

EPREUVE OPTIONNELLE d'INFORMATIQUE CORRIGE

Question 1 : Le système WIN98 est un système :

- A - 16 bits
- B - 32 bits
- C - personnel
- D - professionnel

Question 2 : Le système WIN2000 est un système :

- A - 16 bits
- B - 32 bits
- C - personnel
- D - professionnel

Question 3 : Parmi les langages suivants, lequel est ou lesquels sont orienté(s) objet ? :

- A - C
- B - C++
- C - JAVA
- D - PASCAL

Question 4 : Quels sont les outils de messagerie instantanée sur Internet :

- A - eudora
- B - outlook
- C - ICQ
- D - yahoo messenger

Question 5 : Quels sont les fabricants de micro-processeurs ? :

- A - CISCO
- B - AMD
- C - IBM
- D - INTEL

Question 6 : Quelles est la complexité en pire cas de l'algorithme de détermination d'automate ? (avec n, le nombre de l'automate pris comme quantité caractéristique) :

- A - $O(2^n)$
- B - $O(n \log(n))$
- C - $O(n^2)$
- D - $O(n \log(\log(n)))$

Question 7 : Quels sont les fabricants de chipset de cartes graphiques :

- A - IBM
- B - ATI
- C - AMD
- D - NVIDIA

Question 8 : Parmi les entreprises suivantes, quelles sont celles qui proposent des services de transport de l'information sur longues distances ? :

- A - CISCO
- B - CEGETEL
- C - COLT
- D - BULL

Question 9 : Soit la procédure suivante :

```
Void main (int argc, char ** argv)
{
    printf ("hello world" );
}
```

identifier le langage utilisé :

- A - JAVA
- B - C
- C - C++
- D - PASCAL

Question 10 : Soit la procédure suivante :

Pour i ← 1 à 10 faire

 A ← 2

 A ← A + i

fin pour
afficher (A)

Quelle est la valeur de A ? :

- A - A = 57
- B - A = 2
- C - A = 12
- D - A = 3

Question 11 : TCP/IP est une architecture réseau de :

- A - OSI : Open Systems Architecture
- B - UIT-T : Union Internationale des Télécommunications
- C - DOD : Department of Defense
- D - EIA : Electronic Industrie Association

Question 12 : Le langage de communication utilisé pour dialoguer avec une base ORACLE est :

- A - My SQL
- B - SQL
- C - JAVA
- D - C++

Question 13 : Parmi les codes suivants, lequel est de longueur variable ? :

- A - ASCII
- B - HUFFMAN
- C - EBCDIC
- D - CCITT n°2

Question 14 : HTTP est :

- A - Un langage de programmation de page Web
- B - un protocole de communication
- C - un site Web
- D - un système d'exploitation

Question 15 : Un Firewall est :

- A - un langage programmation
- B - un système de sécurité
- C - un site Web
- D - une base de données

Question 16 : Qu'est ce que DES ? :

- A - un algorithme de cryptage
- B - un algorithme de signature
- C - un algorithme de hackage
- D - un algorithme de compression

Question 17 : Lequel de ces langages n'est pas rationnel :

- A - $\{ a^{2k} b^{2l} \mid k, l \text{ dans } \mathbb{N} \}$
- B - $\{ a^{2k} b^{3n} \mid n \text{ dans } \mathbb{N} \}$
- C - $(ab)^* ab$
- D - $(a+b)^* ab (a+b)^*$

Question 18 : Dans la transmission asynchrone par caractère, le bit de parité permet :

- A - de synchroniser l'horloge du récepteur
- B - de gérer le tour de parole
- C - de détecter les erreurs de transmission
- D - d'indiquer la fin du caractère

Question 19 : Quel est le support de communication le plus fiable ? (taux d'erreur faible) :

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

Question 20 : Une adresse Internet IP.V4 est codée sur :

- A - 16 bits
- B - 32 bits
- C - 128 bits
- D - 4 bits

Question 21 : Une adresse Internet IP.V6 est codée sur :

- A - 16 bits
- B - 32 bits
- C - 128 bits
- D - 6 bits

Question 22 : Que pouvez-vous dire de $((A \Rightarrow B) \wedge A) \Rightarrow B$:

- A - elle est équivalente à $(\text{non } B) \vee A$
- B - elle est équivalente à $A \wedge B$
- C - c'est une tautologie
- D - elle est équivalente à $A \Rightarrow B$

Question 23 : Parmi les systèmes suivants, lequel ne fait pas partie des systèmes de radiocommunications :

- A - GPRS
- B - TRANSPAC
- C - UMTS
- D - GPS

Question 24 : L'unité de mesure « baud » correspond à :

- A - la valence d'un signal
- B - la rapidité de modulation
- C - un débit binaire
- D - un nombre de bits par symbole

Question 25 : Le chiffrement des données consiste à les :

- A - compresser
- B - rendre illisibles
- C - compter
- D - contrôler

REMARQUE :

Les programmes en Pascal, C et Caml 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 fonctionnelle et 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, il n'y a pas forcément d'optimisation entre les programmes Pascal et C qui peuvent être de simple traduction les uns des autres.

A) RIVIERES NUMERIQUES

Le principe est le suivant :

La première chose que nous avons à faire est d'évaluer le nombre suivant d'une séquence et cette tâche sera réalisée par une fonction particulière (riviere_suivante) qui sera appelée à chaque fois.

Son fonctionnement est simple, on donne un point de départ n , on fixe une variable s à n et on ajoute alors la somme des chiffres de s à n . Ceci est fait par une série de divisions successives de s par 10 (le reste étant ajouté à n) jusqu'à ce que s passe à 0. Il est à noter que l'on utilisera directement le modulo.

Pour répondre à la question portant sur la rencontre d'une rivière n quelconque avec les trois rivières (1, 3 et 9), nous devons stocker les valeurs courantes de ces quatre rivières (les valeurs de départ étant respectivement n , 1, 3 et 9).

Avant chaque tour, nous devons vérifier que la rivière n ne rencontre pas une des trois autres. La méthode retenue est la suivante :

Tant que nous n'avons pas trouvé un point de rencontre :

- Calculer répétitivement les positions suivantes des rivières 1, 3 et 9 jusqu'à ce que nous atteignons ou dépassons n .
- Si nous n'avons pas atteint un point de rencontre, nous passons à la position suivante de la rivière n .

Programme Pascal (TP7)

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

Procedure OuvertureFichier;
Begin
    Assign(FichierEntree, 'RIVIERE.IN');
    Reset(FichierEntree);          (* Ouverture de RIVIERE.IN en
lecture *)
    Assign(FichierSortie, 'RIVIERE.OUT');
    Rewrite(FichierSortie);       (* Ouverture de RIVIERE.OUT en
écriture *)
End;

Procedure FermetureFichier;
Begin
    Close(FichierEntree);
    Close(FichierSortie);
End;

Function riviere_suivante(n: integer): integer;
Var s: integer;
Begin
    s := n;
    while s > 0 do begin
```

```

        n := n + (s mod 10);
        s := s div 10;
    end;
    riviere_suivante := n;
End;

Procedure Resolution (n:integer);
Var riviere1, riviere3, riviere9: integer;
    s: integer;
Begin
    (* Initialisation *)
    riviere1 := 1; riviere3 := 3; riviere9 := 9;
    s:=n;

    (* Boucle de calcul *)
    while (n <> riviere1) and (n <> riviere3) and (n <> riviere9) do
    begin
        while riviere1 < n do riviere1 := riviere_suivante( riviere1 );
        while riviere3 < n do riviere3 := riviere_suivante ( riviere3 );
        while riviere9 < n do riviere9 := riviere_suivante ( riviere9 );
        if (n <> riviere1) and (n <> riviere3) and (n <> riviere9)
            then n := riviere_suivante ( n );
        end;

        (* Emission d'un résultat *)
        if riviere1 = n then writeln( FichierSortie, 'la rivière ', s, '
rencontre en premier la rivière 1 en ', n );
        if riviere3 = n then writeln( FichierSortie, 'la rivière ', s, '
rencontre en premier la rivière 3 en ', n );
        if riviere9 = n then writeln( FichierSortie, 'la rivière ', s, '
rencontre en premier la rivière 9 en ', n );
    End;

Procedure LectureDonnee;
Var i,n: integer;
Begin
    while not(Eof(FichieEntree)) do          (* Lecture du nombre de chaque rivière *)
    begin
        readln( FichierEntree, n );
        Resolution(n)                        (* recherche de rencontre pour la
rivière n *)
    End ;
End;

Begin
    OuvertureFichier;
    LectureDonnee;
    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 */

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

```

```

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

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

void riviere_suivante(int *n)
{
    int s=*n;

    while (s > 0)
    {
        *n += (s % 10);
        s /= 10;
    }
}

void resolution (int n)
{
    int riviere1=1;
    int riviere3=3;
    int riviere9=9;
    int s=n;

    /* Boucle de calcul */
    while ((n != riviere1) && (n != riviere3) && (n != riviere9))
    {
        while (riviere1 < n) riviere_suivante(&riviere1);
        while (riviere3 < n) riviere_suivante (&riviere3);
        while (riviere9 < n) riviere_suivante (&riviere9);
        if ((n != riviere1) && (n != riviere3) && (n != riviere9))
            riviere_suivante(&n);
    }

    /* Emission d'un résultat */
    if (riviere1 = n) fprintf( FichierSortie, `la rivière %d rencontre en
premier la rivière %d en %d`, s,1,n );
    if (riviere3 = n) fprintf( FichierSortie, `la rivière %d rencontre en
premier la rivière %d en %d`, s,3,n );
    if (riviere9 = n) fprintf( FichierSortie, `la rivière %d rencontre en
premier la rivière %d en %d`, s,9,n );
}

```

```

void LectureDonnee()
{
    int n;

    /* Lecture du nombre de chaque rivière */
    while( !feof(FichierEntree))
    {
        fscanf(FichierEntree, "%d", &n );
        resolution(n); /* recherché de rencontre pour cette rivière */
    }
}

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

```

Programme Caml (Objective Caml)

Pour Caml, deux versions des petites fonctions (« chiffres » et « suivant ») sont fournies avant la résolution du problème à proprement parlé.

(* version classique *)

```

let chiffres nb =
  let rec chiffres' c =
    if c <> 0 then
      List.rev ((c mod 10)::(chiffres' (c / 10)))
    else
      []
  in
  if nb = 0 then [0] else chiffres' nb

```

```

let suivant a =
  let rec somme = fonction
    | a::q -> a + somme q
    | [] -> 0
  in
  a + somme (chiffres a)

```

(* version "tail-recursive" *)

```

let tl_chiffres c =
  let rec loop acu d =
    if d <> 0 then
      loop ((d mod 10)::acu) (d/10)
    else
      acu
  in
  loop [] c

```

```

let tl_suitant a =
  a + List.fold_left ( + ) 0 (tl_chiffres a)

```

(* Résolution du problème. *)

```

let rec suivants = fonction
  | (x,a)::q -> (x, (suivant a))::(suivants q)

```

```

| [] -> []

let rec intersection riviere = function
| (nb_initial, nb_courant)::q when nb_courant = riviere ->
  (nb_initial, nb_courant)::(intersection riviere q)
| _::q -> intersection riviere q
| [] -> []

let rec recherche_intersection riviere rivieres =
  match (intersection riviere rivieres) with
  | [] -> recherche_intersection
(suivant riviere) (suivants rivieres)
  | 1 -> 1

let probleme riviere =
  recherche_intersection riviere [(1,1);(3,3);(9,9)]

let resultat entree sorties_du_probleme =
  List.fold_left
  (fun acu (nb, nbc) ->
    "la riviere "^(string_of_int entree)^" rencontre la riviere "
    ^^(string_of_int nb)^" en "^(string_of_int nbc)^".\n"^acu)
  ""
  sorties_du_probleme

let rivieres_du_fichier nom_de_fichier =
  let f = open_in nom_de_fichier in
  let r = ref [] in
  let rec lis_tout () =
    try
      r := (int_of_string (input_line f))::!r;
      lis_tout ()
    with _ -> close_in f; !r
  in
  lis_tout ()

let enregistre_les_resultats nom_de_fichier resultats =
  let f = open_out nom_de_fichier in
  List.iter (fun s -> output_string f (s^"\n")) resultats;
  close_out f

let () =
  let rivieres = rivieres_du_fichier "RIVIERE.IN" in
  let resultats =
    List.map (fun e -> resultat e (probleme e)) rivieres
  in
  enregistre_les_resultats "RIVIERE.OUT" resultats

```

B) COLLIER

Le principe est le suivant :

La façon la plus simple de résoudre ce problème est d'avoir deux variables qui mémorisent la plus longue séquence (cptmax) et la perle de coupe de la plus longue séquence (pdcmax). On initialise alors la plus longue séquence à 0, et l'on détermine pour chaque perle la longueur de la séquence (boucle itérative). A chaque évaluation, on compare la séquence avec la plus longue. Si celle-ci est plus longue, elle devient la nouvelle plus longue séquence et l'on mémorise le numéro de la perle de coupure associée. A la fin, on sauvegarde les valeurs maximales dans le fichier COLLIER.OUT. Pour pouvoir réaliser cela le plus simplement possible, nous allons mettre chaque perle dans un élément de tableau (LectureCollier), ce qui implémenté donne :

Programme Pascal (TP7)

```
Program CollierDePerles;
Const
  N=100;

Type
  T_VectNChar = array[1..N] of char;

Var
  FichierEntree,          (* Fichier TEXT pour l'entrée *)
  FichierSortie : Text;   (* Fichier TEXT pour la sortie *)
  Collier          : T_VectNChar;
  ndp              : Integer; (* Nombre de perles *)

Procedure OuvertureFichier;
Begin
  Assign(FichierEntree, 'COLLIER.IN');
  Reset(FichierEntree);          (* Ouverture de COLLIER.IN en
lecture *)
  Assign(FichierSortie, 'COLLIER.OUT');
  Rewrite(FichierSortie);       (* Ouverture de COLLIER.OUT en
écriture *)
end;

Procedure FermetureFichier;
Begin
  Close(FichierEntree);
  Close(FichierSortie);
End;

Procedure LectureCollier;      (* Lecture des données *)
Var
  s : string;
  i : integer;
begin
  Readln(FichierEntree, s);    (* Récupération des perles *)
  i:=1;
  while(i<=Length(s)) do      (* Mise en forme du collier *)
  begin
    Collier[(i+1) DIV 2]:= copy(s,i,1)[1];
    i:=i+2;
  end;
  ndp:=(i-1) DIV 2;
end;

Procedure DecoupeCollier;     (* Calcul de la plus longue séquence de perles
*)
Var
  pdc, pdcmax, cpt, cptmax : integer;
  vg, vd : integer;
Begin
  cptmax:=0;
  for pdc:=1 to ndp do        (* perle de coupure *)
  begin
    cpt:=2;          (* Il y a au moins 2 perles de comptées *)
    vd:=pdc+1;      (* depart vers la droite du collier *)
    while (Collier[vd mod ndp +1]=Collier[vd]) and (vd<>pdc) do
    begin
      vd:=vd mod ndp+1;
      cpt:=cpt+1;
    end;
  end;
end;
```

```

    vg:=pdc;      (* depart vers la gauche du collier *)
    while (Collier[ndp - ((ndp - vg + 1) mod ndp) ]=Collier[vg]) and
(vg<>pdc+1) do
    begin
        vg:=ndp-((ndp-vg+1)mod ndp);
        cpt:=cpt+1;
    end;
    if cpt>cptmax then
    begin
        cptmax:=cpt;
        pdcmax:=pdc;
    end;
    end;
    writeln(FichierSortie, pdcmax, ' ',pdcmax mod ndp +1, ' ',cptmax);
End;

```

```

Begin
    OuvertureFichier;
    LectureCollier;
    DecoupeCollier;
    FermetureFichier;
End.

```

Programme C (GCC)

```

/* Programme Collier_De_Perles */

#include <stdio.h>
#include <stdlib.h>
#define N 100

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

typedef char[N] T_VectNChar;

T_VectNChar Collier;
int ndp;      /* Nombre de perles */

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

void FermetureFichier()

```

```

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

void LectureCollier()    /* Lecture des données */
{
    char s[255];
    int i=1;

    fscanf(FichierEntree, "%s", &s );    /* Récupération des perles */
    while(i<=strlen(s))                /* Mise en forme du collier */
    {
        Collier[(i+1)/2-1]=s[i-1];
        i+=2;
    }
    ndp=(i-1)/2;
}

void DecoupeCollier();    /* Calcul de la plus longue séquence de perles */
{
    int pdc, pdcmax, cpt, cptmax=0;
    int vg, vd;

    for(pdc=0;pdc<=ndp;pdc++)          /* perle de coupure */
    {
        cpt=2;                          /* Il y a au moins 2 perles de comptées */
        vd=pdc+1;                        /* depart vers la droite du collier */
        while ((Collier[vd % ndp]==Collier[vd-1]) && (vd!=pdc))
        {
            vd=vd % ndp + 1;
            cpt++;
        }
        vg=pdc;                          /* depart vers la gauche du collier */
        while ((Collier[ndp - ((ndp - vg + 1) % ndp)-1]==Collier[vg-1]) &&
(vg!=pdc+1))
        {
            vg=ndp-((ndp-vg+1)%ndp)-1;
            cpt++;
        }
        if (cpt>cptmax)
        {
            cptmax=cpt;
            pdcmax=pdc;
        }
    }
    fprintf(FichierSortie, "%d %d %d\n" ,pdcmax,pdcmax % ndp +1,cptmax);
}

void main()
{
    OuvertureFichier() ;
    LectureCollier() ;
    DecoupeCollier() ;
    FermetureFichier() ;
}

```

Programme Caml (Objective Caml)

```
type couleur = R | B

type collier = couleur list

let denombrement c =
  let rec compte = function
    | (i, (c1, n, j)::q1, c2::q2) when c1 = c2 -> compte (i+1, (c1,
      n+1, j)::q1, q2)
    | (i, l1, c::q) -> compte (i+1, (c, 1, i)::l1, q)
    | (_, acu, []) -> acu
  in
  List.rev (compte (0, [], c))

let couleur (c, _, _) = c

let fusionne (c1, n, i) (c2, m, j) = (c1, n + m, max i j)

let dernier_et_reste =
  let rec loop acu =
    function
      | [] -> failwith "la liste vide n'a pas de dernier element"
      | a::[] -> (a, List.rev acu)
      | a::q -> loop (a::acu) q
  in
  loop []

let fusionne_dernier_et_tete = function
  | [] -> []
  | a::[] -> a::[]
  | tete::q ->
    let (dernier, reste) = dernier_et_reste q in
    if couleur tete = couleur dernier then
      reste@[fusionne tete dernier]
    else
      tete::q

let rec sommes = function
  | (_, n, i)::(c, m, j)::q -> (n + m, j)::(sommes ((c, m, j)::q))
  | _ -> []

let selectionne l =
  let rec calcul reponses l' =
    let (mn, j) = List.hd reponses in
    match l' with
      | (m, i)::q when m > mn -> calcul [(m,i)] q
      | (m, i)::q when m = mn -> calcul ((m,i)::reponses) q
      | a::q -> calcul reponses q
      | [] -> reponses
  in
  calcul [(min_int, -1)] l

let resultat (mn, i) =
  (string_of_int i)^" ^"(string_of_int (i+1))^" ^"(string_of_int mn)

let resultats = List.map resultat

let collier_du_fichier nom_de_fichier =
  let f = open_in nom_de_fichier in
  let r = ref "" in
  let ret = ref [] in
  try
    r := input_line f;
```

```

        if !r = "" then failwith "Fichier invalide !"
        else
            for i = 0 to String.length !r -1 do
if !r.[i] = 'b' then
    ret := B::!ret
else if !r.[i] = 'r' then
    ret := R::!ret
else if !r.[i] <> ' ' then
    failwith "Fichier invalide"
            done;
            List.rev !ret
            with _ -> close_in f; failwith "Erreur pendant la lecture."

let enregistre_les_resultats nom_de_fichier resultats =
    let f = open_out nom_de_fichier in
    let count = ref 1 in
    let affiche r =
        begin
            output_string f ("Solution "^(string_of_int !count)^":\n");
            output_string f (r^"\n\n");
        end
    in
    List.iter affiche (List.rev resultats);
    close_out f;;

let solutions p = selectionne (sommes (fusionne_dernier_et_tete
(denombrement p)))

let () =
    let collier = collier_du_fichier "COLLIER.IN" in
    let resultats = resultats (solutions collier) in
    enregistre_les_resultats "COLLIER.OUT" resultats

```