Université Paris-Nord — L1
Institut Galilée — Année 2009-2010
Licence 1 - 1er semestre

Éléments d'informatique, rattrapage

Correction. IMPORTANT dans 2.2 "bientôt l'été" : indiquer aux étudiants dans les salles d'examen qu'ils doivent définir une variable entière `saison` dont les constantes symboliques PRINTEMPS, ETE, AUTRE représentent les valeurs possibles.

RDV salle C102 pour remettre les copies à la fin de l'examen puis salle C213 pour les corrections de copies.

Durée : 3 heures.

Documents autorisés : Aucun.

Recommandations : Un barème vous est donné à titre indicatif afin de vous permettre de gérer votre temps. La notation prendra en compte à la fois la syntaxe et la sémantique de vos programmes, c'est-à-dire qu'ils doivent compiler correctement. Une fois votre programme écrit, il est fortement recommandé de le faire tourner à la main sur un exemple pour s'assurer de sa correction.

1 Somme des éléments d'un tableau (*4 points*)

Nous voulons écrire un programme qui calcule la somme des éléments d'un tableau d'entiers. Une partie du programme est déjà écrite, et Pippo pense qu'il ne reste plus qu'à écrire la partie qui calcule effectivement la somme. Voici son programme :

```
1   #include <stdlib.h> /* EXIT_SUCCESS */
2   #include <stdio.h> /* printf, scanf */
3
4   /* déclaration constantes et types utilisateurs */
5   #define TAILLE 3
6
7   /* Fonction principale */
8   int main()
9   {
10    int t[TAILLE] = {12,10,15}; /* tableau à sommer */
11
12
13
14    /* calcul de la somme (À FAIRE) */
15
```

```
16
17
18
19
20      /* affichage du résultat */
21      printf("La somme est %d\n", somme);
22
23      /* valeur fonction */
24      return EXIT_SUCCESS;
25   }
```

1. La compilation échoue. Expliquer pourquoi, quelle étape de la compilation échoue précisément et ce qu'il manque pour que la compilation réussisse.
2. Expliquer le fonctionnement de l'instruction `#define`.
3. Compléter le programme pour qu'il calcule effectivement la somme des éléments du tableau (pour une taille de tableau arbitraire).

Un exemple de sortie du programme pour le tableau donné dans le code est :

```
La somme est 37
```

Correction.

1. (1.5pt) dont 1 pt pour "déclaration de la variable `somme`" + 0.5 pt pour "analyse sémantique" ou 0.25 pt si "avant la génération du code objet" et/ou "avant l'édition de lien".
2. 1 pt si juste. 0.5 pt si seulement notion de constantes et pas de mise en relation avec la substitution au moment de la compilation.
3. 1.5pt pour le code juste (compilation correction quel que soit TAILLE). Sinon, maxi 1 pt :
   - pas de boucle -> 0
   - 0.5pt une (et une seule) boucle (for, while accepté)
   - 0,5pt la boucle parcours effectivement un tableau de taille TAILLE
   - pas de points en moins ou en plus pour la déclaration de la variable de boucle (redite sous une autre forme de la première question) ou pour l'initialisation de somme à zéro.

2 Sans fonctions

Il est demandé de résoudre les deux problèmes suivants sans définir de fonctions utilisateurs. L'ensemble du code sera à écrire dans la fonction principale `main`.

2.1 For ou while (*4 points*)

Écrire le programme qui :
- demande à l'utilisateur d'entrer les valeurs d'un tableau d'entiers de taille `N` (une constante symbolique)
- puis cherche si le tableau contient l'entier zéro et affiche l'indice de la première case nulle s'il y en a une ou `Pas de case nulle` sinon.

Deux exemples d'exécution sont les suivants (pour `N` valant 4) :

```
Saisissez 4 entiers : 8 31 1 -12
Pas de case nulle

Saisissez 4 entiers : 2 -5 4 0
Case nulle d'indice 3
```

Correction.

- 1 point. pour la saisie correcte on ne compte pas faux si pas de &. Constante symbolique comptée à part.
- 2.5pt. Deux structures admises pour la seconde partie : soit utiliser un booleen "trouve" comme dans d'autres exercices du meme type, soit utiliser le depassement de borne de boucle comme test. Dans les deux cas while puis if -> 1pt. Si ils se debrouillent pour ça fonctionne correctement avec un for au lieu du while ok. Ensuite 0.5 pour une boucle qui termine. 1pt pour l'affichage correct : 0.5 point pour le cas sans zéro, 0.5 point pour les cas avec zéro. Compter faux les cas où il y a plusieurs affichages.
- 0.5pt si définit et utilise correctement la constante symbolique (si elle n'est que définie et qu'ils ne traitent pas les boucles compter juste quand même par contre si les boucles sont là et que la taille du tableau n'est pas donnée par N compter faux !).

```
#include <stdlib.h> /* EXIT_SUCCESS */
#include <stdio.h> /* printf, scanf */

/* déclaration constantes et types utilisateurs */
#define N 4 /* taille du tableau utilisateur */

/* déclaration de fonctions utilisateurs */

int main()
{
    int tab[N]; /* tableau à initialiser par l'utilisateur */
   int i; /* var. de boucle */

    /* demande saisie de N entiers*/
    printf("Saisissez %d entiers : ", N);

    /* saisie des elts du tableau (N entiers) */
    for(i = 0; i < N; i = i + 1) /* chaque case du tableau */
    {
        /* saisie valeur */
        scanf("%d",&tab[i]);
    }
    /* i >= N */

    /* recherche zero */
    i = 0;
    while ((i < N) && (tab[i] != 0))
    {
       i = i + 1;
    }
    /* i >= N  ou bien tab[i] == 0 */
```

```
    if (i < N)
    {
        printf("Cas nulle d'indice %d\n", i);
    }
    else
    {
        printf("Pas de case nulle\n");
    }

    return EXIT_SUCCESS;
}
```

2.2 Bientôt l'été (*5 points*)

Écrire un programme qui demande à l'utilisateur de saisir une date dans l'année et affiche :
- `C'est le printemps`, si la date est au printemps ;
- `Déjà l'été` si la date du jour est en été :
- `Ce n'est ni le printemps, ni l'été`, si la date n'est ni au printemps ni en été ;

Vous utiliserez trois constantes symboliques pour représenter les trois périodes différentes de l'année : `PRINTEMPS`, `ETE`, `AUTRE`.

Vous séparerez clairement la partie saisie, la partie calcul de la saison et la partie affichage.

La date sera saisie sous la forme de deux entiers : l'un entre 1 et 31 (inclus) pour le jour dans le mois et l'autre entre 1 et 12 (inclus) pour le numéro du mois. On suppose que la saisie est une date correcte.

Le printemps commence le 20 mars, l'été le 21 juin, l'automne le 22 septembre.

Exemples d'exécutions :

```
Date : 21 10
Ce n'est ni le printemps, ni l'ete

Date : 11 6
C'est le printemps

Date : 21 6
C'est l'ete
```

Correction.
- 1pt saisie (sur-récompensé).
- 4 points pour calcul et affichage en simulant le programme de l'etudiant. Si ce programme est trop compliqué (des boucles ou autre) c'est zéro à cette partie.
  - 0.5pt si fonctionne pour mois < 3 (jour quelconque)
  - 0.5pt si fonctionne pour mois > 9 (jour quelconque)
  - 0.5pt si mois 4,5 au printemps (jour quelconque)
  - 0.5pt si mois 7,8 en été (jour quelconque)
  - 1.5pt : 0,5 point par traitement correct, à un jour près, des jours de transition dans les mois 3,6,9. (1-19 mars autre et 21-31 mars printemps etc...)
  - 0.5 point si chacun des trois jours de début de saison sont bien dans la saison (20 mars an printemps, 21 juin été, 22 septembre).
- Si correct mais affichage au milieu des tests ou saisie au milieu des tests enlever 0.5 pt.

– 0.5 pt en moins si il y a des variables non declarées, pas de points en moins pour les scanf, 0.5 point en moins si pas de constantes symboliques.

```
#include <stdlib.h> /* EXIT_SUCCESS */
#include <stdio.h> /* printf, scanf */

/* déclaration constantes et types utilisateurs */
#define TAILLE 3
#define PRINTEMPS 1
#define ETE 2
#define AUTRE 42

/* Fonction principale */
int main()
{
    int jour = 0;
    int mois = 0;
    int saison = AUTRE;

    /* saisie */
    printf("Saisir la date : ");
    scanf("%d", &jour);
    scanf("%d", &mois);


    /* calcul de la saison (on peut faire plus cours en initialisant saison a AUTRE) */
    if (mois < 3)
    {
        saison = AUTRE;
    }
    if (mois == 3) {
        if (jour < 20) {
            saison = AUTRE;
        }
        else
        {
            saison = PRINTEMPS;
        }
    }
    if ((mois > 3) && (mois < 5))
    {
        saison = PRINTEMPS;
    }
    if (mois == 6)
    {
        if (jour < 21)
        {
            saison = PRINTEMPS;
        }
        else
        {
```

```
            saison = ETE;
        }
    }
    if ((mois > 6) && (mois < 9))
    {
        saison = ETE;
    }
    if (mois == 9)
    {
        if (jour < 22) {
            saison = ETE;
        }
        saison = AUTRE;
    }
    if (mois > 9)
    {
        saison = AUTRE;
    }


    /* affichage */
    if (saison == AUTRE)
    {
        printf("Ce n'est ni le printemps, ni l'ete\n");
    }
    if (saison == PRINTEMPS)
    {
        printf("C'est le printemps\n");
    }
    if (saison == ETE)
    {
        printf("C'est l'ete\n");
    }
    /* valeur fonction */
    return EXIT_SUCCESS;
}
```

3 Trace d'un programme avec fonctions (*3 points*)

Simulez l'exécution du programme suivant, en réalisant sa **trace**, comme cela a été vu en TD et en cours.

```
1  #include <stdlib.h> /* EXIT_SUCCESS */
2  #include <stdio.h> /* printf, scanf */
3
4  /* declarations constantes et types utilisateurs */
5
6  /* declarations de fonctions utilisateurs */
7  int mystere(int n);
8
```

```
 9  int main()
10  {
11      int x = 2;
12      int res;
13
14      res = mystere(x);
15      printf("mystere(%d) = %d\n", x, res);
16
17      return EXIT_SUCCESS;
18  }
19
20  /* definitions de fonctions utilisateurs */
21  int mystere(int n)
22  {
23     if (n > 1)
24     {
25         return n * mystere(n - 1);
26     }
27     return 1;
28  }
```

Correction.

- 0.5pt pour l'identification des variables du main et leur initialisation
- 0.5pt pour le premier appel de fonction (avec ou sans titre) ligne 14 et identification de l'unique variable (n).
- 0.5 pt pour le passage par valeur (colonne n initialisée à 2) dans cet appel.
- 0.5pt pour la hauteur de pile (appel imbriqué).
- 0.5 pt si le if est correctement suivi dans les deux cas (faux si un seul appel).
- 0.5 pt si res est affecté après l'appel initial a mystere (effet de l'affectaction), même si pas le bon entier (si pas valeur entière c'est zéro).
- 0.5 pt si l'affichage est correct (et donc res vaut bien 2).

`main()`

ligne	x	res	Affichage
initialisation	2	?	
14			
14		2	
15			mystere(2) = 2
17	renvoie EXIT_SUCCESS		

`mystere(2)`

ligne	n	Affichage
initialisation	2	
25		
25	renvoie 2	

`mystere(1)`

ligne	n	Affichage
initialisation	1	
27	renvoie 1	

4 Écriture de fonctions (*4 × 1 point*)

1. Écrire une fonction qui prend en entrée un entier n et calcule la somme :

$$\sum_{k=1}^{n} \frac{1}{k^2}$$

2. Écrire la procédure `rect` qui n'a pas d'entrée et affiche le motif suivant :

```
*****
*   *
*****
```

3. Écrire la procédure `rectangle` qui prend en entrée un entier `largeur` et un entier `hauteur` et affiche un rectangle plein d'étoiles de `hauteur` lignes sur `largeur` colonnes.
4. Écrire une fonction `volume_cylindre` qui prend en entrée un rayon réel `r` et une hauteur réelle `h` et renvoie le volume du cylindre de rayon `r` et de hauteur `h`. vous ferez appel à une fonction `calcul_surface`, supposée déjà définie, qui calcule la surface d'un cercle à partir de son rayon.

Bonus. Écrire une procédure `boite` qui affiche un rectangle de `hauteur` lignes sur `largeur` colonnes, comme la fonction `rectangle` (question 3), mais creux, comme dans l'exemple de la question 2.

Correction. Difficultées diverses mais c'est voulu, on compte : 0.5 pt prototype correct et 0.5 pt code correct (ou 0.25 si quasi-correct ie boucles dans questions 1 et 3).

```
double sommation(int n)
{
  int k;
  double somme = 0;
  for (i = 1; i <= n; i = i + 1)
  {
    somme = somme + 1.0 / (k*k); /* Attention au double : 1.0 */
  }
  return somme;
}

void rect()
{
  printf("*****\n");
  printf("*   *\n");
  printf("*****\n");
}

void rectangle(int hauteur, int largeur)
```

```
{
  int i; /* variable de boucle ligne */
  int j; /* variable de boucle colonne */

  for (i = 0; i < hauteur; i = i + 1)/* pour chaque ligne */
  {
    /* afficher une ligne d'etoiles */
    for (j = 0; j < largeur; j = j + 1) /* pour chaque colonne */
    {
      printf("*"); /* afficher etoile */
    }
    printf("\n"); /* fin de la ligne */
  }
}

double volume_cylindre(double r, double h)
{
  return h * calcul_surface(r);
}
```

Pour la question bonus on compte 2 points si rigoureusement juste. Sinon rien.

```
void boite(int hauteur, int largeur)
{
  int i; /* variable de boucle ligne */
  int j; /* variable de boucle colonne */

  for (i = 0; i < hauteur; i = i + 1)/* pour chaque ligne */
  {
    for (j = 0; j < largeur; j = j + 1) /* pour chaque colonne */
    {
      if ((0 == i)
          || (hauteur - 1 == i)
          || (0 == j)
          || (largeur - 1 == j))
      {
         printf("*"); /* bord : etoile */
      }
      else
      {
        printf(" "); /* centre : espace */
      }
    }
    printf("\n"); /* fin de la ligne */
  }
}
```