

EPREUVE OPTIONNELLE d'INFORMATIQUE CORRIGE

Question 1 : Le langage de communication avec les modems est :

- A - Unix
- B - Hayes
- C - ASCII
- D - Pascal

Question 2 : Le code de HUFFMAN est un code :

- A - de longueur fixe
- B - de longueur variable
- C - pour la téléphonie
- D - pour crypter les données

Question 3 : Le système d'exploitation Unix est :

- A - monotâche
- B - multitâche
- C - mono-utilisateur
- D - multi-utilisateur

Question 4 : L'architecture OSI correspond :

- A - à une architecture de base de données
- B - à une architecture de réseaux
- C - au modèle de référence de l'ISO
- D - à un organisme de normalisation

Question 5 : Une adresse physique (adresse MAC) est codée sur :

- A - 4 octets
- B - 6 octets
- C - 8 octets
- D - 12 octets

Question 6 : Une adresse logique IPV4 est codée sur :

- A - 4 octets
- B - 6 octets
- C - 8 octets
- D - 12 octets

Question 7 : Un connecteur RJ45 supporte un câble :

- A - fibre optique
- B - paire torsadée
- C - coaxial fin
- D - coaxial épais

Question 8 : La rapidité de modulation d'un signal électrique est exprimée en :

- A - bit/sec
- B - band
- C - Hertz
- D - decibel

Question 9 : Pour afficher une image animée en mode VGA à la cadence de 25 images par seconde, il faudra un débit de l'ordre de :

- A - 512 Kb/s
- B - 64 Kb/s
- C - 10 Mb/s
- D - 1 Gb/s

Question 10 : Qu'imprime le programme suivant :

```
main ()
{
    short s ;

    s = 5.8 + 2.3
    s = (short) 5.8 + (short) 2.3 ;
    printf (^`s = % d\n'', s) ;
}
```

- A - 8.1
- B - 7
- C - 8
- D - 9

Question 11 : Unix est une marque déposée des laboratoires :

- A - Microsoft
- B - Bell/ATT
- C - Digital
- D - Bull

Question 12 : Dans le codage en virgule flottante, la valeur 12300 en base 10 sera codée :

- A - $1.23 * 10^4$
- B - $12.3 * 10^3$
- C - $123 * 10^2$
- D - $1230 * 10$

Question 13 : Un firewall est un système de :

- A - gestion de base de données
- B - sécurité de données de l'entreprise
- C - communication
- D - partage de fichiers

Question 14 : Dans le codage EBCDIC, la valeur 128 est codée :

- A - 1001 1001
- B - 0100 0000
- C - 1100 1001
- D - 1000 0000

Question 15 : Pour faire communiquer une plateforme Unix avec une plateforme Windows-NT on utilise :

- A - un pont
- B - une passerelle
- C - un routeur
- D - un répéteur

Question 16 : Lequel de ces outils ne fait pas partie de la technologie Web :

- A - ASP
- B - JAVA
- C - Web Agency

D - Win - NT

Question 17 : Que signifie le sigle VPN

- A - Virtual Permanent Network
- B - Voie Privée Numérique
- C - Virtual Private Network
- D - Voice Private Node

Question 18 : Identifier le masque d'adresse de classe C parmi les adresse suivantes :

- A - 127
- B - 193.100.10.4
- C - 225.255.255.0
- D - 130.100.10.255

Question 19 : ADSL correspond à :

- A - une technologie d'accès à Internet
- B - un système de gestion de base de données
- C - un protocole de communication
- D - un accès di-symétrique sur liaison spécialisée

Question 20 : My SQL correspond à :

- A - un protocole d'accès à Internet
- B - une méthode de lecture séquentielle
- C - un système de gestion de base de données
- D - un forum de discussion sur internet

Question 21 : UMTS représente :

- A - l'union mondiale des télécommunications par satellite
- B - norme d'accès à Internet par le téléphone mobile
- C - une application de management des systèmes temps réel
- D - le module de gestion de Time-Sharing

Question 22 : L'intrajise correspond à :

- A - l'intranet de l'entreprise
- B - une relation entre entreprises
- C - l'appellation d'une entreprise en langue anglaise
- D - une erreur de frappe du mot « entreprise »

Question 23 : Dans le codage ASCII pur on utilise un 8^{ème} bit, dit bit de parité ; celui-ci sert à :

- A - étendre le code ASCII (ASCII étendu)
- B - synchroniser l'horloge du récepteur
- C - délimiter le message (fin du message)
- D - détecter les erreurs de transport

Question 24 : NTIC signifie :

- A - Nombre de Terminaux Internet Connectés
- B - Node Transport Identifier Code
- C - Nouvelles Techniques de l'Information Centralisée
- C - Nouvelles Technologies de l'Information et des Communications

REMARQUE :

Les programmes en Pascal et C sont des exemples et peuvent être discutés en terme d'implémentation et de construction. Le choix qui a été fait, est celui d'une découpe procédurale importante.

Cela permet de mieux faire ressortir un algorithme principal.

Ensuite chaque tâche est détaillée dans la procédure ou fonction correspondante.

D'autre part, le programme Pascal du premier exercice n'est pas optimisé, et se présente comme une traduction quasi-directe du programme C.

B) SCORES

Le principe est le suivant :

Une bonne solution à ce problème est, plutôt que de perdre son temps à évaluer toutes les combinaisons possibles, de rechercher le meilleur ensemble de "valeurs de manches" possible pour chaque score donné. La solution consiste alors à simplement afficher le bon ensemble de "valeurs de manches".

Si l'on prend, par exemple, les scores 4 et 5, nous pouvons, au travers d'une liste, trouver toutes les "valeurs de manches" possibles, et la meilleure façon de les obtenir.

Scores Ensemble de "valeurs de manches" retenu

0	none)
4	4
5	5
8	4+4
9	4+5
10	5+5
12	4+4+4
13	4+4+5
14	4+5+5
15	5+5+5
16	4+4+4+4
17	4+4+4+5
18	4+4+5+5
19	4+5+5+5
20	5+5+5+5 (pas 4+4+4+4+4)
...	

Nous possédons alors pour un total, la combinaison et le nombre optimal de manches nécessaires à son obtention. En utilisant les scores connus, nous pouvons déterminer de nouveaux totaux et vérifier que leur combinaison soit aussi optimale.

Considérant que les données d'entrée sont affectées aux variables suivantes:

- `n` - Le nombre de "valeur de manche" possibles
- `m` - Le nombre de scores
- `Valeurs_Manches` - Un tableau donnant les "valeur de manche" possibles
- `scores` - Un tableau donnant les scores recherchés

Pour tous les nombres de 0 à 1000, nous pouvons représenter les meilleurs scores dans deux tableaux de 1001 éléments :

- `Nombres_Manches[i]` - Le nombre minimum de manches requises pour obtenir un total de `i`
- `Dernier_score[i]` - Le score dans la dernière manche d'un jeu donnant un total de `i`.

`Nombres_Manches[0]` vaut 0 et les autres élément peuvent être initialisés avec une valeur "bidon" (par exemple 9999), le but étant de signaler que ce score ne peut pas être obtenu.

Partant de 0, nous remontons dans la liste jusqu'à un total de 1000. Si une valeur donnée est un score possible (`Nombres_Manches[i] < 9999`), alors pour chaque score `Valeurs_Manches[j]` nous savons que nous

pouvons obtenir un nouveau total $i + \text{Valeur_Manches}[j]$ en $1 + \text{Nombres_Manches}[i]$ manches. Tout ce que nous avons à faire, est de vérifier que c'est actuellement la façon la plus optimale d'obtenir ce score et, si c'est le cas, d'affecter à `Nombres_Manches` et `Dernier_Score` la valeur appropriée. Une fois que nous avons testé les m scores, nous passons au total valide suivant, sachant que nous avons évalué toutes les façons de l'obtenir. Le dernier point consiste à trouver tous les scores donnant un total. Pour cela, on utilisera le tableau `Dernier_Score` manche par manche et ce de manière décroissante.

Programme Pascal (TP7)

```

program JeuxScores;
var
    n, m: integer;                                (* Données en entrée *)
    Valeurs_Manches: array[1..10] of integer;
    Scores: array[1..10] of integer;

    Nombres_Manches: array[0..1000] of integer;
    Dernier_Score: array[0..1000] of integer;

    FichierEntree,                                (* Fichier TEXT pour
l'entrée *)
    FichierSortie : Text;                          (* Fichier TEXT pour la
sortie *)

procedure OuvertureFichier;
begin
    Assign(FichierEntree, 'SCORES.IN');
    Reset(FichierEntree);                          (* Ouverture de SCORES.IN en
lecture *)
    Assign(FichierSortie, 'SCORES.OUT');
    Rewrite(FichierSortie);                        (* Ouverture de SCORES.OUT
en écriture *)
end;

procedure FermetureFichier;
begin
    Close(FichierEntree);
    Close(FichierSortie);
end;

Procedure LectureDonnee;
var i: integer;
begin
    (* Lecture du nombre de manches et de leur valeur respective *)
    readln(FichierEntree, n);
    for i := 1 to n do readln( FichierEntree, Valeurs_Manches[i] );
    end;
    (* Lecture du nombre de scores totaux et de leur valeur respective *)
    readln(FichierEntree, m);
    for i := 1 to m do readln( FichierEntree, Scores[i] );
end;

procedure RechercheScoresPossibles;
var i, j: integer;
begin
    (* Initialisation *)
    for i := 1 to 1000 do begin
        Nombres_Manches[i] := 9999;
        Dernier_Score[i] := 9999;
    end;
    Nombres_Manches[0] := 0;

```

```

Dernier_Score[0] := 0;

(* Trouve les valeurs de manches necessaires à obtenir chaque score *)
i := 0;
while i <= 1000 do begin
  (* Pour chaque score possible, peut-on faire mieux ? *)
  for j := 1 to n do if i + Valeurs_Manches[j] <= 1000 then
    if Nombres_Manches[i + Valeurs_Manches[j]] > Nombres_Manches[i]
+ 1
      then begin
        Nombres_Manches[i + Valeurs_Manches[j]] :=
Nombres_Manches[i] + 1;
        Dernier_Score[i + Valeurs_Manches[j]] :=
Valeurs_Manches[j];
        end;

        (* Cherche le prochain score valide *)
        repeat
          inc(i);
          if i > 1000 then break; (* Hors limite *)
          until Nombres_Manches[i] < 9999; (* Score valide *)
        end;
      end;

end;

procedure Resolution;
var i, j, p, c: integer;
begin
  RechercheScoresPossibles;
  (* Est-ce qu'il existe pour chaque score demandé une solution ? *)
  for i := 1 to m do begin
    if (Scores[i] < 1) or (Scores[i] > 1000) then
      writeln( FichierSortie, 'Score ', Scores[i], ' impossible à
obtenir' )
    else if Nombres_Manches[Scores[i]] = 9999 then
      writeln( FichierSortie,'Score ', Scores[i], ' impossible à
obtenir' )
    else begin
      write(FichierSortie, 'Score ',Scores[i],' en
',Nombres_Manches[Scores[i]],' manches:');
      (* trouve le nombre de fois où chaque valeur est nécessaire *)
      for p := 1 to n do begin
        c := 0;
        j := Scores[i];
        while j > 0 do begin
          if Dernier_Score[j] = Valeurs_Manches[p] then inc(c);
          j := j - Dernier_Score[j];
        end;
        if c > 0 then write( FichierSortie, ' ', c, 'x',
Valeurs_Manches[p] );
        end;
        writeln(FichierSortie);
      end;
    end;
  end;

end;

begin
  OuvertureFichier;
  LectureDonnee;
  Resolution;
  FermetureFichier;
end.

```

Programme C (CC)

```
#include <stdio.h>
#include <stdlib.h>

FILE      *FichierEntree;  /* Fichier TEXT pour l'entrée */
FILE      *FichierSortie; /* Fichier TEXT pour la sortie */

int n, m;                                     (* Données en entrée *)
int Valeurs_Manches[11], Scores[11];

int Nombres_Manches[1001], Dernier_Score[1001];

void OuvertureFichier()
{
    if ((FichierEntree = fopen("SCORES.IN", "r")) == NULL)
    {
        perror("SCORES.IN");
        exit(1);
    }
    rewind(FichierEntree);
    /* Ouverture de SCORES.IN en lecture */
    FichierSortie = fopen("SCORES.OUT", "w");
    if (FichierSortie != NULL)
        rewind(FichierSortie);
    else
        FichierSortie = tmpfile();
    if (FichierSortie == NULL)
    {
        perror("FichierSortie");
        exit(1);
    }
    /* Ouverture de SCORES.OUT en écriture */
}

void FermetureFichier()
{
    if (FichierEntree != NULL)
        fclose(FichierEntree);
    FichierEntree = NULL;
    if (FichierSortie != NULL)
        fclose(FichierSortie);
    FichierSortie = NULL;
}

void LectureDonnee()
{
    int i;

    (* Lecture du nombre de manches et de leur valeur respective *)
    fscanf(FichierEntree, "%d", &n );
    for (i=1;i<=n;i++) fscanf( FichierEntree, "%d", &Valeurs_Manches[i] );

    (* Lecture du nombre de scores totaux et de leur valeur respective *)
    fscanf(FichierEntree, "%d", &m );
    for (i=1;i<=m;i++) fscanf( FichierEntree, "%d", &Scores[i] );
}
```

```

void RechercheScoresPossibles()
{
    int i, j;

    (* Initialisation *)
    for(i=1;i<=1000;i++)
    {
        Nombres_Manches[i] = 9999;
        Dernier_Score[i] = 9999;
    }
    Nombres_Manches[0] = 0;
    Dernier_Score[0] = 0;

    (* Trouve les valeurs de manches necessaires à obtenir chaque score *)
    i = 0;
    while(i <= 1000)
    {
        (* Pour chaque score possible, peut-on faire mieux ? *)
        for(j=1;j<=n;j++)
            if (i + Valeurs_Manches[j] <= 1000)
                if (Nombres_Manches[i + Valeurs_Manches[j]] > Nombres_Manches[i] +
1)
                    {
                        Nombres_Manches[i + Valeurs_Manches[j]] = Nombres_Manches[i] + 1;
                        Dernier_Score[i + Valeurs_Manches[j]] = Valeurs_Manches[j];
                    }

                (* Cherche le prochain score valide *)
                do {
                    i++;
                    if (i > 1000) break; (* Hors limite *)
                } while ( Nombres_Manches[i] >= 9999); (* Score valide *)
            }
    }
}

void Resolution()
{
    int i, j, p, c;

    RechercheScoresPossibles();
    (* Est-ce qu'il existe pour chaque score demandé une solution ? *)
    for(i=1;i<=m;i++)
    {
        if (Scores[i] < 1) || (Scores[i] > 1000)
            fprintf( FichierSortie, 'Score %d impossible à obtenir\n',
Scores[i] )
        else if (Nombres_Manches[Scores[i]] = 9999)
            fprintf( FichierSortie, 'Score %d impossible à obtenir\n',
Scores[i] )
        else {
            fprintf(FichierSortie, 'Score %d en %d manches:',Scores[i],
Nombres_Manches[Scores[i]]);
            (* trouve le nombre de fois où chaque valeur est nécessaire *)
            for(p=1;p<=n;p++)
            {
                c=0;
                j=Scores[i];
                while (j > 0)
                {
                    if (Dernier_Score[j] = Valeurs_Manches[p]) c++;
                }
            }
        }
    }
}

```

```

                j -= Dernier_Score[j];
            }
            if (c > 0) fprintf( FichierSortie, ' %dx%d', c,
Valeurs_Manches[p] );
        }
        fprintf(FichierSortie, "\n");
    }
}

void main()
{
    OuvertureFichier();
    LectureDonnee();
    Resolution();
    FermetureFichier();
}

```

C) NOMBRE ROMAINS

Le principe est le suivant :

Une façon rapide de résoudre ce problème est de remarquer que les milliers, les centaines, les dizaines et les unités peuvent être traduites indépendamment et dans l'ordre. Dans ce cas, il suffit d'avoir pour chacun d'eux un type fournissant pour chaque digit la chaîne de caractère appropriée.

Ce qui pourrait donner un tableau pour chaque puissance de 10 et les implémentations suivantes:

Programme Pascal (TP7)

```

program codes;
uses crt;

const
(* Unités de 0 à 90 *)
    unites = ( '', 'I', 'II', 'III', 'IV', 'V', 'VI', 'VII', 'VIII', 'IX' );

(* Dizaines de 0 à 90 *)
    dizaines = ( '', 'X', 'XX', 'XXX', 'XL', 'L', 'LX', 'LXX', 'LXXX', 'XC'
);

(* Centaines de 0 à 900 *)
    centaines = ( '', 'C', 'CC', 'CCC', 'CD', 'D', 'DC', 'DCC', 'DCCC', 'CM'
);

(* Milliers de 0 à 3000 *)
    milliers = ( '', 'M', 'MM', 'MMM' );

var
    FichierEntree, (* Fichier TEXT pour
l'entrée *)
    FichierSortie : Text; (* Fichier TEXT pour la
sortie *)

procedure OuvertureFichier;
begin
    Assign(FichierEntree, 'ROMAIN.IN');
    Reset(FichierEntree); (* Ouverture de ROMAIN.IN en
lecture *)
    Assign(FichierSortie, 'ROMAIN.OUT');

```

```

    Rewrite(FichierSortie);
en écriture *)
end;

procedure FermetureFichier;
begin
    Close(FichierEntree);
    Close(FichierSortie);
end;

procedure LectureConversionEcriture;
var
    i,Erreur : integer;
    lect      : string;
begin
    while not Eof(FichierEntree) do
        begin
            readln(FichierEntree, lect);
            val(lect,i,Erreur);
            if Erreur<>0
            then writeln(FichierSortie,'Nombre invalide')
            else writeln(FichierSortie,milliers[(i DIV 1000+1)+centaines[(i DIV
100) MOD 10+1]+ dizaines[(i DIV 10) MOD 10+1]+unites[i MOD 10+1]);
            end;
        end;
    end;

begin
    OuvertureFichier;
    LectureConversionEcriture;
    FermetureFichier;
end.

```

Programme C (GCC)

```

/* Programme Nombre_romain */

#include <stdio.h>
#include <stdlib.h>

FILE      *FichierEntree; /* Fichier TEXT pour l'entrée */
FILE      *FichierSortie; /* Fichier TEXT pour la sortie */

// Unités de 0 à 9
static char*[] unites = { '', 'I', 'II', 'III', 'IV', 'V', 'VI', 'VII',
'VIII', 'IX' };

// Dizaines de 0 à 90
static char*[] dizaines = { '', 'X', 'XX', 'XXX', 'XL', 'L', 'LX', 'LXX',
'LXXX', 'XC' };

// Centaines de 0 à 900
static char*[] centaines = { '', 'C', 'CC', 'CCC', 'CD', 'D', 'DC', 'DCC',
'DCCC', 'CM' };

// Milliers de 0 à 3000
static char*[] milliers = { '', 'M', 'MM', 'MMM' };

void OuvertureFichier()
{
    if ((FichierEntree = fopen("ROMAIN.IN", "r")) == NULL)

```

```

    {
        perror("ROMAIN.IN");
        exit(1);
    }
rewind(FichierEntree);
/* Ouverture de ROMAIN.IN en lecture */
FichierSortie = fopen("ROMAIN.OUT", "w");
if (FichierSortie != NULL)
    rewind(FichierSortie);
else
    FichierSortie = tmpfile();
if (FichierSortie == NULL)
    {
        perror("FichierSortie");
        exit(1);
    }
/* Ouverture de ROMAIN.OUT en écriture */
}

void FermetureFichier()
{
    if (FichierEntree != NULL)
        fclose(FichierEntree);
    FichierEntree = NULL;
    if (FichierSortie != NULL)
        fclose(FichierSortie);
    FichierSortie = NULL;
}

char *Conversion(char *nombre)
{
    int n;
    char *temp = (char *) malloc(16); /* au maximum MMMDCCCLXXXVIII */

    n=atoi(Nombre);
    temp[0]='\0';
    strcat(milliers[(n/1000)],temp);
    strcat(centaines[(n/100)%10],temp);
    strcat(dizaines[(n/10)%10],temp);
    strcat(unites[(n)%10],temp);
    return temp;
}

void LectureConversionEcriture()
{
    char *lect=(char *) malloc(5); /* Au maximum 3999 */

    while (fgets(lect, 4, FichierEntree) != NULL)
        {
            fprintf(FichierSortie, "%s\n", Conversion(lect));
        }
}

void main()
{
    OuvertureFichier();
    LectureConversionEcriture();
    FermetureFichier();
}

```