

## EPREUVE OPTIONNELLE d'INFORMATIQUE CORRIGE

Question 1 : Un paquet ou trame comprend :

- A - uniquement des données utilisateur
- B - un en-tête et des données
- C - des unités de transmission de taille fixe
- D - uniquement des informations d'adressage et de contrôle

Question 2 : Les principales ressources partagées au sein d'un groupe de travail comprennent les fichiers, les imprimantes et :

- A - l'information sur la conception du réseau
- B - les outils de maintenance
- C - les systèmes d'exploitation clients
- D - les services de communication

Question 3 : Un câble de catégorie 5 fait référence à :

- A - un câble coaxial
- B - la meilleure paire torsadée non blindée
- C - la fibre optique monomode
- D - la paire téléphonique ordinaire

Question 4 : IPX de Novell diffère de IP en ce sens que IPX ne supporte pas :

- A - des adresses de la forme <réseau> <hôte>
- B - la fragmentation de paquets
- C - le routage de paquets
- D - le service en mode non connecté

Question 5 : Les systèmes Clients et Serveurs de Windows NT supportent :

- A - uniquement les protocoles TCP/IP
- B - les protocoles TCP/IP, Novell et LLC
- C - uniquement les protocoles Novell
- D - les protocoles TCP/IP, Novell, Decnet, SNA

Question 6 : L'émulation d'un équipement peut se faire en sorte qu'un PC :

- A - ressemble et se comporte comme l'équipement émulé
- B - communique avec un grand système en utilisant NETBIOS
- C - transporte de l'information sur des liaisons de grande distance de manière plus performante
- D - se comporte comme un grand système

Question 7 : La représentation suivante « 193. 16. 8. 2 » correspond à :

- A - une adresse MAC
- B - un numéro de port
- C - une adresse IP
- D - un numéro de SAP

Question 8 : Le système Unix est un système :

- A - multi-tâches
- B - mono-tâches
- C - multi-utilisateurs
- D - mono-utilisateurs

Question 9 : Les instructions suivantes sont rédigées en langage :

```
main {  
  char c1 [1000], c2 [1000]  
  int i, c ;  
  i = 0  
  while ((c =getchar( ) ) != | \ n)  
    {      c[i] = c ;  
      if (- -  
    } ;  
  printf ( ^ % f \ n - - - - - ) ;  
}
```

- A - Pascal
- B - Cobol
- C - Scheme
- D - C

Question 10 : L'informatisation consistant à répartir les traitements entre un poste de travail et un serveur est une architecture de type :

- A - systèmes répartis
- B - client / serveur
- C - systèmes distribués
- D - interrogation de bases de données relationnelles

Question 11 : Le contrôle de parité est une technique:

- A - de détection d'erreur
- B - de compression de données
- C - de cryptage
- D - de multiplexage

Question 12 : Les différents éléments d'un ordinateur (mémoire, processeurs, périphériques,...) sont reliés par :

- A - des registres
- B - des pointeurs
- C - le système d'exploitation
- D - des bus

Question 13 : Parmi les mémoires suivantes quelles sont celles qui sont volatiles ? :

- A - RAM
- B - ROM
- C - EPROM
- D - mémoire cache

Question 14 : Quel est le langage de manipulation des bases de données ? :

- A - C
- B - HTML
- C - SQL
- D - JAVA

Question 15 : Dans les équipements informatiques les données sont représentées par un signal électrique de la forme :

- A - analogique
- B - numérique
- C - alphanumérique
- D - alphabétique

Question 16 : Cette suite d'instructions est décrite en langage :

```
Var res : real ;  
begin  
    readln(res) ;  
    writeln(res/2) ;  
end
```

- A - Pascal
- B - C
- C - COBOL
- D - Scheme

  
  
  

Question 17 : Que fait le programme précédent (question 16) ? :

- A - affiche « res »
- B - lit « ln »
- C - lit un nombre au clavier et affiche un autre nombre
- D - calcule la factorielle

  
  
  

Question 18 : Quel est le débit maximum autorisé sur une ligne téléphonique à l'aide d'un modem ? :

- A - 28800 b/s
- B - 64 Kb/s
- C - 9600 b/s
- D - 56 Kb/s

  
  
  

Question 19 : En langage « C » un « long » mesure :

- A - 2 octets
- B - 4 octets
- C - 8 octets
- D - variable

  
  
  

Question 20 : Int \*pointeur :

- A - décrit un pointeur sur un entier
- B - signale une remarque sur « pointeur »
- C - déclare un tableau
- D - rappelle une commande

  
  
  

Question 21 : « IP » (Internet Protocol) décrit :

- A - une messagerie électronique
- B - un transfert de fichier
- C - un protocole de communication
- D - le réseau Ethernet

  
  
  

Question 22 : Quelle est la spécialisation qui n'est pas assurée par l'EPITA :

- A - système et réseau
- B - télécommunication
- C - multimédia
- D - hardware

  
  
  

Question 23 : Le bus micro EISA (Extended Industry Standard Architecture) utilise une longueur de données de :

- A - 8 bits
- B - 16 bits
- C - 32 bits
- D - 64 bits

  
  
  

Question 24 : Un bus d'adresses sur 16 bits représente une capacité d'adressage maximale de :

- A -  $16^2$
- B -  $2^{16}$
- C -  $8^2$

**REMARQUE :**

*Les programmes en Pascal et C sont des exemples et peuvent être discutés en terme 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 en C n'est pas optimisé, et se présente comme une Traduction quasi-directe du programme Pascal.*

**ECHELLES ET TOBOGGANS****Le principe est le suivant :**

Nous récupérons dans un fichier (ECHELLE.IN) les jets de dés que nous plaçons dans un vecteur Jets. Puis, pour chaque partie, nous récupérons le nombre de joueurs et les différentes cases spéciales qui nous permettent de mettre à jour un vecteur Plateau. Une table des joueurs contient la position et l'état de chaque joueur (passe ou rejoue). Ensuite, la partie est jouée, et si un vainqueur est déterminé (ce qui est le cas systématiquement), son numéro est écrit dans un fichier de sortie (ECHELLE.OUT) et cela pour chaque partie.

**Programme Pascal (TP7)**

```

program echelles_toboggans;
uses crt;
const
  Passe = 101;
  Rejoue = 102;
var
  FichierEntree      : Text;                (* Fichier TEXT pour l'entrée *)
  FichierSortie     : Text;                (* Fichier TEXT pour la sortie *)
  Jets               : array[1..1000] of byte;
  Plateau            : array[1..100] of byte;
  NbJoueur           : byte;
  TableDeJoueur     : array[1..5,1..2] of integer;  (* 1 ... 5 joueurs *)
                                                    (* case actuelle(1), case passe le tour(2) *)

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

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

procedure LectureJetsdeDes;                (* Lecture des données *)
Var
  i : Integer;
begin
  i:=1;                                (* Récupération des jets de dés *)
  Repeat
    Read(FichierEntree,Jets[i]);
    write(Jets[i], ' ');
    Inc(i);
  until Jets[i-1]=0;
  writeln;
end;

```

```

procedure LecturePartie;          (* Récupération d'une partie *)
var
  i          : Integer;
  CaseDepart,
  CaseArrivee : byte;
begin
  For i:=0 to 99 do Plateau[i]:=0;      (* initialisation des données *)
  Readln(FichierEntree,NbJoueur);      (* Récupération du nombre de joueurs *)
  if NbJoueur<>0 then
  begin
    For i:=1 to NbJoueur do
    begin
      TableDeJoueur[i,1]:=0;
      TableDeJoueur[i,2]:=Rejoue;
    end;
    Repeat                                (* Récupération échelles & toboggans *)
      Readln(FichierEntree,CaseDepart,CaseArrivee);
      if (CaseDepart<>0) and (CaseArrivee<>0) then
        Plateau[CaseDepart]:=CaseArrivee;
    until (CaseDepart=0) and (CaseArrivee=0);
    Repeat                                (* Récupération des cases spéciales *)
      Readln(FichierEntree,I);
      if i<0 then Plateau[abs(i)]:=Passe
      else if i>0 then Plateau[i]:=Rejoue;
    until i=0;
  end;
end;

```

```

procedure JouerPartie;
Var
  NumJoueur,
  NumJet : integer;
  Continue : boolean;
begin
  Numjoueur:=1;
  NumJet:=1;
  repeat
    while TableDeJoueur[Numjoueur,2]=Passe do
    begin
      TableDeJoueur[Numjoueur,2]:=Rejoue;
      NumJoueur:=NumJoueur mod NbJoueur + 1;
    end;
    if TableDeJoueur[NumJoueur,1]+Jets[Numjet]=100 then
    begin
      Writeln(FichierSortie,NumJoueur);
      exit;
    end;
    if TableDeJoueur[NumJoueur,1]+Jets[Numjet]<100 then
    begin
      TableDeJoueur[NumJoueur,1]:=TableDeJoueur[NumJoueur,1]+Jets[Numjet];
      Continue:=True;
      repeat
        if Plateau[TableDeJoueur[NumJoueur,1]]=0 then continue:=False
        else if Plateau[TableDeJoueur[NumJoueur,1]]=Passe then
        begin
          TableDeJoueur[NumJoueur,2]:=Passe;
          Continue:=False;
        end
        else if Plateau[TableDeJoueur[NumJoueur,1]]=Rejoue then
        begin

```

```

        Inc(Numjet);
        If TableDeJoueur[NumJoueur,1]+Jets[Numjet]=100 then
        begin
            Writeln(FichierSortie,NumJoueur);
            exit;
        end;
        If TableDeJoueur[NumJoueur,1]+Jets[Numjet]<100 then
            TableDeJoueur[NumJoueur,1]:=TableDeJoueur[NumJoueur,1]+Jets[Numjet]
        else continue:=False;
        end
        else TableDeJoueur[NumJoueur,1]:=Plateau[TableDeJoueur[NumJoueur,1]];
    until Not(Continue);
end;
Inc(Numjet);
NumJoueur:=NumJoueur mod NbJoueur +1;
until Jets[Numjet]=0;
end;

begin
    OuvertureFichier;
    LectureJetsdeDes;
    Repeat
        LecturePartie;
        if NbJoueur<>0 then JouerPartie;
    until NbJoueur=0;
    FermetureFichier;
end.

```

### **Programme C (CC)**

```

#include <stdio.h>

typedef unsigned char  boolean;

#ifndef TRUE
# define TRUE      1
# define FALSE    0
#endif

#define Passe      101
#define Rejoue    102

FILE      *FichierEntree; /* Fichier TEXT pour l'entrée */
FILE      *FichierSortie; /* Fichier TEXT pour la sortie */
unsigned char  Jets[1000];
unsigned char  Plateau[100];
unsigned char  NbJoueur;
long          TableDeJoueur[5][2]; /* 1 à 5 joueurs */

/* case actuelle(1), case passe le tour(2) */
void OuvertureFichier()
{
    if ((FichierEntree = fopen("ECHELLE.IN", "r")) == NULL)
    {
        perror("ECHELLE.IN");
        exit(1);
    }
    rewind(FichierEntree);
    /* Ouverture de ECHELLE.IN en lecture */
    FichierSortie = fopen("ECHELLE.OUT", "w");
    if (FichierSortie != NULL)

```

```

    rewind(FichierSortie);
else
    FichierSortie = tmpfile();
if (FichierSortie == NULL)
    {
        perror("FichierSortie");
        exit(1);
    }
/* Ouverture de ECHELLE.OUT en écriture */
}

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

void LectureJetsdeDes()
{
    /* Lecture des données */
    long    i;
    int temp;

    i = 1;    /* Récupération des jets de dés */
    do {
        fscanf(FichierEntree, "%d", &temp);
        Jets[i - 1] = temp;
        i++;
    } while (Jets[i - 2] != 0);
    putchar('\n');
}

void LecturePartie()
{
    /* Récupération d'une partie */
    long    i;
    unsigned char CaseDepart, CaseArrivee;
    int     temp, temp1;

    for (i = -1; i <= 98; i++)    /* initialisation des données */
        Plateau[i] = 0;
    fscanf(FichierEntree, "%d%*[^\\n]", &temp);
    getc(FichierEntree);    /* Récupération du nombre de joueurs */
    NbJoueur = temp;
    if (NbJoueur == 0)
        return;
    for (i = 0; i < NbJoueur; i++)
    {
        TableDeJoueur[i][0] = 0;
        TableDeJoueur[i][1] = Rejoue;
    }
    do {    /* Récupération des échelles et toboggans */
        fscanf(FichierEntree, "%d%d%*[^\\n]", &temp, &temp1);
        getc(FichierEntree);
        CaseDepart = temp;
        CaseArrivee = temp1;
        if (CaseDepart != 0 && CaseArrivee != 0)

```

```

    Plateau[CaseDepart - 1] = CaseArrivee;
} while (CaseDepart != 0 || CaseArrivee != 0);
do { /* Récupération des cases spéciales */
    fscanf(FichierEntree, "%ld%*[\n]", &i);
    getc(FichierEntree);
    if (i < 0)
        Plateau[labs(i) - 1] = Passe;
    else if (i > 0)
        Plateau[i - 1] = Rejoue;
} while (i != 0);
}

void JouerPartie()
{
    long NumJoueur, NumJet;
    boolean Continue;

    NumJoueur = 1;
    NumJet = 1;
    do {
        while (TableDeJoueur[NumJoueur - 1][1] == Passe)
        {
            TableDeJoueur[NumJoueur - 1][1] = Rejoue;
            NumJoueur = NumJoueur % NbJoueur + 1;
        }
        if (TableDeJoueur[NumJoueur - 1][0] + Jets[NumJet - 1] == 100)
        {
            fprintf(FichierSortie, "%12ld\n", NumJoueur);
            return;
        }
        if (TableDeJoueur[NumJoueur - 1][0] + Jets[NumJet - 1] < 100)
        {
            TableDeJoueur[NumJoueur - 1][0] += Jets[NumJet - 1];
            Continue = TRUE;
            do {
                if (Plateau[TableDeJoueur[NumJoueur - 1][0] - 1] == 0)
                    Continue = FALSE;
                else if (Plateau[TableDeJoueur[NumJoueur - 1][0] - 1] == Passe)
                {
                    TableDeJoueur[NumJoueur - 1][1] = Passe;
                    Continue = FALSE;
                }
            } else
            {
                if (Plateau[TableDeJoueur[NumJoueur - 1][0] - 1] == Rejoue)
                {
                    NumJet++;
                    if (TableDeJoueur[NumJoueur - 1][0] + Jets[NumJet - 1] == 100)
                    {
                        fprintf(FichierSortie, "%12ld\n", NumJoueur);
                        return;
                    }
                    if (TableDeJoueur[NumJoueur - 1][0] + Jets[NumJet - 1] < 100)
                        TableDeJoueur[NumJoueur - 1][0] += Jets[NumJet - 1];
                    else
                        Continue = FALSE;
                }
            } else
            {
                TableDeJoueur[NumJoueur - 1]
                [0] = Plateau[TableDeJoueur[NumJoueur - 1][0] - 1];
            } while (Continue);
        }
    }
}

```

```

        NumJet++;
        NumJoueur = NumJoueur % NbJoueur + 1;
    } while (Jets[NumJet - 1] != 0);
}

int main(int argc, char **argv)
{
    FichierSortie = NULL;
    FichierEntree = NULL;
    OuvertureFichier();
    LectureJetsdeDes();
    scanf("%*[^\\n]");
    getchar();
    do {
        LecturePartie();
        scanf("%*[^\\n]");
        getchar();
        if (NbJoueur != 0)
            JouerPartie();
    } while (NbJoueur != 0);
    FermetureFichier();
    if (FichierEntree != NULL)
        fclose(FichierEntree);
    if (FichierSortie != NULL)
        fclose(FichierSortie);
    exit(0);
}

```

## IMMEDIATEMENT DECODABLE

### Le principe est le suivant :

Nous commençons par lire dans un fichier (CODES.IN) les différents codes d'une série en les plaçant dans une table de code. Puis code par code selon leur longueur, ils sont comparés. Si un code est inclus en préfixe dans un autre, on écrit dans le fichier de sortie (CODES.OUT) que ce code n'est pas immédiatement décodable. Dans le cas contraire, l'écriture stipulera qu'il l'est. Puis l'on passe à la série suivante si elle existe.

### Programme Pascal (TP7)

```

program codes;
uses crt;
var
    FichierEntree,                (* Fichier TEXT pour l'entrée *)
    FichierSortie    : Text;      (* Fichier TEXT pour la sortie *)
    TableCode        : array[1..9] of string[10];
    NbCodes,
    NumEnsemble      : Byte;

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

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

```

```

procedure LectureCodes;                                (* Lecture des données *)
Var
  i : Integer;
begin
  i:=1;                                                (* Récupération des jets de dés *)
  while not(eof(FichierEntree)) do
  begin
    Readln(FichierEntree,TableCode[i]);
    if TableCode[i]<>'9' then inc(i)
    else begin
      NbCodes:=i;
      exit;
    end;
  end;
  NbCodes:=i-1;
end;

procedure AnalyseCodes;
Var
  i,j : byte;
begin
  For i:=1 to NbCodes do
  For j:=1 to NbCodes do
  begin
    if i<>j then
    begin
      if length(TableCode[j])>=Length(TableCode[i]) then
      if copy(TableCode[j],1,length(TableCode[i]))=TableCode[i] then
      begin
        writeln(FichierSortie,'L"ensemble ',NumEnsemble,' n"est pas immédiatement décodable. ');
        exit;
      end;
    end;
  end;
  end;
  if NbCodes<>0 then
  writeln(FichierSortie,'L"ensemble ',NumEnsemble,' est immédiatement décodable. ');
end;

begin
  OuvertureFichier;
  NumEnsemble:=1;
  Repeat
    LectureCodes;
    AnalyseCodes;
    Inc(NumEnsemble);
  until Eof(FichierEntree);
  FermetureFichier;
end.

```

### **Programme C (GCC)**

```

#include <stdio.h>
#include <strings.h>

FILE      *FichierEntree;  /* Fichier TEXT pour l'entrée */
FILE      *FichierSortie; /* Fichier TEXT pour la sortie */
static char TableCode[9][11];
unsigned char    NbCodes, NumEnsemble;

void OuvertureFichier()
{

```

```

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

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

void LectureCodes()
{
    /* Lecture des données */
    long    i;
    char    *temp;

    i = 1; /* Récupération des jets de dés */
    while (fgets(TableCode[i - 1], 11, FichierEntree) != NULL)
    {
        temp = strchr(TableCode[i - 1], '\n');
        if (temp != NULL)
            *temp = 0;
        if (!strcmp(TableCode[i - 1], "9"))
        {
            NbCodes = i;
            return;
        }
        i++;
    }
    NbCodes = i - 1;
    exit(0);
}

void AnalyseCodes()
{
    unsigned char    i, j;
    char            str1[256];

    for (i = 0; i < NbCodes; i++)
    {
        for (j = 0; j < NbCodes; j++)

```

```

    {
        if (i + 1 != j + 1)
        {
            if (strlen(TableCode[j]) >= strlen(TableCode[i]))
            {
                sprintf(str1, "%.*s", strlen(TableCode[i]), TableCode[j]);
                if (!strcmp(str1, TableCode[i]))
                {
                    fprintf(FichierSortie,
                        "L'ensemble %12d n'est pas immédiatement décodable.\n",
                            NumEnsemble);
                    return;
                }
            }
        }
    }
}

if (NbCodes != 0)
    fprintf(FichierSortie,
        "L'ensemble %12d est immédiatement décodable.\n",
            NumEnsemble);
}

int main(int argc, char **argv)
{
    FichierSortie = NULL;
    FichierEntree = NULL;
    OuvertureFichier();
    NumEnsemble = 1;
    do {
        LectureCodes();
        AnalyseCodes();
        NumEnsemble++;
    } while (!feof(FichierEntree));
    FermetureFichier();
    if (FichierEntree != NULL)
        fclose(FichierEntree);
    if (FichierSortie != NULL)
        fclose(FichierSortie);
    exit(0);
}

```