Université Paris-Nord — L1
Institut Galilée — Année 2011-2012
Licence 1 - 1er semestre

Éléments d'informatique : partiel de mi-semestre

Durée : 3 heures.

Documents autorisés : Aucun.

Recommandations : Un barème vous est donné à titre indicatif afin de vous permettre de gérer votre temps. Vous pouvez traiter les questions dans l'ordre de votre choix, en faisant bien apparaître la numérotation alphabétique. Les premières questions sont plus faciles que les dernières. Il est recommandé de faire tourner à la main sur un exemple vos programmes pour s'assurer qu'il sont corrects.

1 Étude de programmes et questions de cours (*8 points*)

1.1 Questions de cours

Question A. Rappeler les cinq étapes de la compilation. 1 pt 9 min

Question B. Expliquer le fonctionnement de l'instruction `#define`. Vous pouvez prendre comme exemple le programme de la figure 1. 1 pt 9 min

```
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, 15, 10}; /* tableau */
11
12
13
14    /* calcul du maximum (À FAIRE) */
15
16
17
18
19
20    /* affichage du résultat */
21    printf("Le maximum est %d\n", maximum);
22
23    /* valeur fonction */
24    return EXIT_SUCCESS;
25  }
```

FIGURE 1 – Calcul du maximum d'un tableau (à compléter)

1.2 Maximum des éléments d'un tableau

Nous voulons écrire un programme qui calcule le maximum des éléments d'un tableau d'entiers positifs. Une partie du programme est déjà écrite, figure 1, et Pippo pense qu'il ne reste plus qu'à écrire la partie qui calcule effectivement le maximum.

Question C. Avant de continuer Pippo veut tester son programme, mais la compilation échoue. Expliquer pourquoi, quelle étape de la compilation échoue précisément et ce qu'il manque pour que la compilation réussisse.

1 pt
9 min

Question D. Compléter le programme pour qu'il calcule effectivement le maximum des éléments du tableau (pour n'importe quel tableau contenant des entiers positifs et de taille `TAILLE` fixée arbitrairement).

2 pt
18 min

Exemple. Pour le tableau donné dans le code, une fois terminé, le programme affichera :

```
Le maximum est 15
```

1.3 Trace d'un programme

Question E. Simuler l'exécution du programme figure 2, en réalisant sa **trace**.

3 pt
27 min

Rappel. La trace représente l'exécution du programme par un tableau à $n+2$ colonnes : la première colonne contient le numéro de la ligne exécutée ; les n colonnes suivantes, les variables du programme (n à déterminer) et la dernière colonne, l'affichage réalisé par le programme à l'écran. La trace commence par une ligne nommant les colonnes, puis la ligne reportant les valeurs initiales des variables, après quoi seules les lignes modifiant l'état mémoire du programme ou l'affichage sont à reporter dans le tableau. L'utilisateur doit saisir trois entiers : considerer, comme indiqué en commentaires, qu'il saisit 1 puis 2 puis 3.

```
1  #include <stdlib.h> /* EXIT_SUCCESS */
2  #include <stdio.h> /* printf, scanf */
3
4  /* declarations de constantes et types utilisateurs */
5
6  /* declarations de fonctions utilisateurs */
7
8  /* fonction principale */
9  int main()
10 {
11     int terme;
12     int raison;
13     int n;
14     int i;
15     int serie = 0;
16
17     printf("Entrez le premier terme, la raison puis le nombre de termes : ");
18     scanf("%d",&terme);  /* 1 est saisi par l'utilisateur */
19     scanf("%d",&raison); /* 2 est saisi par l'utilisateur */
20     scanf("%d",&n);      /* 3 est saisi par l'utilisateur */
21
22     for (i = 0; i < n; i = i + 1)
23     {
24         serie = serie + terme;
25         terme = terme * raison;
26     }
27
28     printf("somme des %d premiers termes de la suite : %d\n", n, serie);
29
30     return EXIT_SUCCESS;
31 }
```

FIGURE 2 – Faire la trace du programme

2 Jours d'automne (*4 points*)

Cette année votre premier semestre se déroule sur trois saisons : l'été qui se termine le 22 septembre, l'automne qui commence le 23 septembre et se termine le 21 décembre et l'hiver qui commence le 22 décembre.

Soient deux variables entières `jour` et `mois`, représentant une date entre le premier septembre (1/9) et le 31 décembre (31/12) et que vous initialiserez à des valeurs de votre choix, prises dans cet intervalle (votre programme doit fonctionner pour n'importe quel choix correct de date).

Question F. Écrire un programme qui :

1. définit trois constantes symboliques pour représenter par trois nombres différents les trois saisons : `ETE`, `AUTOMNE`, `HIVER`. — 0.5 pt, 4 min
2. affiche la date représentée par les variables `jour` et `mois` — 0.5 pt, 4 min
3. trouve la saison correspondant à cette date — 2 pt, 18 min
4. affiche : — 1 pt, 9 min
   - `C'est encore l'été`, si la date est en été ;
   - `C'est l'automne`, si la date est en automne ;
   - `Déjà l'hiver`, si la date du jour est en hiver.

Vous séparerez clairement le calcul de la saison et la partie affichage dans votre programme. Pour cela vous utiliserez une variable entière `saison` dont les valeurs possibles seront : `ETE`, `AUTOMNE`, `HIVER`.

Exemples d'exécutions :

```
Date : 20 10
c'est l'automne

Date : 5 9
C'est encore l'ete

Date :  23 12
Dela l'hiver
```

3 For ou while ?

3.1 Puissances successives d'un entier (*4 points*)

Question G. Écrire un programme qui demande un entier n à l'utilisateur et calcule puis affiche n^{10} (n puissance dix). — 2 pt, 18 min

Exemple d'exécution.

```
Donner un entier : 2
2 exposant 10 = 1024
```

Question H. Comment modifier simplement votre programme pour qu'il affiche le calcul des puissances successives de l'entier n saisi, de n^0 jusqu'à n^{10} ? — 1 pt, 9 min

Exemple d'exécution.

```
Donner un entier : 2
2 exposant 0 = 1
2 exposant 1 = 2
2 exposant 2 = 4
...
2 exposant 10 = 1024
```

Question I. Comment modifier simplement votre programme pour qu'il affiche le calcul des puissances successives de l'entier n uniquement à partir de l'exposant 5 ?

1 pt
9 min

Exemple d'exécution.

```
Donner un entier : 2
2 exposant 5 = 32
2 exposant 6 = 64
...
2 exposant 10 = 1024
```

3.2 Unicité des éléments d'un tableau (*4 points*)

Nous disposons d'un tableau t de N entiers. Utiliser une constante symbolique pour N. Nous souhaitons savoir si chaque entier apparaissant dans le tableau n'y apparaît qu'une seule fois, autrement dit on veut savoir si chaque entier est unique dans le tableau.

Question J. Écrire un programme qui, étant donné un tableau initalisé t, teste si le premier élément du tableau est unique et affiche `Vrai` si c'est le cas, `Faux` sinon.

2 pt
18 min

Question K. Écrire un programme qui étant donné un tableau initialisé t, teste si tous les éléments sont uniques et affiche `Vrai` si c'est le cas, `Faux` sinon.

2 pt
18 min

Question bonus (*2 points*)

Cette question est difficile et elle sera notée de manière très stricte. Il vaut beaucoup mieux traiter toutes les autres questions avant de chercher à y répondre.

Question L. Écrire un programme qui :

1. demande à l'utilisateur d'entrer un nombre n dont on supposera qu'il est strictement supérieur à 1.
2. tant que n est différent de 1, répète l'étape suivante :

calculer une nouvelle valeur de n ainsi : – si n est pair, le diviser par deux – si n est impair, le multiplier par trois et lui ajouter un et afficher la nouvelle valeur de n.

3. affiche le nombre de fois où l'étape précédente a été répétée, et la plus grande valeur prise par n au cours de ce calcul.

Ce programme termine-t-il toujours ?