Printemps 2018 **Examen médian- LO01**

2 H - Sans documents (sauf diagrammes de C)

La clarté de vos réponses sera prise en compte. Ne pas écrire au crayon. N'oubliez pas de commenter vos programmes.

Vous aurez besoin de 3 copies : une par exercice

1 – Nombres parfaits (7 points)

Un nombre entier est qualifié de *parfait* si il est égal à la somme de ses diviseurs propres (un diviseur propre est un diviseur différent de soi-même).

Exemples: 6 est un nombre parfait, ses diviseurs propres sont: {1 , 2 , 3 } et on a 1+2+3=6
28 est un nombre parfait, ses diviseurs propres sont {1 , 2 , 4 , 7 , 14 } et on a 1+2+4+7+14=28

Ecrire un programme qui :

- Demande à un utilisateur un nombre nb (on s'assurera qu'il est inférieur à 10000),
- Calcule tous les nombres parfaits inférieurs à nb et les stocke dans un tableau,
- Réécrit dans ce même tableau les nombres du plus grand au plus petit,
- Affiche le contenu de ce tableau et ainsi que la somme de tous ses éléments.

2 – Exercices divers (7 points)

2.1) Ecrire un programme qui lit une phrase terminée par un point et compte le nombre de mots ainsi que le nombre de signes de ponctuation. Le programme affichera le type de la phrase selon les valeurs suivantes :

- Phrase longue si le nombre de mots est supérieur ou égal à 15
- Phrase moyenne si le nombre de mots est compris entre 8 et 14
- Phrase courte si le nombre de mots est inférieur ou égal à 7
- Si la phrase est longue ou moyenne on dira aussi qu'elle est complexe s'il y a plus de 3 symboles de ponctuation dans la phrase (point final compris)

On supposera que 2 mots sont séparés par un espace au moins. Les signes de ponctuation sont les virgules (,), point virgules (;) et le point (.).

Remarque : La phrase sera lue caractère par caractère comme vu en TD

2.2) Ecrire un programme qui demande à l'utilisateur un nombre n compris entre 1 et 9 et affiche le motif pyramide de n lignes comme le montrent les exemples suivants :

Pour le nombre 5

```
    1
   212
  32123
 4321234
543212345
```

pour le nombre 3

```
  1
 212
32123
```

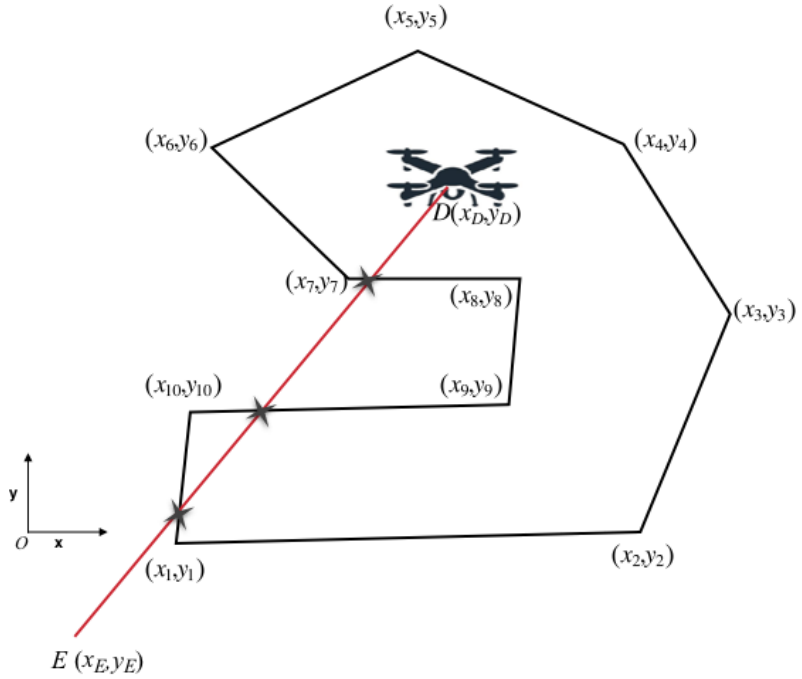
2.3) que fait le programme ?

```c
#include <stdio.h>

int main()
{ int u1,u2,u3;
  int *p;

  u1=25;
  p=&u1;
  u2=u1/3;
  u3=*p%10;
  u1++;
  printf("u1=%d, u2=%d, u3=%d, p=%d ",u1,u2,u3,*p) ;
  return 0 ;
}
```

3 – Drone (7 points)

Afin de localiser un drone lors d'un vol, on cherche à vérifier si sa position projetée au sol (x_D, y_D) est à l'intérieur d'un polygone. Une méthode consiste à calculer le nombre d'intersections du segment [ED] avec les arêtes du polygone. Le point E (x_E, y_E), à choisir judicieusement, doit être à l'extérieur du polygone. Si le nombre d'intersections est impair alors le drone est à l'intérieur du polygone sinon il est à l'extérieur.

Un polygone est d'une suite finie de points (x_i, y_i) du plan appelés sommets (on considérera au maximum 30 sommets)



Ecrire un programme qui :

3.1) demande à l'utilisateur de saisir les coordonnées réelles des sommets d'un polygone et qui les stocke dans un tableau de la manière suivante :

x_1	x_2	x_3	...	x_n			
y_1	y_2	y_3	...	y_n			

Le programme demandera aussi à l'utilisateur de saisir la position D du drone.

3.2) détermine le point E (x_E, y_E) de telle manière qu'il soit à l'extérieur du polygone. On choisira E tel que x_E soit plus petit que tous les x_i et y_E soit plus petit que tous les y_i

3.3) vérifie si le drone est à l'intérieur ou à l'extérieur du polygone en comptant le nombre d'intersections du segment [ED] avec toutes les arêtes du polygone (y compris entre le dernier et le premier sommet). Pour simplifier l'écriture du programme, vous pouvez dupliquer le premier sommet après le dernier sommet.
Le programme affichera l'un des 2 messages suivants : « le drone est à l'intérieur du polygone » ou « le drone est à l'extérieur du polygone »

Rappel :

Deux segments [AB] et [CD] sont sécants si : $\det(AB, AC) * \det(AB, AD) < 0$ et $\det(CD, CA) * \det(CD, CB) < 0$

Avec $\det(AB,CD) = x_{AB} * y_{CD} - y_{AB} * x_{CD}$