Corrigé TP7

[a] Par défaut, lorsque l'automate est appelé avec la fonction `init_par_defaut`, toutes les transitions partant d'un état i ramènent au même état i.

[b] Les caractères d'entrée représentant les interactions entre l'utilisateur et la machine sont :

- **c** pour choisir un café
- **r** pour demander que la monnaie soit rendue
- **2** pour symboliser l'insertion de 20 centimes dans la machine

[c] Lorsque l'on saisi un caractère non prévu, le programme affiche : `entree_invalide`

[d] Une implémentation possible de `simule_automate` :

```
void simule_automate(automate *A) {
   int etat_courant = 0, etat_suivant = 0;
   int symbole_entree = ' ';

   etat_courant = A->etat_initial;

   while (1) {
      /* lire une entree */
      lire_entree(&symbole_entree);

      /* Si q est saisi, arrêter le programme */
      if ('q' == symbole_entree) {
         printf ("Au revoir !\n");
         return;
      }

      /* calculer l'état suivant */
      etat_suivant = A->transitions[etat_courant][symbole_entree];

      /* ecrire le message de sortie */
      ecrire_sortie (A->sortie[etat_courant][symbole_entree]);

      /* mettre à jour l'état courant */
      etat_courant = etat_suivant;
   }
}
```

[e] Sans cette instruction, la chaîne de caractère correspondant à une sortie non définie serait "`entree_invalide`", c'est à dire l'entrée par défaut assignée par la fonction `init_par_defaut`.

[f] Proposition d'implémentation

```
void automate_from_file(automate *A, FILE *file) {
   init_par_defaut(A);
   int nb_transition = 0, nb_sortie = 0;

   // Nombre d'états
   fscanf (file, "%d", &(A->nb_etats));
   printf("nb etats : %d\n", A->nb_etats);

   // Nombre d'états finals
   fscanf (file, "%d", &(A->nb_etats_finals));
   printf("nb etats finals : %d\n", A->nb_etats_finals);
   for (int i=0; i<A->nb_etats_finals; i++) {
      fscanf (file, "%d", &(A->etats_finals[i]));
      printf("\tetat final : %d\n", A->etats_finals[i]);
   }

   // Transitions
   fscanf (file, "%d", &nb_transition);
   printf("nb transitions : %d\n", nb_transition);
   for (int i=0; i<nb_transition; i++) {
      int etat = 0, etat_suivant = 0;
      char entree;
      fscanf (file, "%d %c %d", &etat, &entree, &etat_suivant);
      A->transitions[etat][entree] = etat_suivant;
      A->sortie[etat][entree][0] = '\0';
      printf("\tTransition : %d %c %d\n", etat, entree, etat_suivant);
   }

   // Sorties
   fscanf (file, "%d", &nb_sortie);
   printf("nb sorties : %d\n", nb_sortie);
   for (int i=0; i<nb_sortie; i++) {
      int etat = 0;
      char entree, sortie[LG_MAX_SORTIE];
      fscanf (file, "%d %c %s", &etat, &entree, &sortie);
      strcpy(A->sortie[etat][entree], sortie);
      printf("\tSortie : %d %c %s\n", etat, entree, sortie);
   }
}
```

On en profite pour proposer une implémentation du main modifier qui utilise la fonction de lecture de `Mon_automate.auto`.

```
int main(int argc, char *argv[]) {
   if (argc != 2) {
      printf ("USAGE: ./%s automate_file.auto\n", argv[0]);
      return 1;
   }

   char *automate_filename = argv[1];
   FILE *automate_file = fopen(automate_filename, "r");
   if (automate_file == NULL) {
      printf ("Unable to read file : %s\n", automate_filename);
      return 1;
   }

   automate A ;
   automate_from_file(&A, automate_file);
   simule_automate(&A);

   fclose (automate_file);
   return 0;
}
```

[g] L'instruction `sizeof` permet de connaitre la taille en octets d'un type. On peut placer la ligne suivante au début du main, par exemple :

```
   printf ("Size of automate : %d\n", sizeof(automate));
```

Le programme afficher :

```
Sizeof automate : 2163212
```

On pourrait être étonné d'un tel résultat mais c'est bien correct : notre type automate nécessite plus de 2 Mo pour stocker toutes ses informations. On peut refaire le calcul pour s'en convaincre :

```
typedef struct {
    int nb_etats;                                          => 4 octets
    int nb_etats_finals;                                   => 4 octets
    int etat_initial;                                      => 4 octets
    int etats_finals[NB_MAX_ETATS];                        => 512 octets
    int transitions[NB_MAX_ETATS][NB_MAX_ENTREES];         => 65 536 octets
    char sortie[NB_MAX_ETATS][NB_MAX_ENTREES][LG_MAX_SORTIE]; => 2 097 152 octets
} automate;
```

On a donc bien une taille finale de 2 163 212 octets, soit plus de 2 Mo. On constate ici la limite de l'allocation statique de mémoire. On a prévu de gérer un maximum de 128 états et de 128 entrées différentes. Or, on constate que notre automate n'a que 3 états et 3 transitions. Si l'on avait alloué seulement la mémoire dont avait besoin, l'automate n'aurait fait que quelques centaines d'octets.

[h] Voici un automate possible pour la nouvelle machine à café :

```
4
0
20
0 1 1
0 2 2
0 r 0
0 c 0
0 t 0
1 1 2
1 2 3
1 r 0
1 c 1
1 t 1
2 1 3
2 2 3
2 r 0
2 c 2
2 t 2
3 1 3
3 2 3
3 r 0
3 c 0
3 t 0
13
0 1 credit:10c
0 2 credit:20c
1 1 credit:20c
1 2 credit:30c
1 r CLING(10)!
2 1 credit:30c
2 2 CLING(10)!-credit:30c
2 r CLING(20)!
3 1 CLING(10)!-credit:30c
3 2 CLING(20)!-credit:30c
3 r CLING(10+20)!
3 c Café_servi!
3 t Thé_servi!
```

[i] Le seul état final de l'automate Mystère est l'état 2. On peut quitter la simulation lorsqu'on atteint un état final en rajoutant dans la boucle while :*

```
        /* Si l'état suivant est final, quitter le programme */
        if (A->etats_finals[etat_courant]) {
            ecrire_sortie ("État final atteint !\n");
            return;
        }
```