

A) LA DIVISION ...

En des temps immémoriaux (1965), la calculatrice mécanique effectuait les divisions par soustractions répétées et par décalages. Par exemple, pour diviser 876 543 219 par 2 345 678, on alignait les nombres à partir des chiffres de gauche (cf Fig. 1), et le diviseur était soustrait du dividende le plus grand nombre de fois sans obtenir un reste négatif. Le nombre de soustractions ainsi obtenu devient le premier chiffre du quotient. Le diviseur est ensuite décalé d'une position vers la droite (cf Fig. 2) et à nouveau soustrait successivement du reste pour obtenir le prochain chiffre du quotient. Ce procédé est répété jusqu'à ce que le reste soit plus petit que le diviseur.

Figure 1:

```
876543219
- 2345678   première soustraction valable
=====
641975419
- 2345678   seconde soustraction valable
=====
407407619
- 2345678   troisième soustraction valable
=====
172839819  reste
- 2345678   soustraction rejetée
=====
négatif
```

Figure 2:

```
172839819
- 2345678
=====
149383039
etc.
```

Écrivez un programme qui effectue « LA DIVISION » entre des nombres lus dans un fichier d'entrée (GDIV.IN) et restitue les résultats dans un fichier de sortie (GDIV.OUT).

La structure du fichier d'entrée (GDIV.IN) est la suivante:

La première ligne du fichier d'entrée est un entier positif n , $n < 20$, indiquant le nombre de divisions à effectuer. Chaque paire de lignes suivantes représente respectivement le dividende et le diviseur d'une division. Les nombres fournis seront positifs et d'une longueur maximum de 40 chiffres.

La structure du fichier de sortie (GDIV.OUT) est la suivante:

Pour chaque dividende et diviseur fournis, le fichier de sortie contiendra une paire de lignes indiquant respectivement le quotient et le reste.

EXEMPLE :

GDIV.IN :

```
3
987654321
3456789
```

33
11
11
33

GDIV.OUT :

285
2469456
3
0
0
11

B) LES PARTITIONS ...

Soit un entier positif donné k , une *partition* est une séquence de nombres entiers positifs donnés en ordre décroissant dont la somme est k .
par exemple, $(2,2,2,2,2,2)$ et $(5,3,2,1,1)$ sont des partitions de 12.

Soient deux partitions distinctes (a_1, a_2, \dots, a_n) et (b_1, b_2, \dots, b_m) d'un nombre donnée k , nous dirons que $(a_1, a_2, \dots, a_n) > (b_1, b_2, \dots, b_m)$ si pour le plus petit entier positif t tel que $t \leq n$, $t \leq m$ et $n \leq m$, nous avons $a_t > b_t$.

Cet ordonnancement nous permet d'obtenir pour toutes les partitions d'un entier k donné un ordre « lexicographique », dans lequel chaque partition est supérieure à toutes celles qui la précèdent.

Par exemple, si $k=5$, les partitions en ordre « lexicographique » croissant sont :

(1,1,1,1,1)
(2,1,1,1)
(2,2,1)
(3,1,1)
(3,2)
(4,1)
(5)

Écrivez un programme qui pour une série de couples d'entiers positifs k et i lus dans un fichier d'entrée (PART.IN) détermine la $i^{\text{ème}}$ partition de k dans l'ordre « lexicographique » croissant et l'écrit dans un fichier de sortie (PART.OUT).

La structure du fichier d'entrée (PART.IN) est la suivante:

La première ligne du fichier d'entrée est un entier positif n , indiquant le nombre de couples d'entiers positifs k et i qui seront à traiter. Les n lignes suivantes contiendront chacune un des couples de données.

La structure du fichier de sortie (PART.OUT) est la suivante:

Pour chaque couple d'entiers k et i fournis, le fichier de sortie contiendra une ligne donnant la $i^{\text{ème}}$ partition de k dans l'ordre « lexicographique » croissant ou « trop grand » si i est supérieur au nombre de partitions de k .

EXEMPLE :

PART.IN :

3
1 1
5 4
5 8

PART.OUT :

(1)
(3,1,1)
Trop grand

REMARQUES :

Les Programmes sont à écrire en PASCAL, en C ou en CAML. Pour l'accès aux fichiers en entrée et en sortie, un rappel syntaxique des instructions sur les fichiers « texte » est fourni en annexe.

Le principe de l'algorithme retenu doit être exprimé avant chaque programme.

Question 1 : Un Kilo octet correspond à :

- A 1000 octets
- B 1000 bits
- C 1024 bits
- D 1024 octets

Question 2 : Parmi les noms suivants , lequel ne correspond pas à un bus d'ordinateur ?

- A PCI
- B AMD
- C ISA
- D AGP

Question 3 : Qu'est ce qu'un algorithme ?

- A un langage de programmation primitif
- B une convention de notation mathématique
- C une ébauche de programme
- D une méthode de résolution d'un problème suivant un enchaînement de règles opératoires

Question 4 : Que calcule la fonction définie par « $f(x) \rightarrow$ si $x=0$ alors 1 sinon $x * f(x-1)$ » ?

- A le carré de x
- B la somme des nombres de 0 à x
- C le sinus de x
- D la factorielle de x

Question 5: Qu'est ce qu'un filtre gardé ?

- A un filtre sur aucun argument
- B un filtre sur plusieurs arguments
- C un filtre utilisant le symbole « _ »

D un filtre associé d'une clause « when »

Question 6 : Qu'est ce que une fonction récursive ?

A une fonction qui ne renvoie aucun résultat

B une fonction système

C une fonction qui n'accepte pas d'arguments

D une fonction qui s'appelle elle-même

Question 7: Que signifie l'acronyme ADSL ?

A Asymmetric Digital Subscriber Line

B Asynchronous Digital Subnet Line

C Asynchronous Digital System Line

D Address Destination System Line

Question 8: Les outils qui vous permettent de retrouver des informations sur Internet sont appelés :

A langages de programmation

B index

C moteurs de recherche

D annuaires

Question 9 : Quel est le serveur qui permet de convertir un nom de domaine en une adresse internet ?

A FTP

B DNS

C TFTP

D DHCP

Question-10 : La RJ45 correspond à :

A- une application industrielle

B- une interface de communication une interface de programmation

C- un connecteur

Question-11 : La représentation suivante 200.12.2.34 correspond à :

A- une adresse physique

B- un numéro de port

C- une adresse IP

D- une adresse d'une machine

Question-12 : Dans un ordinateur les données sont représentées sous forme de :

A- signal lumineux

B- signal électrique analogique

C- signal électrique numérique

D- caractères alphabétiques

Question-13 : Qu'imprime cet algorithme ?

```

I = 1
Etiquette-1
  Si I < 10 alors I = I * 1
    Imprimer I
    I = I + 1
    Aller à étiquette-1
  Sinon fin.

```

- A- le logarithme de I
- B- la puissance 10 de I
- C- les 9 premiers chiffres
- D- la somme des 10 premiers chiffres

Question-14 : A = 0

Faire tant que I < 10

```

(
  A = A + 1
  Imprimer A
)

```

est-ce:

- A- une boucle
- B- une alternative
- C- ni une boucle ni une alternative
- D- une suite de commentaires

Question-15 : En langage caml l'expression suivante

```

Let abs x = if x then x else -x;;

```

- A- met x à 1
- B- affecte la valeur 0 à x
- C- calcule la valeur absolue de x
- D- décrémente X de 1

Question-16 : Quelles sont les valeurs que peut prendre une variable booléenne ?

- A- faux
- B- vrai
- C- supérieur
- D- égale

Question-17 :: Sur combien d'octets au minimum peut-on coder la valeur 500 :

- A- un
- B- deux
- C- trois
- D- quatre

Question-18 : Le top level de caml est :

- A- le débuser
- B- l'interpréteur de commande

- C- le compilateur
- D- le gestionnaire d'impression

Question-19 : Parmi les mémoires suivantes lesquelles sont volatiles ?

- A- RAM
- B- ROM
- C- EPROM
- D- Cache processeur

Question-20 : Le réseau WiFi est un :

- A- réseau industriel
- B- réseau local sans fil
- C- réseau grande distance
- D- réseau de téléphonie

Question-21: Le réseau Wimax peut couvrir une distance de :

- A- 200 m
- B- 100 km
- C- 50 km
- D- 100 m

Question 22: Que signifie le sigle CNIL ?

- A Cour Nationale de l'Informatique et des Libertés
- B Commission Nationale de l'Informatique et des Libertés
- C Chambre Nationale des Informations Litigieuses
- D Chambre Nationale des Informaticiens Libres

Question 23 : Dans un courrier électronique, une pièce jointe est :

- A un lien hypertexte vers un site
- B un fichier que j'envoie ou que je reçois avec le courrier
- C un petit fichier de cryptage de l'information
- D une pièce justificative

Question 24 : L'extension d'un fichier est :

- A une nouvelle copie du fichier
- B la partie du nom du fichier indiquant son type
- C la décompression du fichier
- D la modification du nom de fichier

Question 25 : L'arborescence décrit :

- A l'organisation des fichiers et répertoires
- B la taille des fichiers
- C la liste des mots de passe
- D le type de processeur

ANNEXE

LANGAGE PASCAL :

- ◇ **Procedure Assign(var F ; Nom_fic : string) ;**
Affecte le nom d'un fichier disque à une variable-fichier.
- ◇ **Procedure Reset(var f [:FILE ; TailleEnr : Word]) ;**
Ouvre un fichier disque existant.
- ◇ **Procedure Rewrite(var f [:FILE ; TailleEnr : Word]) ;**
Crée et ouvre un fichier.
- ◇ **Procedure Read(var f : TEXT ; v1 [, v2, .. , vN]) ;**
Lit une ou plusieurs valeurs dans une ou plusieurs variables.
- ◇ **Procedure Readln(var f : TEXT ; v1 [, v2, .. , vN]) ;**
Lit une ou plusieurs valeurs dans une ou plusieurs variables jusqu'à la fin de ligne.
- ◇ **Procedure Write(var f : TEXT ; v1 [, v2, .. , vN]) ;**
Ecrit une ou plusieurs valeurs dans le fichier.
- ◇ **Procedure Writeln(var f : TEXT ; v1 [, v2, .. , vN]) ;**
Ecrit une ou plusieurs valeurs dans le fichier plus un retour ligne.
- ◇ **Procedure Close(var f) ;**
Ferme un fichier ouvert.

LANGAGE C :

- ◇ **FILE *fopen(char *path, char *mode) ;**
Ouvre le fichier dont le nom est contenu dans la chaîne pointée par « path » et lui associe un flux. L'argument « mode » pointe vers une chaîne commençant par l'une des séquences suivantes :
 - r fichier existant ouvert en lecture
 - r+ fichier existant ouvert en lecture/écriture
 - w crée un fichier et l'ouvre en écriture
 - w+ crée un fichier et l'ouvre en lecture/écriture
- ◇ **int fscanf(FILE *stream, const char *format, ...) ;**
Lit les données depuis un flux pointé par « stream », convertit ces données selon le « format » décrit et stocke le résultat des conversions dans des arguments pointeurs.
- ◇ **int fprintf(FILE *stream, const char *format, ...) ;**
Ecrit les données sur le flux « stream » indiqué, ces données étant préalablement converties selon le « format » décrit.
- ◇ **int fclose(FILE *stream) ;**
Dissocie le flux nommé « stream » du fichier sous-jacent.

LANGAGE CAML :

◇ **val openfile : string -> open_flag list -> file_perm -> file_descr**

Ouvre un fichier dont le premier argument est le nom, le second la liste des modes d'ouverture, le troisième les permissions, et renvoie un descripteur de fichier.

Le type "open_flag" est défini ainsi :

```
type open_flag =  
  | O_RDONLY  
  (* Ouverture pour lecture seulement *)  
  | O_WRONLY  
  (* Ouverture pour écriture seulement *)  
  | O_RDWR  
  (* Ouverture pour lecture et écriture *)
```

Remarque : il existe d'autres modes, mais ils n'ont pas d'intérêt dans le cas présent.

◇ **val close : file_descr -> unit**

Ferme un fichier à partir d'un descripteur.

◇ **val read : file_descr -> string -> int -> int -> int**

"read fd buff ofs len" lit len caractères du fichier dont le descripteur est fd, stocke ces caractères dans buff à partir de la position ofs et renvoie le nombre d'octets lus.

◇ **val write : file_descr -> string -> int -> int -> int**

"write fd buff ofs len" extrait len caractères contenus dans buff à partir de la position ofs et les écrit dans le fichier dont le descripteur est fd.