DOSSIER DE DÉVELOPPEMENT LOGICIEL:

John LI, Nassim BEN DAALI
Groupe : 105

Table des matières :

1.Présentation brève du projet

Le projet en question est la SAé S1.01 effectué en période A à l'Université Rives-de-Seine en formation BUT Informatique, par les étudiants de 1er année (2022-2023) John LI et Nassim BEN DAALI, Groupe 105.

L'objectif de ce projet consistait à l'implémentation d'un besoin client. Ici, développer un interpréteur de commande permettant de gérer une formation universitaire. Tout en respectant des contraintes telles qu'un cahier des charges et des limites numériques.

Les huit commandes à prendre en compte vont permettre de définir la structure de la formation tels que les semestres, les matières, épreuves et leurs coefficients. Ainsi que les étudiants qui y sont inscrits, d'affecter des notes à ces étudiants et enfin de réaliser les tâches de fin de semestre et d'année (relevés de notes et des décisions de jury).

L'utilisateur pourra en entrée saisir des chaînes de caractères de formats attendus par les commandes en utilisant soit l'entrée standard (le clavier), ou par redirection d'un fichier texte sur l'entrée standard via l'invite de commande.
A partir des entrées, l'application devra faire l'appel des commandes entrées et produire les résultats en sortie. (l'écran)

Exemple :
Entrée : formation [un entier naturel compris entre MIN_UE et MAX_UE] (commande 1)
Sorties possibles:

- "Le nombre d'UE est incorrect"
- "Le nombre d'UE est deja defini"
- "Le nombre d'UE est defini"

A la suite de ces tâches, la nécessité de rédiger un dossier de développement logiciel afin de rendre compte du travail effectué.

2.Organisation des tests de l’application et le bilan des différents sprints

Afin de tester la validité des commandes nous avons d’abord commencé par essayer à la fin de chaque commande par une saisie clavier si elles étaient fonctionnelles ou non et en corrigeant au fur et à mesure les erreurs. Puis lorsque nous arrivons à la fin des commandes nécessaires à un sprint, nous essayons les sprints donnés par le prof afin d’avoir des tests plus poussés et encore une fois de valider notre programme. Ces sprints nous ont beaucoup aidé notamment les sprints erreurs car ils nous faisaient remarquer plusieurs petits problèmes que nous n’avions pas perçus avant. Nous avons réussi à régler ces problèmes en déboguant. Ensuite à l’aboutissement de notre programme nous avons essayé de nouveau les sprints envoyés par le prof afin de voir si aucun problème n'était survenu entre temps et à cela nous avons ajouté les sprints qui nous ont été fournis sur discord et qui eux essayaient des cas beaucoup plus extrêmes. A la réussite des tous ces sprints nous avons conclu que notre programme était fonctionnel pour le jour des recettes.

Bilan de validation des différents sprints :

Sprint 1 : Validé

Sprint 2 : Validé

Sprint 3 : Validé

Sprint 4 : Validé

3.Bilan de projet :

Les difficultés rencontrées :

La première difficulté a été de comprendre la structure “Formation” et de l’appliquer afin de programmer les commandes notamment comment la parcourir afin de récupérer ce dont on avait besoin. Particulièrement, comment imbriquer les boucles afin d’identifier les matières/épreuves.

Ensuite, la création d’une nouvelle structure “Etudiant” et comment l’intégrer à la structure “Formation” avec notamment le rangement des notes dans un tableau et de chercher leur emplacement afin de les ranger correctement.

L’ alignement de l’affichage du relevé et de la décision qui a demandé de s’adapter par rapport aux noms des matières… L’arrondi des notes causé par le printf posait problème car ne correspondait pas aux attentes de la commande.

Les réussites :

Les réussites ont été l’utilisation des pointeurs pour accéder à la structure.
Également, la création de fonctions intermédiaires afin d’éviter la redondance notamment pour la vérification des notes et coefficients.
La réussite majeure a été l’aboutissement du programme complet avec les huit commandes, ce qui signifie avoir réussi à surmonter toutes les difficultés rencontrées lors de ce projet.

Les améliorations possibles :

Les améliorations évidentes sont la création de fonctions intermédiaires supplémentaires notamment par exemple pour la recherche des matières et épreuves.

Également, la création de variables qui stockent les cheminements de la structure formation exemple : f->semestres[i] à stocker dans une variable afin d'éviter de longues lignes de code.

Pour finir, on aurait pu éliminer tous les avertissements de Visual Studio.

4.Annexes :

a. Listing du code source avec l'extension (Code Blocks)

```
/**
* @file Source.c
* @brief Projet Saé R1.01 : Gestion d'une formation universitaire
* @author John LI et Nassim BEN DAALI Grp 105
* @version 1.0
* @date Semestre 1 Période A
*/

#include <stdio.h>
#include <stdlib.h>
#include <string.h>
#include <assert.h>
#include <math.h>
#include <stdbool.h>
#pragma warning(disable:4996)

enum { // Toutes les valeurs constantes nécessaire au programme
	NB_SEMESTRES = 2,
	MIN_UE = 3,
	MAX_UE = 6,
	MAX_MATIERES = 10,
	MAX_EPREUVES = 5,
	MAX_ETUDIANTS = 100,
	MAX_CHAR = 30,
	MAX_NOTES = NB_SEMESTRES * MAX_MATIERES * MAX_EPREUVES
};

const float MIN_NOTE = 0.f, MAX_NOTE = 20.f;
typedef char CH30[MAX_CHAR + 1]; // MAX_CHAR VAUT 30
typedef unsigned int uint;

typedef struct {
	CH30 nomE;
	float coef[MAX_UE];
} Epreuve;
typedef struct {
	CH30 nomM;
	uint nbEpreuves;
	Epreuve epreuves[MAX_EPREUVES];
} Matiere;
typedef struct {
	uint nbMatieres;
	Matiere matieres[MAX_MATIERES];
```

```
} Semestre;
/*Création d'une structure Etudiant nécessaire pour stocker les attributs des étudiants*/
typedef struct {
	uint nbNotes; // nombre des notes
	float notes[MAX_NOTES]; //tableau qui sert à stocker le notes
	CH30 nomEtudiant; // nom de l'étudiant
} Etudiant;
typedef struct {
	uint nbEtudiant;
	uint nbUE; // nombre de coef, commun à toutes les épreuves
	Semestre semestres[NB_SEMESTRES];
	Etudiant etudiants[MAX_ETUDIANTS]; //La formation se compose du tableau des étudiants
} Formation;


/**
* @brief Initialise le nombre de matières, epreuves et leur coefficients, le nombre d'élèves et leur notes à -1 pour la formation f
* @param[in-out] Formation* f, pointeur de la Formation
* @return void
*/
void initFormation(Formation* f) {
	f->nbUE = 0; /*Initialise le nombre d'UE à 0*/
	for (int i = 0; i < NB_SEMESTRES; ++i) {
		f->semestres[i].nbMatieres = 0; /*Initialise le nombre de matières à 0*/
		for (int j = 0; j < MAX_MATIERES; ++j) {
			f->semestres[i].matieres[j].nbEpreuves = 0; /*Initialise le nombre d'épreuves à 0 pour chaque matière*/
		}
	}
	f->nbEtudiant = 0;
	for (int k = 0; k < MAX_ETUDIANTS; ++k) {
		f->etudiants[k].nbNotes = 0;/*Initialise le nombre de notes des étudiants à 0*/
		for (int l = 0; l < MAX_NOTES; ++l)
			f->etudiants[k].notes[l] = -1; /*Initialise les notes de chaque étudiant à -1*/
	}
}


/**
* @brief Vérifie si les coefficients du semestre de la formation f sont corrects
* @param[in] int semestre , le numéro de semestre
* @param[in-out] const Formation* f, pointeur de la Formation
* @return un bool : true si les coeff sont corrects et false si les coeffs sont incorrects
*/
bool verifcoeff(const Formation* f, int semestre) {
	int i, j, k;
	float sommecoef=0.;
	/* Parcours pour chaque UE les coefficients et en fait la somme */
	for (i = 1; i <= f->nbUE; ++i) { // Parcours les UE
		for (j = 0; j < f->semestres[semestre - 1].nbMatieres; ++j) { // Parcours les matières
```

```
                  for (k = 0; k < f->semestres[semestre - 1].matieres[j].nbEpreuves; ++k) { // Parcours
les épreuves
                        sommecoef = sommecoef + f->semestres[semestre -
1].matieres[j].epreuves[k].coef[i - 1]; // Somme de coefficients
                  }
                  k = 0;
            }
            if (sommecoef == 0.) {
                  return false;
            }
            sommecoef = 0;
            j = 0;
      }
      return true;
}

/**
* @brief Vérifie si les notes du semestre de la formation f sont corrects
* @param[in] int sem , le numéro de semestre
* @param[in-out] const Formation* f, pointeur de la Formation
* @return un bool : true si les coeff sont corrects et false si les coeffs sont incorrects

*/
bool verifnotes(const Formation* f, int sem, CH30 nom) {
      sem = sem - 1;
      int total = 0, totalnotes = 0;
      int j;
      for (int i = 0; i < f->semestres[sem].nbMatieres; ++i) // Parcours les matières
            total += f->semestres[sem].matieres[i].nbEpreuves; // Somme des épreuves
      for (j = 0; j < f->nbEtudiant; ++j) // Cherche l'étudiant
      if (strcmp(nom, f->etudiants[j].nomEtudiant) == 0)
                  break;
      if (j == f->nbEtudiant) { // Etudiant pas trouvé
            printf("Etudiant inconnu\n");
            return;
      }
      if (sem == 0) { // Compte combien il y a de notes dans le semestre 2
            for (int b = 0; b < 50; ++b)
                  if (f->etudiants[j].notes[b] != -1) // Parcours les notes
                        totalnotes += 1;
      }
      if (sem == 1) { // Compte combien il y a de notes dans le semestre 2
            for (int b = 50; b < 100; ++b)
                  if (f->etudiants[j].notes[b] != -1) // Parcour les notes
                        totalnotes += 1;
      }
      if (totalnotes < total) {
            return false;
      }
      else
```

```
            return true;
}

/**
* @brief Définie le nombre d'UE et vérifie si déja défini ou incorrect
* @param[in-out] Formation* f, pointeur de la Formation
* @return void

*/
void formation(Formation* f) {
      int a;
      scanf("%d", &a);
      // Vérifie si le nombre d''UE est déja défini ou incorrect
      if (a < 3 || a > 6)
            printf("Le nombre d'UE est incorrect\n");
      else if (f->nbUE >= 1)
            printf("Le nombre d'UE est deja defini\n");
      else {
            printf("Le nombre d'UE est defini\n");
            f->nbUE = a; // Définie le nombre d'UE
      }
}

/**
* @brief Ajoute une épreuve à la formation f
* @param[in-out] Formation* f, pointeur de la Formation
* @return void

*/
void epreuves(Formation* f) {
      int numsem;
      float coeff[MAX_UE], somcoeff = 0;
      CH30 nommatiere, nomepreuve;
      scanf(" %d %s %s", &numsem, &nommatiere, &nomepreuve);
      /*
      Demande la saisie d'un réel(coefficients) autant de fois que le nombre d'UE
      Puis fait la somme des coefficients ajouté ou renvoie un message d'erreur si le coefficient et
négatif
      */
      for (int i = 0; i < f->nbUE; ++i) {
            scanf("%f", &coeff[i]);
            if (coeff[i] >= 0)
                  somcoeff += coeff[i];
            else {
                  printf("Au moins un des coefficients est incorrect\n");
                  return;
            }
      }
      if (f->nbUE == 0) {
            printf("Le nombre d'UE n'est pas defini\n");
            return;
```

```
	}
	else if (numsem < 1 || numsem > NB_SEMESTRES) {
		printf("Le numero de semestre est incorrect\n");
		return;
	}
	else {
		numsem -= 1;
		int n = 0;
		int k = 0;
		/* Vérifie si il existe une même matière dans le tableau des matières en implémentant n dans le cas inverse */
		while (n < f->semestres[numsem].nbMatieres && strcmp(nommatiere, f->semestres[numsem].matieres[n].nomM) != 0) {
			n++;
		}
		/* Si la matière existe déjà vérifie si il existe une même épreuve au sein de cette matière en implémentant k dans le cas inverse */
		if (n < f->semestres[numsem].nbMatieres) {
			k = 0;
			while (k < f->semestres[numsem].matieres[n].nbEpreuves && strcmp(nomepreuve, f->semestres[numsem].matieres[n].epreuves[k].nomE) != 0)
				++k;
		}
		/* Matière déjà existante et épreuve déja existante */
		if (n < f->semestres[numsem].nbMatieres && k < f->semestres[numsem].matieres[n].nbEpreuves)
			printf("Une meme epreuve existe deja\n");
		else if (somcoeff == 0) {
			printf("Au moins un des coefficients est incorrect\n");
			return;
		}

		else if (n == f->semestres[numsem].nbMatieres) { // Pas trouvé, on mémorise la matière et l'epreuve
			strcpy(f->semestres[numsem].matieres[n].nomM, nommatiere);
			strcpy(f->semestres[numsem].matieres[n].epreuves[k].nomE, nomepreuve);
			for (int l = 0; l < MAX_UE; ++l) // Ajout des coefficients à l'épreuve
				f->semestres[numsem].matieres[n].epreuves[k].coef[l] = coeff[l];
			++f->semestres[numsem].nbMatieres; // Une matiere de plus !
			++f->semestres[numsem].matieres[n].nbEpreuves; // Une epreuve de plus
			printf("Matiere ajoutee a la formation\n");
			printf("Epreuve ajoutee a la formation\n");
		}
		else { //Matière trouvé mais nouvelle épreuve
			n = 0;
			while (n < f->semestres[numsem].nbMatieres) { //On cherche l'emplacement de la matière
				if (strcmp(nommatiere, f->semestres[numsem].matieres[n].nomM) != 0)
					++n;
				else
					break;
			}
			strcpy(f->semestres[numsem].matieres[n].epreuves[k].nomE, nomepreuve); //On mémorise
```

```
l'épreuve
                    for (int l = 0; l < MAX_UE; ++l) // Ajout des coefficients à l'épreuve
                          f->semestres[numsem].matieres[n].epreuves[k].coef[l] = coeff[l];
                    ++f->semestres[numsem].matieres[n].nbEpreuves; // Une épreuve de plus
                    printf("Epreuve ajoutee a la formation\n");
              }
       }
}

/**
* @brief Vérifie si les coefficients sont corrects ou non
* @param[in-out] Formation* f, pointeur de la Formation f
* @return void
*/
void coefficient(Formation* f) {
       int i, j, k, sem;
       float sommecoef = 0;
       scanf("%d", &sem);
       if (f->nbUE == 0) {
              printf("Le nombre d'UE n'est pas defini\n");
              return;
       }
       if (sem < 1 || sem > NB_SEMESTRES) {
              printf("Le numero de semestre est incorrect\n");
              return;
       }
       else if (f->semestres[sem - 1].nbMatieres == 0) {
              printf("Le semestre ne contient aucune epreuve\n");
              return;

       }
       /* Parcours les coefficients pour chaque UE en faisant la somme des coefficients à chaque passage
*/
       for (i = 1; i <= f->nbUE; ++i) { // Parcours les UE
              for (j = 0; j < f->semestres[sem - 1].nbMatieres; ++j) { // Parcours les matières
                    for (k = 0; k < f->semestres[sem - 1].matieres[j].nbEpreuves; ++k) { // Parcours les
épreuves
                          sommecoef = sommecoef + f->semestres[sem - 1].matieres[j].epreuves[k].coef[i -
1]; // Somme des coefficients
                    }
                    k = 0;
              }
              if (sommecoef == 0) {
                    printf("Les coefficients d'au moins une UE de ce semestre sont tous nuls\n");
                    return 0;
              }
              sommecoef = 0;
              j = 0;
       }
       printf("Coefficients corrects\n");
}
```

```c
/**
* @brief Ajout d'une note à la formation f
* @param[in-out] Formation* f, pointeur de la Formation f
* @return void
*/
void note(Formation* f) {
	int sem;
	float note;
	CH30 nom, matiere, epreuve;
	scanf("%d %s %s %s %f", &sem, &nom, &matiere, &epreuve, &note);
	if (f->nbUE == 0) {
		printf("Le nombre d'UE n'est pas defini\n");
		return;
	}
	if (sem < 1 || sem > NB_SEMESTRES) {
		printf("Le numero de semestre est incorrect\n");
		return;
	}
	else if (note < MIN_NOTE || note > MAX_NOTE) {
		printf("Note incorrecte\n");
		return;
	}
	sem = sem - 1;
	int emplacement;
	int i;
	int j;
	for (i = 0; i < f->semestres[sem].nbMatieres;) { // Cherche si la matière existe déjà
		if (strcmp(matiere, f->semestres[sem].matieres[i].nomM) == 0) {
			for (j = 0; j < f->semestres[sem].matieres[i].nbEpreuves && strcmp(epreuve,
f->semestres[sem].matieres[i].epreuves[j].nomE) != 0; ++j); // Cherche si l'épreuve existe déjà
			break;
		}
		else
			++i;
	}
	if (i == f->semestres[sem].nbMatieres)
		printf("Matiere inconnue\n");
	else if (j == f->semestres[sem].matieres[i].nbEpreuves)
		printf("Epreuve inconnue\n");
	else {
		int k;
		for (k = 0; k < f->nbEtudiant && strcmp(nom, f->etudiants[k].nomEtudiant) != 0; ++k); //
Cherche l'étudiant
		/*
		Permet de trouver l'emplacement de l'épreuve correspondante dans toute la formation
		Explication :
		Chaque unité correspond à une épreuve différente
		5 épreuves max par matière donc chaque 5 unités correspond à un changement de matière
		50 épreuves max par semestre donc à partir de 50 unités on change de semestre
```

```
            Permet donc de savoir le semestre et la matière dans laquelle on se trouve et l'épreuve
correspondante
            */
            emplacement = (sem * 50) + (i * 5) + j;
            if (f->etudiants[k].notes[emplacement] != -1) { // note déjà définie
                  printf("Une note est deja definie pour cet etudiant\n");
                  return;
            }
            if (k == f->nbEtudiant) { // Etudiant pas trouvé
                  strcpy(f->etudiants[k].nomEtudiant, nom); // On le mémorise
                  ++f->nbEtudiant; // un étudiant en plus
                  printf("Etudiant ajoute a la formation\n");
            }
            f->etudiants[k].notes[emplacement] = note; // Ajout de la note
            ++f->etudiants[k].nbNotes; // une note en plus
            printf("Note ajoutee a l'etudiant\n");
      }
}


/**
* @brief Affiche si les notes sont corrects ou non
* @param[in-out] Formation* f, pointeur de la Formation f
* @return void
*/
void notes(Formation* f) {
      int sem;
      CH30 nom;
      scanf("%d %s", &sem, &nom);
      if (f->nbUE == 0) {
            printf("Le nombre d'UE n'est pas defini\n");
            return;
      }
      if (sem < 1 || sem > NB_SEMESTRES) {
            printf("Le numero de semestre est incorrect \n");
            return;
      }
      sem = sem - 1;
      int total = 0, totalnotes = 0;
      int j;
      for (int i = 0; i < f->semestres[sem].nbMatieres; ++i) // Cherche combien il existe d'épreuve au
total
            total += f->semestres[sem].matieres[i].nbEpreuves;
      for (j = 0; j < f->nbEtudiant; ++j) // Cherche l'étudiant
            if (strcmp(nom, f->etudiants[j].nomEtudiant) == 0)
                  break;
      if (j == f->nbEtudiant) {
            printf("Etudiant inconnu\n");
            return;
      }
      if (sem == 0) { // Compte combien il y a de notes dans le semestre 1
```

```
		for (int b = 0; b < 50; ++b)
			if (f->etudiants[j].notes[b] != -1) // Parcours les notes
				totalnotes += 1;
	}
	if (sem == 1) { // Compte combien il y a de notes dans le semestre 2
		for (int b = 50; b < 100; ++b)
			if (f->etudiants[j].notes[b] != -1) // Parcours les notes
				totalnotes += 1;
	}
	if (totalnotes < total) { // Nombre de notes différents du nombre d'épreuves
		printf("Il manque au moins une note pour cet etudiant\n");
		return;
	}
	else
		printf("Notes correctes\n");
}


/**
* @brief Affiche le releve des notes de l'étudiant
* @param[in-out] Formation* f, pointeur de la Formation f
* @return void
*/
void releve(const Formation* f) {
	int sem, cpt = 0, pos_nomE;
	CH30 nom;
	scanf("%d %s", &sem, &nom);

	/*Vérifie le nombre d'UE et le numéro de semestre*/
	if (f->nbUE == 0) {
		printf("Le nombre d'UE n'est pas defini\n");
		return;
	}
	if (sem < 1 || sem > NB_SEMESTRES) {
		printf("Le numero de semestre est incorrect\n");
		return;
	}

	/*Verifie si l'étudiant existe en comparant son nom avec tous les noms*/
	for (int i = 0; i < f->nbEtudiant; ++i) {
		if (strcmp(f->etudiants[i].nomEtudiant, nom) != 0) {
			cpt += 1;
		}
		else if (strcmp(f->etudiants[i].nomEtudiant, nom) == 0) {
			pos_nomE = i;
		}
	}
	if (cpt == f->nbEtudiant) {
		printf("Etudiant inconnu\n");
		return;
	}
```

```
	/*Vérification des coefficients et notes*/
	else if (verifcoeff(f, sem) == false) {
		printf("Les coefficients de ce semestre sont incorrects\n");
		return;
	}
	else if (verifnotes(f, sem, nom) == false) {
		printf("Il manque au moins une note pour cet etudiant\n");
		return;
	}


	sem = sem - 1;
	int maxcaractere = strlen("Moyennes"); /*Pour les alignements*/
	float totalnotes[MAX_UE], totalcoeff[MAX_UE]; /*Deux tableaux stockant à chaque position(=UE) le
total des notes et coefficients*/

	for (int i = 0; i < f->nbUE; ++i) {
		totalnotes[i] = 0.;
		totalcoeff[i] = 0.;
	}
	/*Recherche de la plus grande chaîne de caractères*/
	for (int i = 0; i < f->semestres[sem].nbMatieres; ++i) {
		if (maxcaractere < strlen(f->semestres[sem].matieres[i].nomM))
			maxcaractere = strlen(f->semestres[sem].matieres[i].nomM);
	}
	/*Pour respecter les alignements*/
	for (int i = 0; i < maxcaractere + 1; ++i)
		printf(" ");
	for (int o = 1; o <= f->nbUE; ++o) {
		printf(" UE%d ", o);
	}
	printf("\n");


	for (int i = 0; i < f->semestres[sem].nbMatieres; ++i) {
		float notes[MAX_UE], coeff[MAX_UE]; /*Deux tableaux stockant à chaque position(=UE) les notes
et coefficients*/
		for (int i = 0; i < f->nbUE; ++i) {
			notes[i] = 0.;
			coeff[i] = 0.;
		}
		/*<Récupère les notes et coefficients pour chaque matière d'une UE*/
		for (int j = 0; j < f->semestres[sem].matieres[i].nbEpreuves; ++j) {
			int emplacement = (sem * 50) + (i * 5) + j; /*Pour obtenir l'emplacement de la note*/
			for (int w = 0; w < f->nbUE; ++w) {
				notes[w] += f->semestres[sem].matieres[i].epreuves[j].coef[w] *
f->etudiants[pos_nomE].notes[emplacement];
				coeff[w] += f->semestres[sem].matieres[i].epreuves[j].coef[w];
			}
		}
```

```
            /*Pour respecter les alignements*/
            printf("%s ", f->semestres[sem].matieres[i].nomM);
            for (int t = 0; t < maxcaractere - strlen(f->semestres[sem].matieres[i].nomM); ++t)
                  printf(" ");

            /*Calcule le total des notes et coefficients pour chaque UE */
            for (int g = 0; g < f->nbUE; ++g) {
                  totalnotes[g] += notes[g];
                  totalcoeff[g] += coeff[g];

                  /*Affichage de la moyenne pondérée de chaque matière calculée pour chaque UE*/
                  if (coeff[g] == 0)
                        printf("  ND ");
                  else if (notes[g] / coeff[g] < 10.) /*Ajout d'un espace si la note est inférieure à
10*/
                        printf(" %.1f ", truncf(notes[g] / coeff[g] * 10.0) / 10.0); /*truncf permet de
tronquer sur une décimale , ex: entrée = 2.5 , sortie = 2.0, permet d'éviter l'arrondie*/
                  else
                        printf("%.1f ", truncf(notes[g] / coeff[g] * 10.0) / 10.0);
            }
            printf("\n");
      }
      /*Pour respecter les alignements*/
      printf("--\n");
      printf("Moyennes ");
      for (int t = 0; t < maxcaractere - strlen("Moyennes"); ++t)
            printf(" ");

      /*Affiche le calcul de la moyenne pondérée pour chaque UE*/
      for (int m = 0; m < f->nbUE; ++m) {
            if (totalnotes[m] / totalcoeff[m] < 10.)
                  printf(" %.1f ", truncf(totalnotes[m] / totalcoeff[m] * 10.0) / 10.0); /*ex: le total
des notes divisé par coefficients à la position du tableau 0 correspondent à la moyenne de l'UE1*/
            else
                  printf("%.1f ", truncf(totalnotes[m] / totalcoeff[m] * 10.0) / 10.0);
      }
      printf("\n");
}


/**
* @brief Affiche la décision positive pour l'étudiant s'il a acquis plus de la moitié des UE
de l'année et redouble dans le cas contraire
* @param[in-out] Formation* f, pointeur de la Formation f
* @return void
*/
void decision(const Formation* f) {
      CH30 nom;
      int cpt = 0, nomE;
      int maxcaractere = strlen("Moyennes annuelles");
```

```
scanf("%s", &nom);

/*Vérifie le nombre d'UE*/
if (f->nbUE == 0) {
	printf("Le nombre d'UE n'est pas defini\n");
	return;
}
/*Verifie si l'étudiant existe*/
for (int i = 0; i < f->nbEtudiant; ++i) {
	if (strcmp(f->etudiants[i].nomEtudiant, nom) != 0) {
		cpt += 1;
	}
	else if (strcmp(f->etudiants[i].nomEtudiant, nom) == 0) {
		nomE = i;
	}
}
if (cpt == f->nbEtudiant) {
	printf("Etudiant inconnu\n");
	return;
}

/*Vérification des coefficients et notes*/
for (int k = 1; k <= NB_SEMESTRES; ++k) {
	if (verifcoeff(f, k) == false) {
		printf("Les coefficients d'au moins un semestre sont incorrects\n");
		return;
	}
	else if (verifnotes(f, k, nom) == false) {
		printf("Il manque au moins une note pour cet etudiant\n");
		return;
	}
}

/*Pour respecter les alignements*/
for (int i = 0; i < maxcaractere + 1; ++i)
	printf(" ");
for (int o = 1; o <= f->nbUE; ++o) {
	printf(" UE%d ", o);
}
printf("\n");

/*Quatre tableaux stockant à chaque position(=UE) le total des notes, coefficients,
la moyenne annuelle des notes et coefficients*/
float totalnotes[MAX_UE], totalcoeff[MAX_UE], moyenneAnNote[MAX_UE], moyenneAnCoeff[MAX_UE];
for (int i = 0; i < f->nbUE; ++i) {
	totalnotes[i] = 0.;
	totalcoeff[i] = 0.;
	moyenneAnNote[i] = 0.;
	moyenneAnCoeff[i] = 0.;
}
float notes[MAX_UE], coeff[MAX_UE];
```

```
	for (int k = 1; k <= 2; ++k) {
		printf("S%d", k);
		for (int i = 0; i < maxcaractere - 1; ++i)
			printf(" ");
		for (int i = 0; i < f->semestres[k - 1].nbMatieres; ++i) {
			for (int i = 0; i < f->nbUE; ++i) {
				notes[i] = 0.;
				coeff[i] = 0.;
			}

			/*<Récupère les notes et coefficients pour chaque matière d'une UE*/
			for (int j = 0; j < f->semestres[k - 1].matieres[i].nbEpreuves; ++j) {
				int emplacement = ((k - 1) * 50) + (i * 5) + j; /*Pour obtenir l'emplacement de
la note*/
				for (int w = 0; w < f->nbUE; ++w) {
					notes[w] += f->semestres[k - 1].matieres[i].epreuves[j].coef[w] *
f->etudiants[nomE].notes[emplacement];
					coeff[w] += f->semestres[k - 1].matieres[i].epreuves[j].coef[w];
				}
			}

			/*Calcule le total des notes et coefficients pour chaque UE */
			for (int g = 0; g < f->nbUE; ++g) {
				totalnotes[g] += notes[g];
				totalcoeff[g] += coeff[g];
			}
		}

		/*Affiche le calcul de la moyenne pondérée pour chaque UE*/
		for (int m = 0; m < f->nbUE; ++m) {
			if (totalnotes[m] / totalcoeff[m] < 10.)
				printf(" %.1f ", truncf(totalnotes[m] / totalcoeff[m] * 10.0) / 10.0);
			else
				printf("%.1f ", truncf(totalnotes[m] / totalcoeff[m] * 10.0) / 10.0);
		}
		printf("\n");

		/*Calcule la moyenne des notes et coefficients pour chaque UE */
		for (int h = 0; h < f->nbUE; ++h) {
			moyenneAnNote[h] += totalnotes[h] / totalcoeff[h];
			moyenneAnCoeff[h] += 1.0;
		}

		/*Réinitialise le total des notes et coefficients*/
		for (int i = 0; i < f->nbUE; ++i) {
			totalnotes[i] = 0.;
			totalcoeff[i] = 0.;
		}
	}

	/*Pour respecter les alignements*/
```

```
printf("--\n");
printf("Moyennes annuelles");
for (int t = 0; t < maxcaractere - strlen("Moyennes anuelles"); ++t) {
	printf(" ");
}

/*Affiche le calcul de la moyenne annuelle pondérée pour chaque UE*/
for (int b = 0; b < f->nbUE; ++b) {
	if (moyenneAnNote[b] / moyenneAnCoeff[b] < 10.) {
		printf(" %.1f ", truncf(moyenneAnNote[b] / moyenneAnCoeff[b] * 10.0) / 10.0);
	}
	else
		printf("%.1f ", truncf(moyenneAnNote[b] / moyenneAnCoeff[b] * 10.0) / 10.0);
}

/*Pour respecter les alignements*/
printf("\n");
printf("Acquisition");
for (int t = 0; t < maxcaractere - strlen("Acquisition"); ++t) {
	printf(" ");
}

int cptAcq = 0;
char* virgule = "";
printf(" ");
/*Affiche les UE acquis*/
for (int b = 0; b < f->nbUE; ++b) {
	if (b <= f->nbUE - 1 && moyenneAnNote[b] / moyenneAnCoeff[b] >= 10.) {
		printf("%sUE%d", virgule, b + 1);
		virgule = ", ";
	}
	else
		cptAcq += 1;
}
if (cptAcq == f->nbUE)
	printf(" Aucune");

/*Pour respecter les alignements*/
printf("\n");
printf("Devenir");
for (int t = 0; t < maxcaractere - strlen("Devenir"); ++t) {
	printf(" ");
}

/*Affiche la décision */
int cptUE = 0;
for (int b = 0; b < f->nbUE; ++b) {
	if (moyenneAnNote[b] / moyenneAnCoeff[b] >= 10.) {
		cptUE += 1;
	}
}
```

```
	/*L'étudiant passe dans l'année suivante s'il a acquis plus de la moitié des UE de l'année et redouble dans le cas contraire*/
	if (cptUE > f->nbUE / 2)
		printf(" Passage");
	else
		printf(" Redoublement");
	printf("\n");

}

/**
* @brief Lit à partir d'une entrée standard puis fait l'appel des fonctions entrées. Doit produire des résultats en sortie
*/
int main() {
	Formation f;
	initFormation(&f);
	char b[31] = "";

	do {
		scanf("%s", b);

		if (strcmp(b, "formation") == 0) {
			formation(&f);
		}
		else if (strcmp(b, "epreuve") == 0) {
			epreuves(&f);
		}
		else if (strcmp(b, "coefficients") == 0) {
			coefficient(&f);
		}
		else if (strcmp(b, "note") == 0) {
			note(&f);
		}
		else if (strcmp(b, "notes") == 0) {
			notes(&f);
		}
		else if (strcmp(b, "releve") == 0) {
			releve(&f);
		}
		else if (strcmp(b, "decision") == 0) {
			decision(&f);
		}
	} while (strcmp(b, "exit") != 0);
}
```

b. Trace d’exécution du test du sprint de plus haut niveau atteint

Sprint 4 :

- **in-sp4-base.txt**

Résultat :

```
Le nombre d'UE est defini
Matiere ajoutee a la formation
Epreuve ajoutee a la formation
Epreuve ajoutee a la formation
Matiere ajoutee a la formation
Epreuve ajoutee a la formation
Epreuve ajoutee a la formation
Matiere ajoutee a la formation
Epreuve ajoutee a la formation
Epreuve ajoutee a la formation
Matiere ajoutee a la formation
Epreuve ajoutee a la formation
Epreuve ajoutee a la formation
Etudiant ajoute a la formation
Note ajoutee a l'etudiant
Note ajoutee a l'etudiant
Note ajoutee a l'etudiant
Note ajoutee a l'etudiant
Note ajoutee a l'etudiant
Note ajoutee a l'etudiant
Note ajoutee a l'etudiant
Note ajoutee a l'etudiant
Etudiant ajoute a la formation
Note ajoutee a l'etudiant
Note ajoutee a l'etudiant
Note ajoutee a l'etudiant
Note ajoutee a l'etudiant
```

```
Note ajoutee a l'etudiant
Note ajoutee a l'etudiant
Note ajoutee a l'etudiant
Note ajoutee a l'etudiant
Etudiant ajoute a la formation
Note ajoutee a l'etudiant
Note ajoutee a l'etudiant
Note ajoutee a l'etudiant
Note ajoutee a l'etudiant
Note ajoutee a l'etudiant
Note ajoutee a l'etudiant
Note ajoutee a l'etudiant
Note ajoutee a l'etudiant
                    UE1  UE2  UE3
S1                 11.2 10.2 13.0
S2                  9.3  8.1 13.0
--
Moyennes annuelles 10.2  9.1 13.0
Acquisition        UE1, UE3
Devenir            Passage
                    UE1  UE2  UE3
S1                  8.0  9.8  5.0
S2                  9.3  8.1 13.0
--
Moyennes annuelles  8.6  8.9  9.0
Acquisition        Aucune
Devenir            Redoublement
                    UE1  UE2  UE3
S1                 11.2 10.2 13.0
S2                 17.6 16.8 15.9
--
Moyennes annuelles 14.4 13.5 14.4
Acquisition        UE1, UE2, UE3
Devenir            Passage
```

- **in-sp4-erreur.txt**

Résultat :

```
Le nombre d'UE est defini
Le nombre d'UE est defini
Matiere ajoutee a la formation
Matiere ajoutee a la formation
Epreuve ajoutee a la formation
Epreuve ajoutee a la formation
Epreuve ajoutee a la formation
Epreuve ajoutee a la formation
Matiere ajoutee a la formation
Matiere ajoutee a la formation
Epreuve ajoutee a la formation
Epreuve ajoutee a la formation
Epreuve ajoutee a la formation
Epreuve ajoutee a la formation
Etudiant ajoute a la formation
Etudiant ajoute a la formation
Note ajoutee a l'etudiant
Note ajoutee a l'etudiante
Note ajoutee a l'etudiant
Note ajoutee a l'etudiant
Note ajoutee a l'etudiant
Note ajoutee a l'etudiant
Note ajoutee a l'etudiant
Note ajoutee a l'etudiant
Les coefficients d'au moins un semestre sont incorrects
Les coefficients d'au moins un semestre sont incorrects
Matiere ajoutee a la formation
Matiere ajoutee a la formation
Epreuve ajoutee a la formation
Epreuve ajoutee a la formation
Epreuve ajoutee a la formation
Epreuve ajoutee a la formation
Note ajoutee a l'etudiant
Note ajoutee a l'etudiant
Il manque au moins une note pour cet etudiant
Il manque au moins une note pour cet etudiant
Note ajoutee a l'etudiant
Note ajoutee a l'etudiant
```

```
Etudiant inconnu
Etudiant inconnu
                   UE1  UE2  UE3
                   UE1  UE2  UE3
S1                 11.2 10.2 13.0
S1                 11.2 10.2 13.0
S2                 18.0 12.0 14.0
S2                 18.0 12.0 14.0
--
--
Moyennes annuelles 14.6 11.1 13.5
Moyennes annuelles 14.6 11.1 13.5
Acquisition        UE1, UE2, UE3
Acquisition        UE1, UE2, UE3
Devenir            Passage
```