Correction.
L'objectif pédagogique des colles est de :

- faire un bilan avec l'étudiant de son niveau en EI, en lui suggérant éventuellement des révisions pour le partiel (même niveau que les colles) ;
- récompenser l'assiduité avec l'évaluation sur des exos simples vus en TD ;
- leur faire apprendre par coeur la résolution de sous-problèmes simples qu'ils rencontreront tout au long du L1.

Vous disposez de 15 minutes, en tout, pour évaluer un étudiant. Cela se déroule de la façon suivante :

- appel d'un étudiant qui n'appartient pas à votre groupe. il y a un point de rdv de l'étage pour aller chercher les étudiants.
- tirage au sort d'une feuille d'exercice (il ne doit pas écrire dessus)
- vérification de son identité
- écriture du programme
- trace du programme à partir d'une initialisation arbitraire des données du problèmes que vous donnez. Le déroulement du programme doit être donné à voix haute pour accompagner l'écriture de la trace. L'initialisation des données doit être différente de celle par défaut et l'étudiant doit répercuter toutes les modifs dans le programme. Cela fait partie de l'évaluation.
- ne pas donner la note à l'étudiant, car ca peut être discuté après.

La notation proposée se veut très simple en espérant éviter les écarts de notation entre évaluateurs d'une part, et entre exercices avec différents niveaux de difficulté d'autre part. L'idée est qu'un étudiant dispose exactement de 15 minutes en tout et sa note est fonction du nombre de points qu'il a réussi à faire correctement (c'est du tout ou rien), qu'il soit aidé ou pas : plus il est aidé plus il perd de temps ; certains des progs les plus faciles sont les plus longs, etc. Le barême est le suivant :

- programme vide 4 points
- programme de l'exercice avec commentaires 6 points (ou l'algo en français 4 points sans points d'indentation et de trace possibles)
- indentation 2 points
- modif du programme pour initialisation des données fournie par l'évaluateur 2 points
- trace 4 points (une trace incomplète par manque de temps mais ne posant pas de soucis doit être comptée juste)
- 2 points laissés à l'appréciation de l'évaluateur

Par exemple, un étudiant ayant tout fait avec de l'aide aura entre 18 et 20. Cette note vaut entre 1/5 et 1/10 de la note finale.

La séance d'évaluation dure 15 minutes en tout. Il vous est demandé d'écrire le programme correspondant à l'exercice, avec l'ensemble des commentaires pertinents et une indentation correcte. Une trace de ce programme vous sera alors demandée, avec une initialisation des données du problème fournie par l'évaluateur. Lors de l'écriture de la trace, vous lirez le déroulement du programme à voix haute.

1 Quel temps fait-il ?

En vous inspirant du codage du genre vu en cours (1 pour MASCULIN, 2 pour FÉMININ), proposez un codage pour indiquer si le temps est COUVERT, ENSOLEILLÉ ou PLUVIEUX. Écrivez un programme principal qui, étant donné le temps affecté à une variable, affiche le temps qu'il fait.

Correction.

```
Exemple : Soit temps = 1 avec 0 COUVERT, 1 ENSOLEILLÉ et 2 PLUVIEUX.
Affiche ‘‘Le temps est ensoleillé.’’

Algo :
    /* Codage :
       0 COUVERT
       1 ENSOLEILLE
       2 PLUVIEUX
    */
si temps = 0 alors affiche couvert
si temps = 1 alors affiche ensoleille
si temps = 2 alors affiche pluvieux

les cas sont mutuellement exclusifs.

/* declaration de fonctionnalites supplementaires */
#include <stdlib.h> /* EXIT_SUCCESS */
#include <stdio.h> /* printf */

/* declaration constantes et types utilisateurs */

/* declaration de fonctions utilisateurs */

/* fonction principale */
int main()
{
    /* declaration et initialisation variables */
    int temps = 1; /* temps qu'il fait */

    /* si temps = 0 alors affiche couvert */
    if(temps == 0) /* COUVERT */
```

```
    {
        printf("Le temps est couvert.\n");
    }

    /* si temps = 1 alors affiche ensoleille */
    if(temps == 1) /* ENSOLEILLE */
    {
        printf("Le temps est ensoleille.\n");
    }

    /* si temps = 2 alors affiche pluvieux */
    if(temps == 2) /* PLUVIEUX */
    {
        printf("Le temps est pluvieux.\n");
    }

    /* valeur fonction */
    return EXIT_SUCCESS;
}

/* implantation de fonctions utilisateurs */
```

La séance d'évaluation dure 15 minutes en tout. Il vous est demandé d'écrire le programme correspondant à l'exercice, avec l'ensemble des commentaires pertinents et une indentation correcte. Une trace de ce programme vous sera alors demandée, avec une initialisation des données du problème fournie par l'évaluateur. Lors de l'écriture de la trace, vous lirez le déroulement du programme à voix haute.

2 Dans 1 seconde, il sera exactement...

Écrire un programme principal qui, étant donnée une heure représentée sous la forme de 3 variables pour les heures, `h`, les minutes, `m` et les secondes, `s`, affiche l'heure qu'il sera 1 seconde plus tard. Il faudra envisager tous les cas possibles pour le changement d'heure. Deux exemples de sortie sont :

```
L'heure actuelle est : 23h12m12s
Dans une seconde, il sera exactement : 23h12m13s

L'heure actuelle est : 23h59m59s
Dans une seconde, il sera exactement : 00h00m00s
```

Correction.

```
algo :
- affiche l'heure actuelle
- ajoute une seconde
- si tour du cadran des secondes alors
    - remise a 0 des secondes
    - il est une minute de plus
    - si tour du cadran des minutes alors
        - remise a 0 des minutes
        - il est une heure de plus
        - si tour du cadran des heures alors
         - remise a zero des heures
- affiche la nouvelle heure
```

```
1   /* declaration de fonctionnalites supplementaires */
2   #include <stdlib.h> /* EXIT_SUCCESS */
3   #include <stdio.h> /* printf */
4
5   /* declaration constantes et types utilisateurs */
6
7   /* declaration de fonctions utilisateurs */
8
9   /* fonction principale */
10  int main()
11  {
12      /* soient 23h59m59s */
13      int h = 23;
```

```
14      int m = 59;
15      int s = 59;
16
17      /* affiche l'heure actuelle */
18      printf("L'heure actuelle est : %dh%dm%ds\n",h,m,s);
19
20      /* une seconde de plus */
21      s = s + 1;
22
23      if(s == 60) /* tour du cadran des secondes */
24      {
25          /* remise a 0 */
26          s = 0;
27
28          /* une minute de plus */
29          m = m + 1;
30
31          if(m == 60) /* tour du cadran des minutes */
32          {
33              /* remise a 0 */
34              m = 0;
35
36              /* une heure de plus */
37              h = h + 1;
38
39              if(h == 24) /* tour du cadran des heures */
40              {
41                  /* remise a zero */
42                  h = 0;
43              }
44          }
45      }
46      /* h,m,s contiennent l'heure une seconde plus tard */
47
48      /* affiche l'heure */
49      printf("Dans une seconde, il sera exactement : %dh%dm%ds\n",h,m,s);
50
51      /* valeur fonction */
52      return EXIT_SUCCESS;
53  }
54
55  /* implantation de fonctions utilisateurs */
56
```

La séance d'évaluation dure 15 minutes en tout. Il vous est demandé d'écrire le programme correspondant à l'exercice, avec l'ensemble des commentaires pertinents et une indentation correcte. Une trace de ce programme vous sera alors demandée, avec une initialisation des données du problème fournie par l'évaluateur. Lors de l'écriture de la trace, vous lirez le déroulement du programme à voix haute.

3 Le minimum de 3 valeurs

Soient 3 variables `a`, `b`, `c`, initialisées à des valeurs quelconques. Écrire un programme qui calcule et affiche à l'écran le minimum des 3 valeurs.

Correction. Nous donnons une version "standard" pour la recherche d'un extremum : une val par défaut. On parcourt les autres valeurs et on modifie en conséquence. Servira plus tard quand calcul extremum d'une série, bien qu'en général il soit plus difficile d'écrire le code initialisant l'extremum avec la première valeur de la série et on préfère initialiser avec une valeur limite (i.e. min = +infini).

```
algorithme :
soit min = a /* valeur par defaut */
si b < min /* b plus petit que min courant */
  /* b est le min courant */
  min = b
/* min contient min(a,b) */
si c < min /* c plus petit que min courant */
  /* c est le min courant */
  min = c
/* min contient min(min(a,b),c) = min(a,b,c) */
affiche min
```

La séance d'évaluation dure 15 minutes en tout. Il vous est demandé d'écrire le programme correspondant à l'exercice, avec l'ensemble des commentaires pertinents et une indentation correcte. Une trace de ce programme vous sera alors demandée, avec une initialisation des données du problème fournie par l'évaluateur. Lors de l'écriture de la trace, vous lirez le déroulement du programme à voix haute.

4 Calcul de $\Sigma_1^n i$

Écrire un programme qui calcule et affiche la somme des entiers de 1 à n : $\Sigma_1^n i$. n est un entier quelconque.

Correction.

```
/* déclaration de fonctionalités supplémentaires */
#include <stdlib.h> /* EXIT_SUCCESS */
#include <stdio.h> /* printf */

/* déclaration constantes et types utilisateurs */

/* déclaration de fonctions utilisateurs */

/* fonction principale */
int main()
{
    /* déclaration et initialisation variables */
    int somme = 0; /* élément neutre pour l'addition */
    int i; /* var. de boucle */

    for(i = 1;i <= 4;i = i + 1) /* i allant de 1 à 4 */
    {
        /* ajoute i à la somme partielle */
        somme = somme + i;
    }
    /* i > 4 */

    /* somme vaut 0 + 1 + 2 + 3 + 4 */
    printf("somme = %d\n",somme);

    return EXIT_SUCCESS;
}

/* implantation de fonctions utilisateurs */
```

La séance d'évaluation dure 15 minutes en tout. Il vous est demandé d'écrire le programme correspondant à l'exercice, avec l'ensemble des commentaires pertinents et une indentation correcte. Une trace de ce programme vous sera alors demandée, avec une initialisation des données du problème fournie par l'évaluateur. Lors de l'écriture de la trace, vous lirez le déroulement du programme à voix haute.

5 Affichage d'un rectangle d'étoiles

Écrire un programme qui affiche, étant données deux variables, `longueur` et `largeur`, initialisées à des valeurs strictement positives quelconques, un rectangle d'étoiles ayant pour longueur `longueur` étoiles et largeur `largeur` étoiles. Deux exemples d'exécution, avec deux initialisations différentes, sont les suivants :

```
Affichage d'un rectangle d'etoiles de longueur 10 et largeur 5.
**********
**********
**********
**********
**********

Affichage d'un rectangle d'etoiles de longueur 6 et largeur 3.
******
******
******
```

Correction. Durée 3/4 d'heure ?

Les algos sont à faire (les extraire du code).

Vous pouvez dans un premier temps supprimer la boucle la plus imbriquée en leur demandant d'afficher un rectangle de longueur exactement "**********".

```
/* declaration de fonctionnalites supplementaires */
#include <stdlib.h> /* EXIT_SUCCESS */
#include <stdio.h> /* printf */

/* declaration constantes et types utilisateurs */

/* declaration de fonctions utilisateurs */

/* fonction principale */
int main()
{
    /* declaration et initialisation variables */
    int largeur = 3; /* largeur du rectangle en nb d'etoiles */
    int longueur = 6; /* longueur du rectangle en nb d'etoiles */
    int i; /* var. de boucle */
    int j; /* var. de boucle */
```

```
    printf("Affichage d'un rectangle d'etoiles de longueur %d et largeur %d.\n",longu

    for(i = 0;i < largeur;i = i + 1) /* chaque ligne d'étoiles */
    {
        /* affiche longueur etoiles */
        for(j = 0;j < longueur;j = j + 1) /* chaque colonne d'etoiles */
        {
            /* affiche une etoile */
            printf("*");
        }
        /* j >= longueur */

        /* passe a la ligne suivante */
        printf("\n");
    }
    /* i >= largeur */

    return EXIT_SUCCESS;
}

/* implantation de fonctions utilisateurs */
```

La séance d'évaluation dure 15 minutes en tout. Il vous est demandé d'écrire le programme correspondant à l'exercice, avec l'ensemble des commentaires pertinents et une indentation correcte. Une trace de ce programme vous sera alors demandée, avec une initialisation des données du problème fournie par l'évaluateur. Lors de l'écriture de la trace, vous lirez le déroulement du programme à voix haute.

6 Affichage d'un demi-carré d'étoiles

Écrire un programme qui affiche, étant donnée la variable, `cote`, initialisée à une valeur quelconque, un demi-carré d'étoiles (triangle rectangle isocèle) ayant pour longueur de côté `cote` étoiles. Deux exemples d'exécution, avec deux initialisations différentes, sont les suivants :

```
Affichage d'un demi-carre d'etoiles de cote 6.
*
**
***
****
*****
******

Affichage d'un demi-carre d'etoiles de cote 2.
*
**
```

Correction. Durée 3/4 d'heure ?

Les algos sont à faire (les extraire du code).

```
/* declaration de fonctionnalites supplementaires */
#include <stdlib.h> /* EXIT_SUCCESS */
#include <stdio.h> /* printf */

/* declaration constantes et types utilisateurs */

/* declaration de fonctions utilisateurs */

/* fonction principale */
int main()
{
    /* declaration et initialisation variables */
    int cote = 2; /* cote du demi-carré en nb d'etoiles */
    int i; /* var. de boucle */
    int j; /* var. de boucle */

    printf("Affichage d'un demi-carre d'etoiles de cote %d.\n",cote);

    for(i = 1;i <= cote;i = i + 1) /* chaque numero de ligne d'étoiles */
```

```
    {
        /* affiche autant d'etoiles que le numero de ligne */
        for(j = 0;j < i;j = j + 1) /* chaque colonne d'etoiles */
        {
            /* affiche une etoile */
            printf("*");
        }
        /* j >= i */

        /* passe a la ligne suivante */
        printf("\n");
    }
    /* i > cote */

    return EXIT_SUCCESS;
}

/* implantation de fonctions utilisateurs */
```

La séance d'évaluation dure 15 minutes en tout. Il vous est demandé d'écrire le programme correspondant à l'exercice, avec l'ensemble des commentaires pertinents et une indentation correcte. Une trace de ce programme vous sera alors demandée, avec une initialisation des données du problème fournie par l'évaluateur. Lors de l'écriture de la trace, vous lirez le déroulement du programme à voix haute.

7 Initialisation des cases d'un tableau à zéro

Soit un tableau d'entiers, déclaré d'une taille quelconque. Écrire un programme qui :
- affiche le tableau non initialisé
- initialise chaque case à zéro
- affiche le tableau initialisé

Un exemple d'exécution pour un tableau de taille 5 est :

```
affichage du tableau non initialise :
tab[0] = 134513308
tab[1] = -1208832012
tab[2] = 134518316
tab[3] = -1076845768
tab[4] = 134513753
affichage du tableau initialise :
tab[0] = 0
tab[1] = 0
tab[2] = 0
tab[3] = 0
tab[4] = 0
```

Correction. L'exemple plus haut correspond à l'exécution sur un PC 32 bits. Bien insister sur les valeurs arbitraires des variables non initialisées, qui dépendent des programmes qui ont été exécutés avant etc. On les représente par le symbole ' ?' dans la trace.

```
algo:
- affichage du tableau non initialise
- pour chaque case du tableau :
  - affecte 0 a la case
- affichage du tableau initialise


/* declaration de fonctionnalites supplementaires */
#include <stdlib.h> /* EXIT_SUCCESS */
#include <stdio.h> /* printf */

/* declaration constantes et types utilisateurs */

/* declaration de fonctions utilisateurs */

/* fonction principale */
```

```
int main()
{
    /* declaration et initialisation variables */
    int tab[5]; /* tableau a initialiser */
    int i; /* var. de boucle */

    /* affichage du tableau non initialise */
    printf("affichage du tableau non initialise :\n");
    for(i = 0;i < 5;i = i + 1) /* chaque case du tableau */
    {
        /* affichage de la case */
        printf("tab[%d] = %d\n",i,tab[i]);
    }
    /* i >= 5 */

    /* initialisation des cases a 0 */
    for(i = 0;i < 5;i = i + 1) /* chaque case du tableau */
    {
        /* initialisation de la case a 0 */
        tab[i] = 0;
    }
    /* i >= 5 */

    /* affichage du tableau initialise */
    printf("affichage du tableau initialise :\n");
    for(i = 0;i < 5;i = i + 1) /* chaque case du tableau */
    {
        /* affichage de la case */
        printf("tab[%d] = %d\n",i,tab[i]);
    }
    /* i >= 5 */

    return EXIT_SUCCESS;
}

/* implantation de fonctions utilisateurs */
```

La séance d'évaluation dure 15 minutes en tout. Il vous est demandé d'écrire le programme correspondant à l'exercice, avec l'ensemble des commentaires pertinents et une indentation correcte. Une trace de ce programme vous sera alors demandée, avec une initialisation des données du problème fournie par l'évaluateur. Lors de l'écriture de la trace, vous lirez le déroulement du programme à voix haute.

8 Calcul de la somme des éléments d'un tableau

Soit un tableau d'entiers, initialisé à une taille et des valeurs quelconques. Écrire un programme qui calcule et affiche à l'écran la somme des éléments du tableau.

Correction.

```
/* declaration de fonctionnalites supplementaires */
#include <stdlib.h> /* EXIT_SUCCESS */
#include <stdio.h> /* printf */

/* declaration constantes et types utilisateurs */

/* declaration de fonctions utilisateurs */

/* fonction principale */
int main()
{
    /* declaration et initialisation variables */
    int tab[5] = {2,-4,8,12,-1};
    int somme = 0; /* element neutre pour l'addition */
    int i; /* var. de boucle */

    for(i = 0;i < 5;i = i + 1) /* chaque case du tableau */
    {
        /* ajoute la case a la somme partielle */
        somme = somme + tab[i];
    }
    /* i >= 5 */

    /* somme vaut 0 + 2 - 4 + 8 + 12 - 1 */
    printf("somme = %d\n",somme);

    return EXIT_SUCCESS;
}

/* implantation de fonctions utilisateurs */
```

La séance d'évaluation dure 15 minutes en tout. Il vous est demandé d'écrire le programme correspondant à l'exercice, avec l'ensemble des commentaires pertinents et une indentation correcte. Une trace de ce programme vous sera alors demandée, avec une initialisation des données du problème fournie par l'évaluateur. Lors de l'écriture de la trace, vous lirez le déroulement du programme à voix haute.

9 Calcul du minimum d'un tableau

Soit un tableau d'entiers, initialisé à une taille et des valeurs quelconques. Écrire un programme qui calcule et affiche à l'écran la valeur minimum des éléments du tableau.

Correction.

```
/* declaration de fonctionnalites supplementaires */
#include <stdlib.h> /* EXIT_SUCCESS */
#include <stdio.h> /* printf */
#include <limits.h> /* INT_MAX */

/* declaration constantes et types utilisateurs */

/* declaration de fonctions utilisateurs */

/* fonction principale */
int main()
{
    /* declaration et initialisation variables */
    int tab[5] = {2,-4,8,12,-1};
    int min = INT_MAX; /* +infini >= valeurs tableau*/
    int i; /* var. de boucle */

    for(i = 0;i < 5;i = i + 1) /* chaque case du tableau */
    {
        if(tab[i] < min) /* plus petit que min courant */
        {
            /* nouveau min courant */
            min = tab[i];
        }
    }
    /* i >= 5 */

    /* min contient le minimum de tous les elements du tableau */
    printf("le minimum des elements du tableau est : %d\n",min);

    return EXIT_SUCCESS;
}
```

```
/* implantation de fonctions utilisateurs */
```