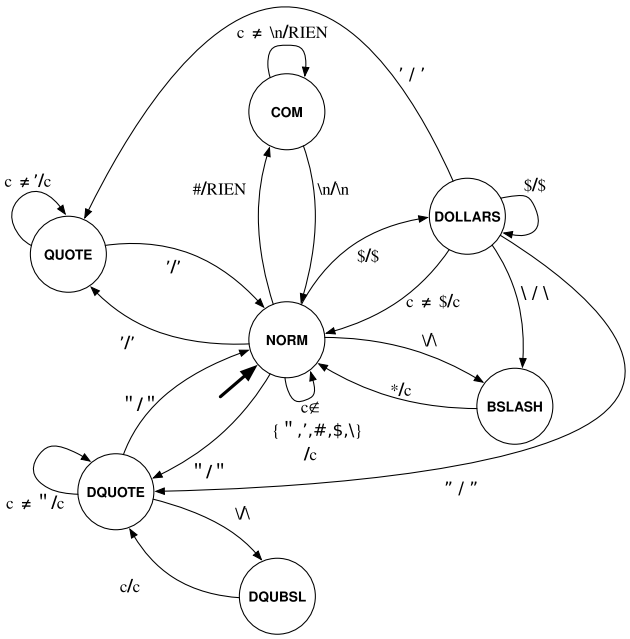
Le but de cet exercice est d'écrire un programme C permettant de supprimer les commentaires d'un script *shell* passé en paramètre. Nous rappelons qu'un commentaire en *shell* commence par le caractère `#` et termine par un passage à la ligne (`\n`). Attention, vous devrez gérer les cas particuliers suivants :

- `#` n'est pas le début d'un commentaire s'il est protégé par un `\` (lui-même non protégé) ;
- `#` n'est pas le début d'un commentaire s'il est précédé par un $ non protégé ;
- les `#` protégés par des guillemets simples ou doubles ne sont pas des débuts de commentaires.

Question 1. Dessinez l'automate modélisant la suppression des commentaires en *shell* en adoptant les conventions suivantes :

- `c` désigne le caractère lu ;
- `/` sépare l'entrée de la sortie ;
- `RIEN` désigne la sortie vide (qu'on pourra omettre par soucis de simplification) ;
- `*` désigne un caractère quelconque ;
- `c` $\notin$ {`"`, `'`, `#`, `$`, `\`} signifie que `c` est un caractère différent des caractères spéciaux `"`, `'`, `#`, `$` et `\`.

Solution :



Dans les questions suivantes, il vous est demandé de programmer les fonctions de l'automate **en version non-tabulée**, c'est-à-dire, **en utilisant des conditions** (`if` ou `switch`). Vous pouvez, si vous le souhaitez, utiliser les macros suivantes pour désigner vos états de manière plus intuitive :

```
#define NORM 0
#define QUOTE 1
#define DOLLARS 2
#define COM 3
#define BSLASH 4
#define DQUOTE 5
```

Question 2. Donner le code C des fonctions de transition et de sortie de l'automate. Leur prototypes sont les suivants :

```
char sortie(int etatcourant, char entree);
int transition(int etatcourant, char entree);
```

Question 3 En utilisant les fonctions `sortie` et `transition`, donnez le code C de la fonction `main` d'un programme qui écrit vers sa sortie le texte d'un script *shell* lu sur son entrée **privé de ses commentaires**. Ce programme accepte au plus 2 arguments de la ligne de commande qui sont interprétés comme des noms de fichiers. Il a le comportement suivant, qui varie selon le nombre d'arguments qui lui sont donnés :

- avec plus de deux arguments, il affiche un message d'erreur ;
- avec deux arguments, le premier est pris comme entrée et le deuxième comme sortie ;
- avec un seul argument, celui-ci est pris comme entrée et la sortie est la sortie standard ;
- sans argument, l'entrée est l'entrée standard et la sortie est la sortie standard.

Il vous est aussi demandé d'afficher les messages d'erreur adéquats lorsqu'une ouverture de fichier échoue.

Solution :

```
#include<stdio.h>

#define NORM 0
#define QUOTE 1
#define DOLLARS 2
#define COM 3
#define BSLASH 4
#define DQUOTE 5

#define RIEN -1

/*

 commentaires : # ..... \n (on affiche quand même le \n)
 cas speciaux :
 - ' ... # ... '
 - " ... # ... "
 - \\\# nombre impair de \
 - $$$# quelque soit le nombre
 */



char sortie(int courant, char entree){
  switch (courant){
  case NORM:
    if(entree=='#') return RIEN;
    return entree;
    break;
```

```
  case COM:
    if(entree=='\n') return entree;
    return RIEN;
    break;

  /*case QUOTE:
  case DOLLARS:
  case DQUOTE:
  case BSLASH:*/
  default:
    return entree;
    break;

  }
}

int etatsuivant(int courant, char entree){
  switch (courant){
  case NORM:
    if(entree=='$') return DOLLARS;
    if(entree=='\\') return BSLASH;
    if(entree=='#') return COM;
    if(entree=='\'') return QUOTE;
    if(entree=='\"') return DQUOTE;
    return NORM;
    break;

  case QUOTE:
    if(entree=='\'') return NORM;
    return QUOTE;
    break;

  case DOLLARS:
    if(entree=='$') return DOLLARS;
    return NORM;
    break;

  case COM:
    if(entree=='\n') return NORM;
    return COM;
    break;

  case DQUOTE:
    if(entree=='\"') return NORM;
    return DQUOTE;
    break;

  /* case BSLASH: */
  default:
    return NORM;
    break;

  }
}



int main(int argc, char *argv[]){
  FILE *s=stdin,*d=stdout;
  int courant=NORM, suivant;
```

```
  char c, res;

  if(argc > 3){
    fprintf(stderr,"Usage %s [source] [destination]\n",argv[0]);
    return 1;
  }

  if(argc > 1){
    s=fopen(argv[1],"r");
    if (!s) {
      fprintf(stderr,"Source inexistante\n");
      return 2;
    }
  }

  if(argc > 2){
      d=fopen(argv[2],"w");
      if (!d) {
      fprintf(stderr,"Probleme destination\n");
      return 3;
    }
  }


  fscanf(s,"%c",&c);
  while(!feof(s)){
    suivant=etatsuivant(courant,c);
    res=sortie(courant,c);

    if(res!=RIEN) fprintf(d,"%c",res);

    courant=suivant;
    fscanf(s,"%c",&c);
  }

  fclose(s);
  fclose(d);

  return 0;
}
```