

---

# Éléments d'informatique : partiel de fin de semestre

---

**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 recommandé de le faire tourner à la main sur un exemple pour s'assurer de sa correction.


## 1 Étude de programmes et questions de cours

### 1.1 Somme des éléments d'un tableau (*3,5 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 }
```


**Question A.** Avant d'aller plus loin 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 B.** Expliquer le fonctionnement de l'instruction `#define`.

 1 pt  
9 min

**Question C.** Compléter le programme pour qu'il calcule effectivement la somme des éléments du tableau (pour une taille de tableau arbitraire).

 1.5 pt  
13 min

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


La somme est 37

## 1.2 Une erreur classique (3 points)

Pippo a écrit un programme C. Celui-ci compile, mais une erreur survient à l'exécution, qu'il ne comprend pas.

```
$ gcc puissance.c -o puissance.exe
$ puissance.exe
Entrer un nombre reel : 2.3
Entrer son exposant (entier positif) : 2
Segmentation fault
```


**Question D.** Expliquer brièvement ce que signifie ce message d'erreur (dernière ligne).

 1 pt  
9 min


Voici quelques lignes choisies du programme.

```
10  int main()
11  {
12      /* Déclaration et initialisation des variables */
13      double x;
14      int n;
15
16      :
17
22      printf("Entrer un nombre reel : ");
23      scanf("%lg", x);
24      printf("Entrer son exposant (entier positif) : ");
25      scanf("%d", n);
```


**Question E.** Que faut-il corriger ?

 1 pt  
9 min

**Question F.** Quelle option de compilation aurait dû utiliser Pippo pour obtenir de l'aide ?

 0.5 pt  
4 min

**Question G.** Compléter le programme pour que, lorsque l'entier  $n$  saisi est positif, il calcule et affiche  $x^n$ .

 1 pt  
9 min

## 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.

**Question H.** Écrire un programme qui, étant donné un tableau initialisé  $t$ , teste si le premier élément du tableau est unique et affiche **Vrai** si c'est le cas, **Faux** sinon.

2 pt  
18 min

**Question I.** 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

*Tout le traitement sera effectué dans le main, sans faire appel à des fonctions utilitateurs.*

### 3 Faut-il aller au cinéma ? (4 points)

Je viens de gagner une place de cinéma pour un séance ce soir. Je dispose des informations suivantes, codées dans des variables entières, pour décider si je vais me rendre à cette séance ou rester chez moi :

- **critique** une critique du film lui attribuant une appréciation parmi MAUVAIS, MOYEN, BON.
- **acteurs** mon appréciation personnelle du choix des acteurs : MOYEN ou BON.
- **distance** la distance approximative (en kilomètres) qui me sépare du cinéma.

Vous utiliserez des constantes symboliques pour coder les appréciations et vous initialiserez les trois variables, **critique**, **acteurs**, **distance** à des valeurs de votre choix. Pour prendre ma décision je dispose de l'arbre de décision suivant.

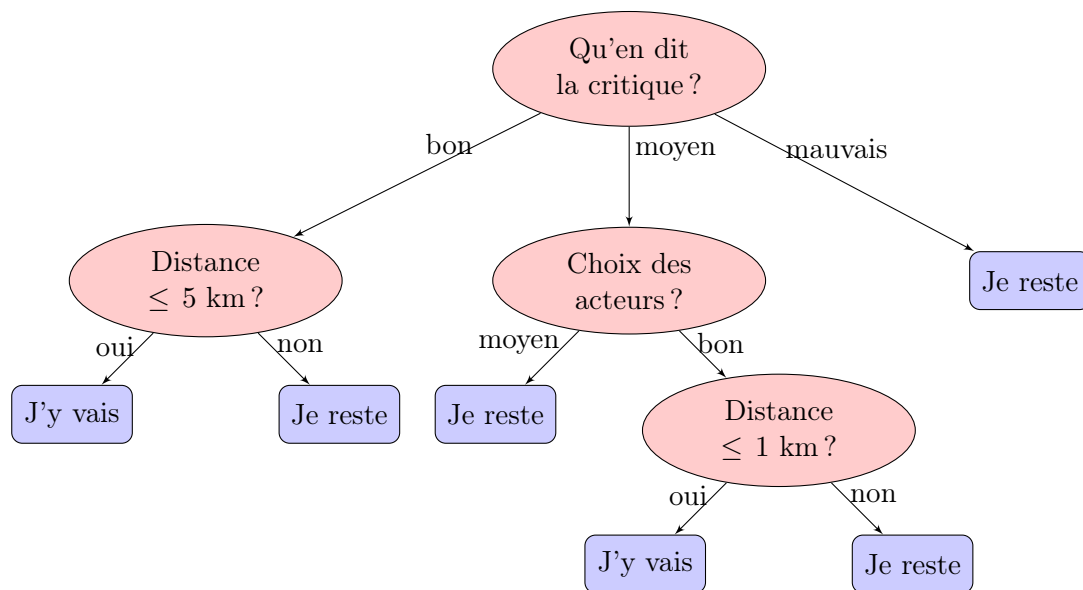


FIGURE 1 – Décider si je vais au cinéma

**Question J.** Écrire un programme implantant cet arbre de décision et affichant la décision à prendre (quel que soit le choix d'initialisation des variables).

4 pt  
36 min

**Barème.** On accepte le fait que la distance soit une variable réelle, sans pénalité.

*Compter :*

- 0,5 points pour le codage **BON**, **MOYEN**, **MAUVAIS** à l'aide de `define` s'il est correct (pas d'erreur de syntaxe, comme un signe égal), retirer une pénalité de 0.25 s'il a trop de constantes symboliques. S'il y a par exemple une constante symbolique pour définir la valeur d'initialisation de la distance.
- 0,5 points pour des déclarations avec initialisations correctes.
- 0,5 points pour l'indentation (si celle-ci est absente la sanctionner).

- Une pénalité de 0,25 points par oubli du `stdlib`, du `stdio` ou du `return` final (on ne sanctionne pas l'utilisation d'un 0 à la place du `EXIT_SUCCESS`).
  - On admet que les cas exclusifs soient traités dans des `if` en cascade (comme dans le corrigé à la racine de l'arbre) plutôt qu'avec des `if else` emboîtés. On admet les `else if` même si ça n'a pas été vu en cours.
  - Pour le reste, dessiner l'arbre sous-jacent au programme de l'étudiant et calculer la distance d'édition entre cet arbre et celui demandé par les opérations suivantes (il ne s'agit pas d'un arbre ordonné) :
    - étiquette à modifier sur un sommet ou un arc
    - inversion père-fils
    - sommet absent
- On part d'un total de 2.5 et on compte 0.5 point de pénalité par opération d'édition nécessaire.

## 4 Histogrammes (5 points)

**Question K.** Soit un entier  $n$  initialisé à une valeur positive de votre choix. Écrire un programme qui affiche une ligne contenant  $n$  fois le caractère '#' et se terminant par un saut de ligne. Pour cette question, vous pouvez répondre en ne donnant que le code de la fonction principale du programme (`main`).

1 pt  
9 min

**Barème.** On part de 1 point et on enlève 0.25 pt par erreur : oubli de déclarer la variable de boucle, absence du retour à la ligne etc. Zéro s'il n'y a pas de `for`.

Soit un tableau d'entiers, initialisé à des valeurs de votre choix, toutes positives ou nulles, et dont la taille  $N$  sera définie par une constante symbolique. Dans les exemples suivants la taille du tableau est 10 et les valeurs contenues dans le tableau sont 3, 0, 2, 8, 9, 10, 3, 5, 5, 2.

On veut écrire un programme qui affiche sous forme d'histogramme (graphique en bâtons, ou graphique barres) les données du tableau : chaque barre de l'histogramme aura une longueur égale à la donnée représentée.

**Question L.** Écrire un programme qui affiche sous forme d'histogramme les données du tableau. Les barres seront dessinées horizontalement, à l'aide du caractère '#'.

3 pt  
27 min

Exemple d'exécution :

Graphique barre 1 :


```
###
##
#####
#####
#####
###
#####
#####
##
```

**Barème.** On compte entre un et trois points si il y a une double boucle imbriquée : on part de 3 on enlève des pénalités jusqu'à 1 (ne pas descendre en dessous). Pénalité faible : il manque un `include`, le  $N$  n'est pas dans un `define` et non utilisé dans la boucle, le tableau n'est pas déclaré, pas initialisé, il manque un saut de ligne, variable de boucle

non déclarée. Cas particulier : on ne compte que 1 point si les deux boucles ont la même variable, sans regarder plus loin.


S'il n'y a pas de double boucle on ne corrige pas plus on met zéro.

**Question M.** Indiquer comment modifier le programme de la question précédente, de manière à demander à l'utilisateur le caractère qui sera utilisé pour dessiner les barres (à la place du #).

 1 pt  
9 min

**Barème.** On ne compte pas de pénalité si le `scanf` est fait dans un format `"%c"` au lieu de `" %c"` ou si c'est de type entier au lieu d'être un `char`, mais un bonus de 0.25pt si utilise un `char`. On compte un demi point pour le `scanf` correct (avec `É`, sinon 0) et un demi point pour le `printf` correct. Pénalité de 0.25 si la variable n'est pas correctement déclarée.

**Question bonus (plus difficile).** Ajouter à votre programme un nouvel affichage en histogramme des données dont les barres progressent cette fois verticalement. Indication : avant cela, votre programme devra trouver le maximum parmi les données du tableau.

 2 pt  
18 min

Exemple d'exécution :

Graphique barre 2 :

```

          ##
        ## ##
      ## ## ##
    ## ## ##
  ## ## ##
 ## ## ##  ## ##
## ## ##  ## ##
##      ## ## ## ## ## ##
##    ## ## ## ## ## ## ## ##
##    ## ## ## ## ## ## ## ##

```

**Barème.** Deux points si c'est vraiment parfait, uniquement.

On donne plus d'un point uniquement lorsque la réponse est très proche d'une solution correcte. On admet une réponse à base de tableau bidimensionnel.

On peut aller jusqu'à un point si des parties du programme ne fonctionnent pas (recherche du maximum par exemple), mais qu'il y a de bonnes idées (par exemple, penser à balayer la ligne courante).

S'il n'y a que la recherche du maximum on ne compte qu'un quart de point.