TD 5 : Testez, testez, testez ...

L'ensemble de ces exercices sont destinés à approfondir votre apprentissage
Merci de ne pas vous servir de ChatGpt ou Copilot

Assertion - Vous avez dit voyelles ? :

```
#include <stdio.h>
#include <stdlib.h>
#include <time.h>
#include <ctype.h>

enum alphabet
{
	A,B,C,D,E,F,G,H,I,J,K,L,M,N,O,P,Q,R,S,T,U,V,W,X,Y,Z
};

int main(int argc, char * argv[])
{
	for (int i = 1; i < argc; i++)
	{
		int toControl = toupper(argv[i] [0])  - 65;
		switch(toControl)
		{
			case A: printf("Voyelle - A \n"); break;
			case E: printf("Voyelle - E \n"); break;
			case I: printf("Voyelle - I \n"); break;
			case O: printf("Voyelle - O \n"); break;
			case U: printf("Voyelle - U \n"); break;
			case Y: printf("Voyelle - Y \n"); break;
			default: printf("Pas une voyelle %c \n",argv[i] [0]); break;
		}
	}
	return 0;
}
```

Sauvegardez ce programme dans un fichier : assertion.c
Compilation : gcc -g assertion.c -o main
Exécution : ./main a r e
Voyelle - A
Pas une voyelle - R
Voyelle - E

1. static_assertion

Ajoutez une static_assertion qui vérifie que l'alphabet est complet

2. assertion

Ajoutez :

- #include <assert.h> dans l'entête
- une assertion qui vérifie qu'il y a au moins un argument au lancement de l'exécutable
- une (ou plusieurs) assertion(s) qui vérifie(ent) que les arguments passés à l'exécutable sont valides

3. Jeu de test

Trouvez une liste d'arguments qui permettrait de couvrir 100% du code (code coverage)

Pour tester compilez avec :	**gcc –coverage assertion.c -o main** (– *coverage s'écrit avec deux tirets*)
Executez :	**assertion a T e** (c'est un exemple)
Recherchez :	**ls -l main-assertion.c.gcda** (ce fichier trace les chemins qui ont été exécutés)
Lancez :	**gcov main-assertion.gcda** (conversion en langage humain lisible)
Message :	Lines executed : 71.4% par exemple

A chaque nouvelle exécution du main avec d'autres arguments, un nouveau fichier gcda sera créé, relancez **gcov main-assertion.gcda** pour obtenir le résultat
Quel est le jeu de test qui permet de couvrir 100 % des chemins dans le code ?

TU avec Cunit - Des statistiques et des moyennes

1. Créer un programme

Réaliser un programme qui réalise des calculs sur une série de nombre de type float stockée dans un tableau (taille max déterminée via un define)

```
#include <stdio.h>
#include <stdlib.h>

#define TAILLE_SUITE	5
#define TRUE		1
#define FALSE		0

int main()
{
	float * suite = creerSuite();
	initSuite(suite);

	float stat = 7;
	ajouterEnTeteApresDecalage(suite,&stat);
	stat = 6;
	ajouterEnTeteApresDecalage(suite,&stat);
	stat = 5;
	ajouterEnTeteApresDecalage(suite,&stat);
	stat = 4;
	ajouterEnTeteApresDecalage(suite,&stat);
	stat = 3;
	ajouterEnTeteApresDecalage(suite,&stat);
	afficheSuite(suite);

	stat = 2;
	ajouterEnTeteApresDecalage(suite,&stat);
	stat = 1;
	ajouterEnTeteApresDecalage(suite,&stat);
	afficheSuite(suite);
	moyenne(suite);
	plusPetitElement(suite);
	plusGrandElement(suite);
	printf("Nbre element renseignés %d \n",nombreElementsRenseignesDansSuite(suite));

	free(suite);
}
```

Prototype de fonctions :

float * creerSuite()
	Créer un malloc de la taille de TAILLE_SUITE et le return

void initSuite(float * maSuite)
	Initialise l'ensemble des éléments de la suite à -1.0f

void afficheSuite(float * maSuite)
	Affiche tous les éléments de la suite : "Index %02d Valeur %.f \n"

int nombreElementsRenseignesDansSuite(float * maSuite)
	Return le nombre d'éléments de la suite qui ont été initialisés (<> de -1)

void ajouterEnTeteApresDecalage(float * maSuite, float * valeur)
	Décale tous les éléments de la liste vers la droite, le premier élément prend la nouvelle valeur

float moyenne(float * maSuite)
	Calcule la moyenne de tous les éléments (<> -1) et affiche un message avec le résultat

int plusPetitElement(float * maSuite)
	Affiche le plus petit élément de la suite (<> -1)

int plusGrandElement(float * maSuite)
	Affiche le plus grand élément de la suite (<> -1)

Exemple d'affichage (suite de longueur 5):
	Index	00	Valeur 1
	Index	01	Valeur 2
	Index	02	Valeur 3
	Index	03	Valeur 4
	Index	04	Valeur 5
	Moyenne : 3.00
	Plus petit : 1.00
	Plus grand : 5.00
	Nombre d'éléments renseignés : 5

Consignes :
	Vous utiliserez l'arithmétique des pointeurs
	Vous passerez l'ensemble des paramètres par référence
	Vous bouclerez (par exemple et si besoin) en utilisant *(p++)

2. Ajouts des TU

Ajoutez au début du source:

```
#include <CUnit/Basic.h>

#define TAILLE_SUITE	5
```

Déclarez une fonction :

```
void tu_Initialisation(void)
{
	CU_ASSERT_PTR_NOT_NULL(creerSuite());
}
```

Ajoutez au début du main :

```
// Init de CUnit
if (CUE_SUCCESS != CU_initialize_registry()) return CU_get_error();     // Initialisation du registre de tests
CU_pSuite pSuite = NULL;                                                // Déclaration d'une pSuite de tests
pSuite = CU_add_suite("Mes tests unitaires",NULL,NULL);
if (pSuite == NULL)
{
        CU_cleanup_registry();
        return CU_get_error();
}
```

Ajoutez à la fin du main :

```
if (CU_add_test(pSuite,"Init de la suite",tu_Initialisation) == NULL)      // Ajout test unitaire dans la pSuite
{
        CU_cleanup_registry();
        return CU_get_error();
}

CU_basic_set_mode(CU_BRM_VERBOSE);                          // Lancement des tests
CU_basic_run_tests();
CU_basic_show_failures(CU_get_failure_list()) ;

CU_cleanup_registry();                                      // Clean up des registres et autres data de CUnit
return CU_get_error();
```

Compilez :

```
gcc --coverage statistique_cu.c -o main -lcunit
```

Affichage :

Insérez au moins un test et vous obtiendrez cet affichage
Suite : Mes tests unitaires
Test : Init de la suite … passed
Run summary : un tableau avec des informations sur la réussite des tests

Ajoutez :

Ajout d'un élément (test de la fonction → ajouterEnTeteApresDecalage)

```
if (CU_add_test(pSuite,"Ajout d'un élément",tu_AjoutUnElement) == NULL)
{
        CU_cleanup_registry();
        return CU_get_error();
}
```

Vérifiera qu'un élément a bien été ajouté en testant la valeur située à l'indice 0 de la suite
Utilisation de CU_PASS/CU_FAIL

Réalisation d'un décalage (test de la fonction → ajouterEnTeteApresDecalage)

```
if (CU_add_test(pSuite,"Vérifier décalage de la suite si + de 5 éléments",tu_ProvoquerDecalage) == NULL)
{
        CU_cleanup_registry();
        return CU_get_error();
}
```

Ajoutez 6 valeurs à une suite, vérifiez que les valeurs sont intégrées à la suite et bien placées
Utilisation de CU_PASS/CU_FAIL

Comptage des éléments d'une suite (test de la fonction → nombreElementsRenseignesDansSuite)

Ajoutez le test suivant « Nbre éléments de la suite » → tu_CompterElement
Créez puis ajoutez 4 éléments à la suite pour vérifier que:

- le nombre d'éléments de la suite vaut bien 4
- le nombre d'éléments de la suite ne vaut pas 2
- que la fonction retourne bien « true »

Vérification du plus grand élément (test de la fonction → plusGrandElement/plusPetitElement)

Ajoutez le test suivant "Tests des fonctions plusPetit et plusGrand" → tu_PetitGrand

Créez une suite, initialisez là :

tester que la fonction plusGrandElement retourne bien FALSE

tester que la fonction plusPetitElement retourne bien FALSE

Ajoutez 2 éléments à cette suite :

tester que la fonction plusGrandElement retourne bien TRUE

tester que la fonction plusPetitElement retourne bien TRUE

Analysez :

La couverture de code existante en utilisation gcov comme vu en TD

Essayez d'approcher 100% ? Est-ce critique dans notre cas ?