**CPI - 1ère Année - Année Universitaire 2013/2014 - (Semestre 1)**

## EMD1 d'ALGORITHMIQUE

Date : Mardi 3 décembre 2013
Durée : 3 Heures
**DOCUMENTS INTERDITS**

Depuis l'antiquité et Euclide les nombres premiers n'ont cessé d'étonner les mathématiciens. Savez – vous :

- Qu'il existe 2 nombres premiers qui sont les diviseurs de **TOUS** les nombres de la forme abcdabcd (autrement dit qui sont composés de 2 parties identiques de 4 chiffres chacune, tel que 47384738 par exemple) ? à vous de trouver **un seul** de ces 2 nombres. ***(Partie 1)***

- Qu'il y a aussi 11 nombres premiers, tous inférieurs à 1 000 000, qui, lorsque vous les divisez par tous les nombres compris entre 2 est 11, vous trouverez toujours un reste égal à 1. Comment retrouver ces 11 nombres premiers ***(Partie 2 )***
  Exemples : les nombres premiers 55441 et 970201.

- Que le nombre premier 37, est très mystérieux. En effet, si vous prenez n'importe quel multiple de 37et que vous le multipliez à son tour par 28, ensuite si vous additionnez le nombre représenté par les 3 dernières positions de ce produit à ceux du nombre représenté par les positions restantes, vous obtenez toujours un multiple de 37.
  Vérifiez cela pour les 15 premiers multiples de 37 ***(Partie 3)***
  Exemples : pour les quatre premiers multiples :

1*37*28= 1036 et 1 + 36 = 37 est un multiple de 37
2*37*28= 2072 et 2 + 72 = 74 est un multiple de 37
3*37*28= 3108 et 3 + 108 = 111 est un multiple de 37
4*37*28= 4144 et 4 + 144 = 148 est un multiple de 37
……

**Travail à Faire :**

- Les trois parties sont indépendantes, donc votre copie doit comporter distinctement trois parties. Pour chaque partie vous devez faire son découpage et sa justification, les analyses et les algorithmes de tous les modules nouveaux (non vus en TDs). Si votre solution utilise des modules déjà faits en TDs vous devez impérativement préciser, pour chaque module : son en tête et son rôle. (Attention tout module dont l'interface ou le rôle est incorrect sera rejeté)

Exple :
Fonction ARRANGE (N, p : entier) : entier
*// Donne le nombre d'arrangement de N objets pris p à p*

- Cependant vous ne présenterez qu'un seul algorithme principal ( et son analyse bien sur !)
- Pour la partie programmation vous devez programmer un module de votre choix et l'algorithme principal

**Barème ( sur 21 points)** :
Partie 1 (4 pts) , Partie 2 (4 pts) , Partie 3 (4 pts) , algorithme principal (6 pts) , programmation (3 pts)

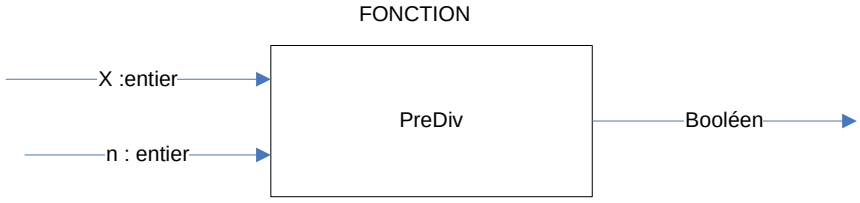
**ATTENTION**: Votre solution doit **ABSOLUMENT** tenir compte du **formalisme étudié** en cours et tout travail non soigné ne pourra en aucun cas prétendre à la note max. prévue dans le barème !

**Alors soignez votre travail et bon courage !**

**PARTIE 1 :**

**Découpage**

- On a besoin d'un module qui nous permet de savoir si un nombre (X) divise TOUS les nombres de la forme AA ou A représente un nombre composé de n chiffres ( abab si n=2, abcabc si n=3 ,abcdabcd si n =4 etc... – (Module **PreDiv)**




**Rôle** : donne VRAI si X divise TOUS les nombres de la forme AA ou A représente un nombre composé de n chiffres ( abab si n=2, abcabc si n=3 ,abcdabcd si n =4 etc...

**Construction de PreDiv :**

a) Analyse

On a besoin des fonctions concat.fon, puiss.fon (donnés en cours)

- Recherche = Vrai *{Recherche va permettre de savoir si on a trouvé un diviseur de tous les nombres ou pas }*
- J est la valeur initiale du nombre et Fin la valeur finale. *{Si ce nombre est composé de 2 chiffres, j = 10 et fin = 99 , s'il est composé de 3 chiffres j= 100 et Fin = 999},* donc si n est le nombre de chiffres, alors j = $10^{n-1}$ et Fin = $10^n - 1$
- On Répète ce qui suit :
  - On construit le nombre K en concaténant j avec j *{pour avoir un nombre de la forme JJ}*
  - Si X ne divise pas K alors recherche = Faux *{ ce n'est plus la peine de continuer}*
  - On passe au J suivant en l'incrémentant de 1

Et on s'arrête quand (recherche = Faux) OU ( j = fin) *{ x n'est pas un diviseur ou alors j atteint la valeur max}*

- PreDiv = recherche *{soit il divise Tous les K alors il est égal à Vrai, soit il ne divise pas un seul et on s'est arrêté, alors il est égal à Faux}*

b) Algorithme

```
Fonction PreDiv (x,n : entier) : booléen
Variables        j,k,fin :   Entier
                 recherche :booléen
 fonctions concat.fon, puiss.fon
DEBUT
Recherche ← Vrai
J ← puiss(10,n-1)
Fin ← puiss(10,n)-1
Répéter
   |K ← concat(j,j)
   |Si  k mod x <> 0 alors  recherche ← Faux
   |j ← j+1
Jusqu'à (recherche = Faux)  OU ( j = fin)
PreDiv ← recherche
FIN
```

c) programme

```
Function PreDiv(x,n:longint):boolean;
// donne VRAI si X divise TOUS les nombres de la forme abab si n=2
// abcabc si n=3 ,abcdabcd si n =4 etc...
var j,k,fin :   longint;
    recherche :boolean;
{$i E:\algo\modules\concat.fon}
{$i E:\algo\modules\puiss.fon}
BEGIN
recherche:=true;
j:=puiss(10,n-1);
fin:=puiss(10,n)-1;
repeat
```

```
    k:=concat(j,j);
    if k mod x <> 0 then recherche := false;
    j:=j+1;
until (recherche = false) or (j=fin);
PreDiv:=recherche;
END;
```

**PARTIE 2**

**Découpage**

- on a besoin d'un module qui vérifie si en divisant N par tous les nombres compris entre B1 et B2 , on obtient Toujours un reste égal à la même valeur (module **MemeRest**)


**Rôle** : donne Vrai si en divisant N par tous les nombres compris entre B1 et B2 , on obtient Toujours un reste égal à Reste

**Construction de MemeReste**

a) Analyse

- Trouve = Vrai *{trouve est un aiguillage qui permet de savoir si on a le même reste ou pas. Il est initialisé à Vrai au départ}*
- J est initialisé avec B1
- On répète ce qui suit :
  - Si le reste de la division de N par j (N mod j) est différent de Reste alors trouve = Faux *{et ce n'est plus la peine de continuer}*
  - On incrémente j de 1 *{on passe au j suivant}*

Et on s'arrête quand ( j > B2) OU (trouve = Faux) *{j est > à la borne max B2 ou alors on a trouvé un reste différent de Reste}*

- MemeReste = trouve *{Trouve = vrai si Tous les restes trouvés sont égaux à Reste sinon il est égal à Faux}*

b) Algorithme

```
fonction MemeReste(N,b1,b2,reste: entier ):Booléen
variables  i,j: entier
          trouve: booléen

DEBUT
Trouve ← Vrai
J ← B1
Répéter
   Si N mod j <> Reste Alors trouve := Faux
   j ← j+1
Jusqu'à (j > B2) OU (trouve= Faux)
MemeReste ← trouve
FIN
```

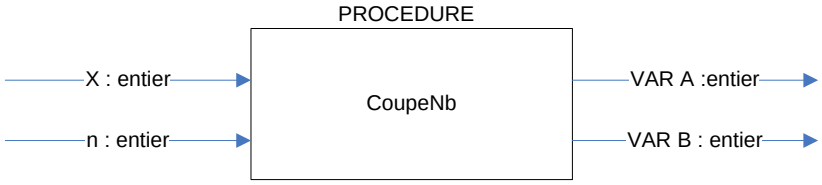
c) Programme

```
function MemeRest(N,b1,b2,reste:longint):boolean ;
// donne Vrai si en divisant N par tous les nombres compris
// entre B1 et B2 , on obtient Toujours un reste egal a Reste
var i,j:longint;
        trouve:boolean;
BEGIN
trouve:=true;
j:=B1;
repeat
    if N mod j <> Reste then trouve :=false;
    j:=j+1;
until (j>B2) or (trouve=false);
MemeRest:=trouve;
```

```
END;
```

**PARTIE 3**
**Découpage**

- On a besoin d'un module qui coupe X en 2 nombres: A compose des n derniers chiffres et B compose par les chiffres restants module (module **coupeNb** )




**Rôle** : coupe X en 2 nombres: A composé des n derniers chiffres et B composé par les chiffres restants

**Construction du module CoupeNb**

a) Analyse

On utilisera les modules ExtPos et Nb_Pos (faits en principe en TDs)

- *Pour enlever les N derniers chiffres d'un nombre :*

*Exple : Si on veut enlever les 4 dernières positions de x = 178954123 et les mettre dans A (ie. 4123). Comme x contient 9 positions, on extrait donc 4 chiffres à partir de la 6ème position soit 9 – 4 +1 ( ie. nb_pos(x) – n +1).*
Donc A = extpos(x,n,nb_pos(x)-n+1)

- *Pour enlever les chiffres restants, on extrait de X ,5 chiffres (c'est-à-dire 9-4), à partir de la 1ère position.*

Donc B = extpos(x,nb_pos(x)-n,1);

b) Algorithme

```
Procedure coupeNb (X,n: entier;var A,B: entier)
Fonctions Nb_pos.fon, ExtPos.fon
DEBUT
A ← extpos(x,n,nb_pos(x)-n+1)
B ← extpos(x,nb_pos(x)-n,1)
FIN
```

c) Programme

```
Procedure coupeNb(X,n:longint;var A,B:longint);
// coupe X en 2 nombres: A compose des n derniers chiffres
//                et B compose par les chiffres restants
{$i E:\algo\modules\Nb_pos.fon}
{$i E:\algo\modules\ExtPos.fon}
BEGIN
A:= extpos(x,n,nb_pos(x)-n+1);
B:=extpos(x,nb_pos(x)-n,1);
END;
```

**NOTA : vous avez constaté que lors de notre découpage nous n'avons pas tenu compte du fait que les nombres sont premiers ou pas car nous avons tenté de généraliser au maximum la solution dans le cas d'une réutilisation possible de nos modules.**

**Construction de l'Algorithme principal**

a) Analyse

On aura besoin des fonctions prem, Nb_pos, PreDiv, MemeReste (modules Prem et nb_Pos faits en TDS)
Et de la procédure coupeNb

Partie 1 ----------

On fait varier I = 2,3,4,5,6,….

Et on s'arrête dès que l'on trouve un i qui est en même temps premier ET le diviseur de tous les nombres de la forme abcdabcd

Partie 2 ----------

- Cpt = 0 *{ nous permet de compter le nombre premiers qui nous intéressent}*
- I = 2 *{ le nombre premier de départ}*
- On répète ce qui suit :
  - Si i est premier ET donne toujours un reste = 1 lorsqu'on le divise par tous les nombres compris entre 1 et 11
    - On incrémente Cpt de 1
    - On écrit i

i:=i+1;

et on s'arrête lorsque cpt = 11 *{on sait qu'il y 11 cas. Cf.énoncé}*

**Nota** : on aurait pu faire varier i de 2 à 1000 000

// Partie 3 ----------

a) On fait varier i de 1 à 15 *{on cherche les 15 premier multiples . Cf.énoncé}*
   - On multiplie i par 37*28
   - On extrait les 3 dernières positions que l'on met dans M et les autres dans L ( module CoupeNb)
   - On écrit i*37*38, M, L, M+L

b) Algorithme

```
Algorithme  ed1_1314
Variables  i, l, m, Cpt  : entier
Fonctions prem, Nb_pos, MemeReste, PreDiv
Procédure CoupeNb

DEBUT
I ← 2;
      Tant Que  (prem(i)= Faux ) OU  (PreDiv(i,4)= Faux ) Faire  i ←  i+1
Ecrire ('Partie 1 : ce premier nombre premier est : ',i)
Cpt ←  0;
I ←  2
 Ecrire ('Partie 2 : ces nombres sont :')
Répéter
    Si   prem(i) ET  memeReste(i, 2, 11, 1) = Vrai Alors
            DSI
            Cpt  ←  Cpt +1
            Ecrire (Cpt ,' : ',i,' ')
            FSI
  I ←  i+1
Jusqu'à  Cpt = 11

Ecrire ('Partie 3 : ')
Pour  i allant de  1 à 15  Faire
      Dpour
      coupeNb (i*37*28, 3, L, M)
      Ecrire (,i*37*28, M,  L, L+M)
      Fpour
FIN
```

c) Programme

```
Program ed1_1314;
uses crt;
var i,l,m,cpt :longint;
{$i E:\algo\modules\prem.fon}
{$i E:\algo\modules\Nb_pos.fon}
{$i E:\algo\modules\preDiv.fon}
{$i E:\algo\modules\CoupeNB.pro}
{$i E:\algo\modules\MemeRest.fon}
BEGIN
clrscr;
// Partie 1 ------------------------------------------------
i:= 2;
        while (prem(i)=false) or  (PreDiv(i,4)=false) do i:=i+1;
writeln ('Partie 1 : ce premier nombre premier est : ',i);

// Partie 2 ------------------------------------------------
cpt:=0;
i:=2;
 writeln('Partie 2 : ces nombres sont :');
repeat
     if   prem(i) and memeRest(i,2,11,1)= true then
                Begin
                cpt :=cpt +1;
                writeln (cpt,' : ',i,' ');
                end;
  i:=i+1;
until cpt =11;

// Partie 3 ------------------------------------------------
writeln('Partie 3 : ');
for i := 1 to 15  do
    Begin
    coupeNb(i*37*28,3,L,M);
    writeln(i,'*37*28= ',i*37*28,', ',M,'+', L,' =',L+M,' | Mod 37 = ',(L+M) mod 37);
    End;
write('ok');
readln;
END.
```

```
Free Pascal IDE
Partie 1 : ce premier nombre premier est : 73
Partie 2 : ces nombres sont :
1 : 55441
2 : 110881
3 : 332641
4 : 388081
5 : 415801
6 : 471241
7 : 498961
8 : 526681
9 : 748441
10 : 859321
11 : 970201
Partie 3 :
1*37*28= 1036, 1+36 =37 | Mod 37 = 0
2*37*28= 2072, 2+72 =74 | Mod 37 = 0
3*37*28= 3108, 3+108 =111 | Mod 37 = 0
4*37*28= 4144, 4+144 =148 | Mod 37 = 0
5*37*28= 5180, 5+180 =185 | Mod 37 = 0
6*37*28= 6216, 6+216 =222 | Mod 37 = 0
7*37*28= 7252, 7+252 =259 | Mod 37 = 0
8*37*28= 8288, 8+288 =296 | Mod 37 = 0
9*37*28= 9324, 9+324 =333 | Mod 37 = 0
10*37*28= 10360, 10+360 =370 | Mod 37 = 0
11*37*28= 11396, 11+396 =407 | Mod 37 = 0
12*37*28= 12432, 12+432 =444 | Mod 37 = 0
13*37*28= 13468, 13+468 =481 | Mod 37 = 0
14*37*28= 14504, 14+504 =518 | Mod 37 = 0
15*37*28= 15540, 15+540 =555 | Mod 37 = 0
```

BAREME

1. Ce corrigé est un corrigé type, toute autre solution sera retenue dans la mesure où elle est cohérente et les concepts appliqués
2. La démarche doit être modulaire sinon la note est divisée par 2 pour la partie ou cela n'est pas respecté – exception faite éventuellement à la partie 3
3. Tenir compte :
   - De la Justification du découpage
   - des Dessins des modules (interfaces cohérents, rôles clairs)
   - de la Structuration des idées dans l'analyse , la clarté, la précision, la concision et le soin
   - des Interfaces et rôles des modules faits en TDs
4. tout travail non soigné ne pourra en aucun cas prétendre à la note max. prévue dans le barème !

| | Analyse | Algorithme |
|---|---|---|
| PARTIE 1 | • Comment vérifier qu'un nombre divise tous les nombres de la forme abcdabcd ? **1.5 pts** | Respect de l'analyse proposée ,du formalisme et Présentation **2.5 pts** |
| PARTIE 2 | • Comment vérifier que l'on trouve toujours un reste =1 quand on divise un nombre par 2,3,4,…,11 **1.5 pts** | Respect de l'analyse proposée ,du formalisme et Présentation **2.5 pts** |
| PARTIE 3 | Comment on découpe un nombre en deux autres nombres ? **1.5 pts** | Respect de l'analyse proposée, du formalisme et Présentation **2.5 pts** |
| Algorithme principal | 1. comment est faite la boucle de recherche du Nbre premier de la partie 1 ? **0.5 pt**<br>2. comment est faite la boucle de recherche des 11 nbres premiers de la partie 2 ? **0.5 pt**<br>3. comment est faite la boucle de recherche des 15 resultats de la partie 3 , **0.5 pt** | justesse et cohérence des Déclarations **1 pt**<br>Respect de l'analyse proposée, du formalisme et de la Présentation 0.5 **pt**<br>Partie 1 : **1 pt**<br>Partie 2 : **1 pt**<br>Partie 3 : **1 pt** |
| Programmation sur 3 pts | **- 0.25 pt par erreur** de programmation<br>12 erreurs = 0<br>Tout programme qui n'aura pas un lien direct avec le problème posé et l'algorithme proposé ne sera pas pris en considération. | |