Correction Partiel INF304 (2020–2021)

Remarque : suite à un problème de versionnement, la variable `tableau_entiers t` *n'est pas présentée sous forme de pointeur dans la fonction* `traitement()` *sur le sujet du partiel. Bravo à ceux qui ont repéré cette erreur. La correction qui suit prend un pointeur en entrée de la fonction* `traitement()`.

Exercice 1 (/2 pts)

La fonction `traitement()` supprime les doublons dans un tableau d'entiers.

Exercice 2 (/2 pts)

Le format des entrées de la méthode `traitement()` est une variable de type `tableau_entiers`, c'est-à-dire une structure contenant une séquence d'entiers de taille 10000 et un entier `taille` inférieur à 10000.

Exercice 3 (/4 pts)

```
#include <stdio.h>
#include "traitement.h"
#include "es_tableau.h"

int main(int argc, char ** argv) { // aussi : int main(int argc, char * argv[]) {
  tableau_entiers t;
  int i;
  FILE *f ;

  // Ouverture fichier tableau
  if (argc != 2) {
    printf("Usage: %s <fichier> \n", argv[0]) ; return 1;
  }
  f = fopen(argv[1], "r");
  if (f == NULL) {
    printf("Erreur lors de l'ouverture du fichier. \n"); return 2;
  }

  // Lecture et traitement
  lire_tableau(f, &t);
  fclose(f);
  traitement(&t) ; //Format pointeur optionnel

  // Affichage du résultat
  printf("Résultat du traitement: \n [");
  for(i = 0; i < t.taille; i++){
    printf("%d\t", t.tab[i]);
  }
  printf("]\n");
  return 0 ;
}
```

Exercice 4 (/4 pts)

Format : <Jeu de test> → <Sortie attendue>

Raisonnement en boîte-noire

On s'appuie sur le comportement attendu de la fonction `traitement()`*. On peut chercher des cas limites et des cas standards.*

Cas Limites

a) Tableau vide, doit rester tel quel : [ ] → [ ]
b) Singleton, doit rester tel quel : [1] → [1]
c) Que des doublons, doit donner un singleton : [1 1 1 1] → [1]

Cas Standards

d) 0 doublon dans un tableau de taille quelconque : [1 2 3 4] → [1 2 3 4]
e) 1 doublon en position quelconque : [1 2 1 3] → [1 2 3]
f) 1 doublon en début de séquence : [1 1 2 3] → [1 2 3]
g) 1 doublon en fin de séquence : [1 2 3 1] → [1 2 3]
h) Plusieurs doublons en position quelconque : [1 2 3 2 1 0] → [1 2 3 0]

Raisonnement en boîte-blanche

On s'appuie sur le code de la fonction `traitement()`*, en listant les différents chemins que l'on peut prendre en exécutant cette fonction. Les jeux de test précédents conviennent aussi à ce type de raisonnement.*

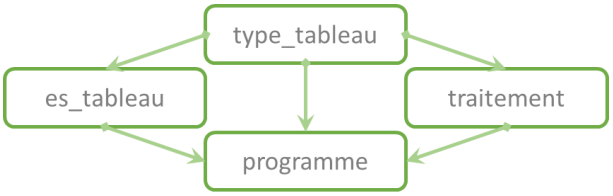
Parmi les chemins d'exécution pertinents, on trouve :

- Ne pas entrer dans la 1ère boucle for (l.8) : c'est le test (a)
  - Les autres tests entrent dans cette boucle *for*
- Ne pas entrer dans la 2ème boucle for (l.9) : c'est le test (b)
  - Les tests d'après entrent dans cette boucle *for*
- Ne pas entrer dans le *if* (l.10) : c'est le test (d)
  - Les tests (c, e, f, g, h) entrent dans le *if*
- Ne pas entrer dans la 3ème boucle for (l.11) : c'est le test (g)
  - Les tests, (c, e, f, h) entrent dans cette boucle *for*
- Avec le test (h), on fait un nombre arbitraire de tours dans chaque boucle for ; et on n'entre / n'entre pas dans le *if* en fonction des entiers parcourus.

On attendait au moins 4 jeux de tests, dont un cas limite (a ou b), le cas sans doublon (d) et un cas avec doublon (e, f, g ou h). On a retiré des points à :
- ceux qui proposaient des tests de robustesse ;
- ceux qui ne décrivaient pas le raisonnement derrière leurs jeux de tests.

Exercice 5 (/5 pts)

Question 1 (/2 pts)



Question 2 (/3 pts)

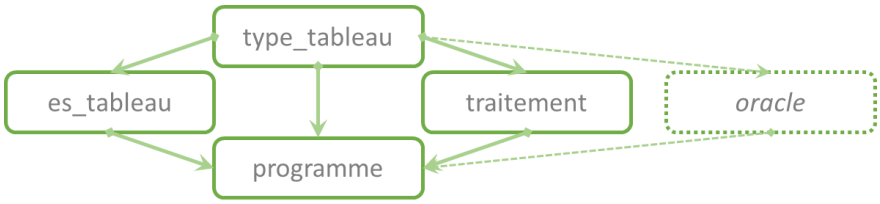
```
CC = clang

all: programme

# ##########################################################
# Compilation : Version Avec Motif
# %.o : %.c
#     $(CC) -c $<
#
# # Dépendances à conserver
# programme.o: traitement.h es_tableau.h type_tableau.h
#
# es_tableau.o: es_tableau.h type_tableau.h
#
# traitement.o: traitement.h type_tableau.h
# ##########################################################

# Compilation : Version Sans Motif
programme.o: programme.c traitement.h es_tableau.h type_tableau.h
	$(CC) -c $<
es_tableau.o: es_tableau.c es_tableau.h type_tableau.h
	$(CC) -c $<
traitement.o: traitement.c traitement.h type_tableau.h
	$(CC) -c $<


# Edition de liens
programme: programme.o es_tableau.o traitement.o
	$(CC) $^ -o $@

# (Optionnel)
clean:
	rm -f programme *.o
```

Exercice 6 (/3 pts)

- Fichier oracle.h : *signature* de la fonction `oracle_traitement` qui prend en entrée le tableau initial et le tableau après traitement et retourne un entier.

- Fichier oracle.c : *implémentation* de la fonction `oracle_traitement` qui vérifie que :
  - taille du tableau traité ≤ taille du tableau initial ;
  - le tableau traité contient `taille` éléments distincts ;
  - tous les éléments distincts du tableau initial sont contenus dans le tableau traité.

- Arbre des dépendances modifié :



- Modifications dans le programme de test :
  - `#include "oracle.h"`

- Modifications dans le Makefile :
  - `oracle.o : oracle.c type_tableau.h oracle.h`
    `$(CC) -c $<`
  - `programme.o : ... oracle.h`
  - `programme : ... oracle.o`