

DS N° 1 INFORMATIQUE II
FILIÈRES MI-MEF

Calculatrice et documents non autorisés

Le langage utilisé doit être le langage C.

Le barème est donné à titre indicatif et peut être sujet à modification.

Dans ces exercices, l'ordre des questions est important : plusieurs questions nécessitent d'utiliser les fonctions définies dans les questions précédentes. Si toutefois vous ne parvenez pas à écrire une fonction, vous pouvez tout de même l'utiliser.

Un extrait de la table ASCII est disponible à la fin du sujet si besoin.

Exercice 1 (Plaque d'immatriculation) [5.75 pts]

On souhaite créer un code qui va vérifier qu'une chaîne de caractères respecte le format d'une plaque d'immatriculation. Une plaque d'immatriculation en France contient 9 caractères donc le format est le suivant :

2 lettres, 1 tiret, 3 chiffres, 1 tiret, et enfin 2 lettres

Quelques lettres ne sont pas utilisées : il s'agit des lettres I, O et U, pour éviter les confusions avec les chiffres 1 et 0 et la lettre V, respectivement.

Une seule association de deux lettres est interdite, il s'agit de SS, pour se conformer au Code pénal qui *"réprime le port ou l'exhibition d'insignes/emblèmes rappelant ceux d'organisations ou de personnes responsables de crimes contre l'humanité"*.

Une autre association de deux lettres est spécifique, WW : si cette association est placée en début de plaque, cela permet d'identifier une immatriculation provisoire du véhicule le temps que les démarches d'immatriculation soient finalisées.

IMMATRICULATIONS CORRECTES

AA-001-AA
ZZ-999-ZZ
GT-498-ED
GW-245-RY
WA-124-TL
WW-123-AB (immatriculation temporaire)
AB-123-WW (immatriculation classique)

IMMATRICULATIONS INCORRECTES

AB-444-SS (à cause de SS)
SS-987-AA
UA-375-EF (à cause du U)
AI-164-GH (à cause du I)
AF-684-OP (à cause du O)

1. [1.75 pts] Écrire une fonction/procédure longueur(...) qui va compter le nombre de caractères utiles d'une chaîne.

Les consignes suivantes doivent être respectées :

- prendre une chaîne de caractères en paramètre
- retourner une valeur entière indiquant le nombre de caractères utiles qui constituent la chaîne
- si un problème survient pendant la fonction, retourner la valeur -1
- interdiction d'utiliser les fonctions du module `string.h`
- dans cette question, le parcours de tableau doit se faire à l'aide de pointeur (pas de crochets [] autorisés !)

2. [4.00 pts] Écrire une fonction/procédure format(...) qui va vérifier la validité d'une plaque d'immatriculation.

Les consignes suivantes doivent être respectées :

- prendre une chaîne de caractères en paramètre
- retourner une valeur entière indiquant si le format est correct ou non (0 pour un mauvais format ou une erreur, 1 pour un format correct et 2 pour un format correct d'une immatriculation temporaire)
- Un format correct ne concerne que les lettres majuscules : si des lettres minuscules sont détectées, le format sera considéré comme mauvais
- dans cette question, le parcours de tableau doit se faire à l'aide de pointeur (pas de crochets [] autorisés !)

Exercice 2 (Normalisation d'un vecteur) [3.25 pts]

A partir d'un vecteur $\vec{V}$ d'un plan 2D orthonormé, on peut extraire chacune de ses composantes V_x et V_y . La normalisation d'un vecteur consiste à modifier ses composantes de manière à ce que la norme du vecteur soit égale à 1, mais sans changer la direction du vecteur. On notera alors ses composantes V_{nx} et V_{ny} (n pour *normalisé*). Cela revient à diviser individuellement chaque composante du vecteur par la norme du vecteur $\|\vec{V}\|$.

Le calcul de normalisation d'un vecteur $\vec{V}$ est le suivant :

$$\text{pour } \vec{V} = (V_x, V_y) \quad \left\{ \begin{array}{l} V_{nx} = \frac{V_x}{\|\vec{V}\|} \\ V_{ny} = \frac{V_y}{\|\vec{V}\|} \end{array} \right. \quad \text{et avec la norme } \|\vec{V}\| = \sqrt{(V_x)^2 + (V_y)^2}$$

1. [2.0 pts] Écrire une procédure `norm(...)` qui va normaliser les 2 composantes d'un vecteur.

Les consignes suivantes doivent être respectées :

- les composantes sont des valeurs réelles.
- la fonction appelante doit pouvoir récupérer les composantes normalisées du vecteur
- si la procédure rencontre un problème le programme doit s'arrêter
- les fonctions de la bibliothèque `math.h` peuvent être utilisées

2. [1.25 pts] Écrire un programme principal qui va :

- déclarer 2 variables, 1 pour chaque composante d'un vecteur
- stocker une valeur aléatoire réelle entre les bornes 1.00 et 10.00 incluses dans chacune des composantes (avec 2 décimales significatives après la virgule)
- appeler la procédure `norm` pour normaliser le vecteur
- afficher les valeurs des composantes après la normalisation

Exercice 3 (Hérisson bleu et Renard à 2 queues) [11.0 pts]

On souhaite coder un petit jeu vidéo en 2D qui met en scène de petits animaux mignons et vifs en couleur, qui courent extrêmement vite pour ramasser des anneaux dorés. Ces personnages possèdent des caractéristiques fixes : leur nom, leur vitesse de déplacement de base, ainsi que la hauteur maximale de leur saut.

Au delà de ces caractéristiques fixes, les personnages possèdent également des données variables comme un nombre de points de vie et une quantité d'anneaux qu'il amasse tout au long des niveaux.

1. [0.75 pt] Créer une structure `Info` qui va contenir les caractéristiques fixes d'un personnage. Les 2 valeurs numériques sont des valeurs entières, le nom est une chaîne de caractères avec 25 caractères utiles au maximum.

2. [0.75 pt] Créer une structure `Personnage` qui va contenir l'ensemble des données d'un personnage joueur :

- une valeur entière contenant le nombre de vies que le personnage possède
- une valeur entière avec le nombre d'anneaux dorés ramassés
- une structure `Info` contenant les caractéristiques fixes du personnage

3. [2.25 pts] Créer une fonction constructeur pour la structure `Personnage` qui ne prendra aucun paramètre, et qui va retourner une structure `Personnage`. Les champs doivent être remplis de la manière suivante :

- le nom doit être entré au clavier par l'utilisateur
- le nombre de vies est fixé à 3
- le nombre d'anneaux est fixé à 0 en début de partie
- la vitesse est une valeur entière aléatoire entre 700 et 1300 (valeurs incluses)
- le saut est une valeur entière aléatoire entre 300 et 400 (valeurs incluses)

4. [1.25 pts] Écrire une fonction/procédure `vitesse(...)` qui va prendre en paramètre une structure `Personnage` et va retourner la vitesse réelle du personnage. La vitesse réelle est égale à la caractéristique de vitesse de base augmentée de 1% pour chaque anneau en possession du personnage.

Si le nombre d'anneaux ramassés par le personnage est supérieur à 100, la vitesse totale sera tout de même plafonnée à 200% de la caractéristique de base.

5. [2.5 pts] Écrire une fonction/procédure `degats(...)` qui va simuler le fait qu'un **Personnage** passé en paramètre s'est fait touché par un projectile ennemi. Ce morceau de code va modifier les caractéristiques de ce **Personnage** et le retourner à la fonction appelante.

Les conséquences de se faire toucher par un projectile dans le jeu dépendent du nombre d'anneaux en possession du **Personnage** :

- si le nombre d'anneaux en possession du **Personnage** est supérieur ou égal à 100, le **Personnage** perd simplement la moitié de ses anneaux et il continue de jouer.
- si le nombre d'anneaux est positif et inférieur à 100, le **Personnage** perd tous ses anneaux.
- enfin si le nombre d'anneaux est égal à 0 lors de l'appel à ce code, le **Personnage** perd 1 point de vie.
- ce morceau de code ne fera aucun traitement si le nombre de points de vie du **Personnage** est égal à 0.

6. [3.5 pts] Écrire une procédure `course(...)` qui va permettre de comparer les vitesses de déplacement de 2 **Personnage** distincts et déterminer lequel est plus rapide pour terminer un niveau.

Les consignes à respecter sont les suivantes :

- prendre en paramètre deux **Personnage**
- et prendre en paramètre une hauteur, qui sera la hauteur maximale de saut que les **Personnage** devront effectuer lors de cette course
- déterminer si les **Personnage** peuvent franchir la hauteur maximale imposée, et si ils le peuvent, déterminer celui qui finira la course le plus rapidement (celui qui a la plus grande vitesse).
- afficher le nom du gagnant, ou une égalité le cas échéant

TABLE ASCII et PROTOTYPES de fonctions utiles au dos de cette feuille!

TABLE ASCII

Decimal Char	Decimal Char	Decimal Char
32 [SPACE]	64 @	96 `
33 !	65 A	97 a
34 "	66 B	98 b
35 #	67 C	99 c
36 \$	68 D	100 d
37 %	69 E	101 e
38 &	70 F	102 f
39 '	71 G	103 g
40 (	72 H	104 h
41)	73 I	105 i
42 *	74 J	106 j
43 +	75 K	107 k
44 ,	76 L	108 l
45 -	77 M	109 m
46 .	78 N	110 n
47 /	79 O	111 o
48 0	80 P	112 p
49 1	81 Q	113 q
50 2	82 R	114 r
51 3	83 S	115 s
52 4	84 T	116 t
53 5	85 U	117 u
54 6	86 V	118 v
55 7	87 W	119 w
56 8	88 X	120 x
57 9	89 Y	121 y
58 :	90 Z	122 z
59 ;	91 [	123 {
60 <	92 \	124
61 =	93]	125 }
62 >	94 ^	126 ~
63 ?	95 _	127 [DEL]

PROTOTYPES DE FONCTIONS

SRAND

```
#include <stdlib.h>
void srand(unsigned int seed);
```

RAND

```
#include <stdlib.h>
int rand(void);
```

TIME

```
#include <time.h>
time_t time(time_t* arg);
```