

EPREUVE OPTIONNELLE d'INFORMATIQUE

CORRIGE

Question 1 : S = 0, A = 0, B = 0

```
pour I = 1 à 2 ;  
    pour J = 1 à 5  
        A = A + J ;  
    fin ;  
B = B + I ;  
fin ;  
afficher S = A + B
```

Quelle est la valeur de S ?

- A - 15
- B - 20
- C - 33
- D - 52

Question 2 : S = 0

```
pour N = 1 à 5  
    S = S + (N * (N + 1)) / 2  
fin ;  
afficher S
```

Que fait ce programme ?

- A - calcule la somme des 5 premiers nombres
- B - calcule la division des 5 premiers nombres
- C - calcule la multiplication des 5 premiers nombres
- D - calcule la somme des 5 premiers nombres entiers

Question 3 : Indiquez l'ordre d'évaluation de l'expression suivante en langage C : a-b-c-d :

- A - a-(b-c-d)
- B - (a-b)-(c-d)
- C - (a-b-c)-d
- D - (((a-b)-c)-d)

Question 4 : Dans un ordinateur, le micro-processeur communique avec les périphériques à l'aide de :

- A - bus de données
- B - bus de commandes
- C - bus d'adresses
- D - sans bus

Question 5 : La norme RS232C de l'EIA, est une interface de communication entre :

- A - deux ordinateurs
- B - un ordinateur et un modem
- C - deux modems
- D - un ordinateur et un téléphone

Question 6 : Un bit de parité permet de :

- A - compter le nombre de bits à '1'
- B - compter le nombre de bits à '0'
- C - compresser les données
- D - détecter des erreurs de transport

Question 7 : Pour compresser des données, vous utiliserez quel codage ?

- A - CCITT n°2 (Baudot)
- B - CCITT n°5 (ASCII)
- C - EBCDIC
- D - HUFFMAN

- Question 8 : Un FIREWALL est :
- A - un système de messagerie
 - B - un système de sécurité
 - C - un langage de programmation
 - D - un système d'exploitation
- Question 9 : Parmi ces outils, quel est celui qui ne fait pas partie des technologies Web ?
- A - ASP
 - B - HTML
 - C - JAVA
 - D - MERISE
- Question 10 : Un ERP est :
- A - un langage de programmation
 - B - un portail d'application
 - C - un système de gestion intégré
 - D - un protocole de communication
- Question 11 : Un WIRELESS est :
- A - un réseau de fibre optique
 - B - un réseau de firewall
 - C - un réseau sans fil
 - D - un service de sécurité
- Question 12 : La biométrie est une technologie de contrôle d'accès par :
- A - saisie de mot de passe
 - B - analyse des empreintes digitales
 - C - analyse vocale
 - D - présentation de pièces d'identité
- Question 13 : Parmi les systèmes suivants quel est celui qui ne fait pas partie des réseaux de radio-communication :
- A - GPRS
 - B - UMTS
 - C - WIN-NT
 - D - WAP
- Question 14 : TCP/IP est :
- A - une architecture de réseau
 - B - un système de contrôle de processus industriel
 - C - un serveur de fichier
 - D - un système d'exploitation
- Question 15 : Pour coder la valeur décimale 2003, il faut combien d'octets ?
- A - un
 - B - deux
 - C - trois
 - D - quatre
- Question 16 : Une variable booléenne peut prendre comme valeur :
- A - égal
 - B - vrai
 - C - supérieur
 - D - faux
- Question 17 : Internet explorer est :
- A - un langage de programmation
 - B - un navigateur
 - C - un protocole de contrôle des erreurs
 - D - une base de données

Question 18 : Soit 3 variables : B (booléenne), N (entier) égal à 10 et C (entier) initialisé à 0. Une des instructions suivantes est invalide, laquelle ?

- A - $C = C + N$
- B - $N = N * 2$
- C - $B = B * 2$
- D - si B est faux alors $C = 0$

Question 19 : « Je conduis mon véhicule. J'aperçois un feu tricolore. Si le feu est rouge alors je freine sinon je poursuis ma route ». Quel type de structure représente la suite d'actions précédente ?

- A - une structure composée
- B - une structure alternative
- C - une structure mixte
- D - une structure répétitive

Question 20 : LIS N

```
vecteur TAB(N) LIS TAB
en I range 1
en somme range 0
tant que I <= N répète
    $21 en somme range somme + TAB(I)
    en I range I + 1
: 21 $
édition résultat
```

Que fait ce programme ?

- A - compare la variable I avec la variable N
- B - permute les éléments d'un tableau
- C - recherche le plus grand nombre d'un tableau
- D - calcule la somme des éléments d'un tableau

Question 21 : let abs X = if X >= 0 then X
else - X ; ;

Que calcule l'expression précédente ?

- A - met X à 1
- B - affecte la valeur 0 à X
- C - compare X et abs
- D - la valeur absolue de X

Question 22 : ≠ let j = 20 ; ;
≠ 30 - j ; ;
≠ let K = 3 * J + 27 ; ;

Quelle est la valeur de K ?

- A - 87
- B - 57
- C - 47
- D - 37

Question 23 : ≠ let som-vect X1 Y1 X2 Y2 = (X1 + X2) , (Y1 + Y2) ; ;

Que fait ce programme ?

- A - compare les variables X et Y
- B - cherche la variable X la plus petite
- C - cherche la variable Y la plus petite
- D - calcule la somme de 2 vecteurs à 2 dimensions

Question 24 : ≠ let som-vect V1 V2 = (fst V1) + (fst V2),
(snd V1) + (snd V2) ; ;

Que fait ce programme ?

- A - calcule la somme de 2 vecteurs à 2 dimensions
- B - compare V1 et V2
- C - cherche la variable X la plus grande
- C - cherche la variable Y la plus grande

Question 25 : CAML est un langage:

- A - objet
- B - fonctionnel
- C - à fort typage
- C - à dominante itérative

REMARQUE :

Les programmes en Pascal, 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 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) A) PARN

Le principe est le suivant :

Le problème peut être interprété de la manière suivante : n objets sont disponibles (les clubs), un objet i ayant un poids a_i . Chaque objet est disponible en nombre illimité. Nous cherchons alors à déterminer les x_i , nombre d'exemplaires de chaque objet, permettant d'atteindre, sans le dépasser, un poids p (la distance à couvrir). Cette version générale est appelée *sac à dos en nombres entiers*. Dans notre cas, la recherche porte sur une solution exacte (valeur égale au poids p).

La solution est obtenu par système d'empilement. On commence par trier en ordre décroissant les valeurs lues des différents clubs. On associe au vecteur trié un vecteur correspondant au nombre de chaque club utilisé pour atteindre la somme exacte. Ce vecteur est géré comme une pile, c'est à dire que tant qu'une solution n'est pas trouvé, on remonte vers les clubs de plus fort poids, on en enlève un puis on recommence les empilements sur les valeurs inférieures. Si l'on dépèle le dernier club de plus fort poids, c'est qu'il n'existe pas de solution à notre problème.

Programme Pascal (Delphi mode console)

```
Program Golf;  
{ $APPTYPE CONSOLE }  
  
uses  
  SysUtils;  
  
const  
  NbMaxClub = 24;  
  
type  
  T_Vect_NbMaxClub_Ent = array [1..NbMaxClub] of integer;  
  
var
```

```

FichierEntree,                                (* Fichier TEXT pour l'entrée *)
FichierSortie  : Text;                        (* Fichier TEXT pour la sortie *)
NbClubLu      : integer;
ListeClub,
CoupParClub   : T_Vect_NbMaxClub_Ent; (* Pile de coups *)
DistanceTotale : integer;

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

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

procedure LectureDistanceEtClubs;            (* Lecture des données *)
Var
    i : integer;
begin
    Readln(FichierEntree, DistanceTotale);  (* Nbre de cadeaux à partager *)
    Readln(FichierEntree, NbClubLu);        (* Nbre de cadeaux à partager *)
    For i:=1 to NbClubLu do
        begin
            Readln(FichierEntree, ListeClub[i]); (* lecture des cadeaux *)
            CoupParClub[i]:=0;                  (* Initialisation du nombre de
coups *)
        end;
    end;                                     (* par club *)
end;

procedure TriClubs;    (* Tri décroissant des valeurs de cadeaux *)
Var
    i, temp : integer;
    nontrie : boolean;
begin
    nontrie:=true;
    while nontrie do
        begin
            nontrie:=false;
            for i:=1 to NbClubLu-1 do
                if ListeClub[i]<ListeClub[i+1] then
                    begin
                        temp:=ListeClub[i];
                        ListeClub[i]:=ListeClub[i+1];
                        ListeClub[i+1]:=temp;
                        nontrie:=true;
                    end;
            end;
        end;
end;

procedure RechercheClubs; (* heuristique du sac à dos objets en nombre
illimité*)
var
    i, somme,
    Nbcoups : integer;
    nontrouve : boolean;
begin

```

```

    somme:=0;
    Nbcoups:=0;
    nontrouve:=true;
    i:=1;
    while nontrouve and (i>0) do    (* boucle principale de recherche de somme
*)
    begin
        if somme+ListeClub[i]<=DistanceTotale then    (* si club utilisable, on
prend *)
        begin
            Nbcoups:=Nbcoups+1;
            CoupParClub[i]:=CoupParClub[i]+1;
            somme:=somme+ListeClub[i];
        end
        else i:=i+1;

        if somme = DistanceTotale    (* si trouve, on sort *)
        then nontrouve:=false
        else if i>NbClubLu then    (* sinon, si i>limite, on remonte de
niveau *)
        begin
            i:=NbClubLu;
            Somme:=Somme-(ListeClub[i]*CoupParClub[i]);
            Nbcoups:=Nbcoups-CoupParClub[i];
            CoupParClub[i]:=0;
            while (i>0) and (CoupParClub[i]=0) do
            begin
                i:=i-1;
            end;
            if i>0 then
            begin
                Somme:=Somme-ListeClub[i];
                CoupParClub[i]:=CoupParClub[i]-1;
                Nbcoups:=Nbcoups-1;
                i:=i+1;
            end;
        end;
    end;
    if not nontrouve then
    begin
        writeln(FichierSortie,Nbcoups,' coups');
        for i:=1 to NbClubLu do
            if CoupParClub[i]>0 then writeln(FichierSortie,CoupParClub[i],' x
',ListeClub[i]);
        end
        else writeln(FichierSortie,' Objectif impossible à atteindre ');
    end;

begin
    OuvertureFichier;
    LectureDistanceEtClubs;
    TriClubs;
    RechercheClubs;
    FermetureFichier;
end.

```

Programme C (DevC++ 4)

```

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

#define NBMAXCLUB 24

FILE    *FichierEntree;    /* Fichier TEXT pour l'entrée */

```

```

FILE      *FichierSortie;   /* Fichier TEXT pour la sortie */

int  NbClubLu, DistanceTotale;
int  ListeClub[NBMAXCLUB+1], CoupParClub[NBMAXCLUB+1];   /* Pile de coups
*/

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

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

void LectureDistanceEtClubs()   /* Lecture des données */
{
    int i;

    fscanf(FichierEntree,"%d",&DistanceTotale);   /* Nbre de cadeaux à
partager */
    fscanf(FichierEntree,"%d",&NbClubLu);           /* Nbre de cadeaux à
partager */
    for(i=1;i<=NbClubLu;i++)
        {
            fscanf(FichierEntree,"%d",&ListeClub[i]);   /* lecture des cadeaux */
            CoupParClub[i]=0;                               /* Initialisation du nombre de
coups */
        }
    /* par club */
}

void TriClubs()   /* Tri décroissant des valeurs de cadeaux */
{
    int i, temp, nontrie;

    nontrie=1;
    while (nontrie)
        {
            nontrie=0;
            for(i=1;i<NbClubLu;i++)
                if (ListeClub[i]<ListeClub[i+1])

```

```

        {
            temp=ListeClub[i];
            ListeClub[i]=ListeClub[i+1];
            ListeClub[i+1]=temp;
            nontrive=1;
        }
    }
}

void RechercheClubs() /* heuristique du sac à dos objets en nombre
illimité*/
{
    int i, somme, Nbcoups, nontrive;

    somme=Nbcoups=0;
    nontrive=i=1;
    while ((nontrive) && (i>0))
    {
        /* boucle principale de recherche de somme */
        if ((somme+ListeClub[i])<=DistanceTotale) /* si le club est
utilisable, on le prend */
        {
            Nbcoups=Nbcoups+1;
            CoupParClub[i]=CoupParClub[i]+1;
            somme=somme+ListeClub[i];
        }
        else i++;

        if (somme==DistanceTotale) /* si trouve, on sort */
            nontrive=0;
        else
            if (i>NbClubLu) /* sinon, si l'on est en limite, on remonte de
niveau */
            {
                i=NbClubLu;
                somme=somme-(ListeClub[i]*CoupParClub[i]);
                Nbcoups=Nbcoups-CoupParClub[i];
                CoupParClub[i]=0;
                while ((i>0) && (CoupParClub[i]==0)) i--;
                if (i>0)
                {
                    somme=somme-ListeClub[i];
                    CoupParClub[i]=CoupParClub[i]-1;
                    Nbcoups=Nbcoups-1;
                    i++;
                }
            }
    }

    if (!nontrive)
    {
        fprintf(FichierSortie, "%d coups\n", Nbcoups);
        for(i=1;i<=NbClubLu;i++)
            if (CoupParClub[i]>0) fprintf(FichierSortie,"%d x
%d\n",CoupParClub[i],ListeClub[i]);
    }
    else fprintf(FichierSortie,"Objectif impossible à atteindre\n");
}

int main(int argc, char *argv[])
{
    OuvertureFichier();
    LectureDistanceEtClubs();
    TriClubs();
}

```

```

RechercheClubs();
FermetureFichier();
return 0;
}

```

B) PARTAGE EQUITABLE

Le principe est le suivant :

Le problème est le même que pour l'exercice précédent, à deux différences près, les objets ne sont disponibles qu'en un seul et unique exemplaire et nous cherchons à approcher au mieux une somme, non à l'égaliser. Cela se ramène à un système d'empilement simple.

Là aussi, on commence par trier en ordre décroissant les valeurs lues des différents cadeaux. On applique ensuite un algorithme (de type glouton) qui consiste à tester l'opportunité d'utiliser chaque cadeau dans la somme de ceux-ci. Si le poids actuel augmenté de celui du nouveau cadeau ne dépasse pas la somme que l'on cherche à atteindre, on continue.

Notre problème portant sur un partage équitable, la somme à atteindre est initialisée à la moitié de celle de tous les cadeaux. Une fois cette valeur approchée au mieux, on obtient l'autre part en soustrayant la valeur atteinte à la somme globale.

Programme Pascal (Delphi mode console)

```

Program partage;
{$APPTYPE CONSOLE}

uses
  SysUtils;

const
  NbMaxCadeau = 100;

type
  T_Vect_NbMaxCadeau_Ent = array [1..NbMaxCadeau] of integer;

var
  FichierEntree,           (* Fichier TEXT pour l'entrée *)
  FichierSortie : Text;   (* Fichier TEXT pour la sortie *)
  NbCadeauLu      : integer;
  ListeCadeau     : T_Vect_NbMaxCadeau_Ent;
  ValeurTotale,
  ValeurMoyenne   : integer;

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

end;

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

procedure LectureCadeaux;      (* Lecture des données *)

```

```

Var
  i : integer;

begin
  Readln(FichierEntree,NbCadeauLu);          (* Nbre de cadeaux à partager *)
  ValeurTotale:=0;
  For i:=1 to NbCadeauLu do
  begin
    Readln(FichierEntree,ListeCadeau[i]);    (* lecture des cadeaux *)
    inc(ValeurTotale,ListeCadeau[i]);
  end;
  ValeurMoyenne:=ValeurTotale DIV 2;
end;

procedure TriCadeaux;      (* Tri décroissant des valeurs de cadeaux *)
Var
  i,temp : integer;
  nontrie : boolean;
begin
  nontrie:=true;
  while nontrie do
  begin
    nontrie:=false;
    for i:=1 to NbCadeauLu-1 do
      if ListeCadeau[i]<ListeCadeau[i+1] then
      begin
        temp:=ListeCadeau[i];
        ListeCadeau[i]:=ListeCadeau[i+1];
        ListeCadeau[i+1]:=temp;
        nontrie:=true;
      end;
    end;
  end;
end;

procedure RepartitionCadeaux;  (* heuristique du sac à dos 0-1 *)
var
  i, somme : integer;

begin
  somme:=0;
  i:=1;
  for i:=1 to NbCadeauLu do
    if somme+ListeCadeau[i]<= ValeurMoyenne
      then somme:=somme+ListeCadeau[i];
  writeln(FichierSortie,somme);
  writeln(FichierSortie,ValeurTotale-somme);
end;

begin
  OuvertureFichier;
  LectureCadeaux;
  TriCadeaux;
  RepartitionCadeaux;
  FermetureFichier;
end.

```

Programme C (CC)

```

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

#define NBMAXCADEAU 24

FILE      *FichierEntree;  /* Fichier TEXT pour l'entrée */

```

```

FILE      *FichierSortie;    /* Fichier TEXT pour la sortie */

int  NbCadeauLu,  ValeurTotale, ValeurMoyenne;
int  ListeCadeau[NBMAXCADEAU+1];

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

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

void LectureCadeaux()      /* Lecture des données */
{
    int i;

    fscanf(FichierEntree, "%D", &NbCadeauLu);    /* Nbre de cadeaux à partager */
    ValeurTotale=0;
    for(i=1; i<=NbCadeauLu; i++)
        {
            fscanf(FichierEntree, "%d", &ListeCadeau[i]);    /* lecture des cadeaux */
            ValeurTotale=ValeurTotale+ListeCadeau[i];
        }
    ValeurMoyenne=ValeurTotale/2;
}

void TriCadeaux()      /* Tri décroissant des valeurs de cadeaux */
{
    int i, temp, nontrie;

    nontrie=1;
    while (nontrie)
        {
            nontrie=0;
            for(i=1; i<NbCadeauLu; i++)
                if (ListeCadeau[i]<ListeCadeau[i+1])
                    {

```

```

        temp=ListeCadeau[i];
        ListeCadeau[i]=ListeCadeau[i+1];
        ListeCadeau[i+1]=temp;
        nontrie=1;
    }
}

void RepartitionCadeaux() /* heuristique du sac à dos */
{
    int i, somme;

    somme=0;
    i=1;
    for(i=1;i<=NbCadeauLu;i++)
        if ((somme+ListeCadeau[i])<= ValeurMoyenne) somme=somme+ListeCadeau[i];
    fprintf(FichierSortie,"%d\n",somme);
    fprintf(FichierSortie,"%d\n",ValeurTotale-somme);
}

int main(int argc, char *argv[])
{
    OuvertureFichier();
    LectureCadeaux();
    TriCadeaux();
    RepartitionCadeaux();
    FermetureFichier();

    return 0;
}

```