

Exercice 1 :

Soit un fichier d'étudiants 'Section.dat'. Chaque étudiant est défini par un **Matricule** entier, **Nom** et **Prénom** chaînes de 20 caractères, un **Sexe** (M : masculin et F : Féminin) et une **Moyenne** réelle.

- 1- Donner la déclaration du Type **TEtudiant** et du fichier correspondant.
- 2- Ecrire une action paramétrée **Delib** permettant de créer deux fichiers 'Garcons.dat' et 'Filles.dat' contenant respectivement les étudiants admis de sexe masculin et de sexe féminin.

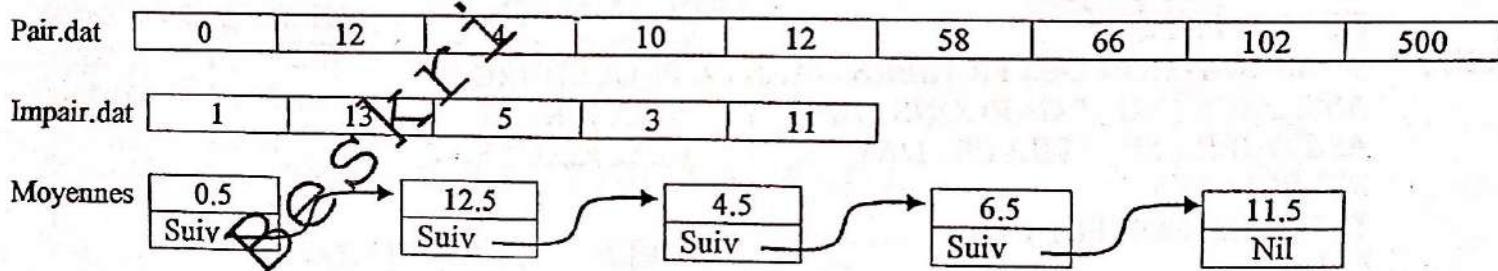
COPIE

Exercice 2 :

Soient deux fichiers d'entiers 'Pair.dat' et 'Impair.dat', on veut créer une liste chaînée 'Moyenne' où chaque élément représente la moyenne des valeurs ayant la même position dans les deux fichiers.

- 1- Donner la déclaration de la liste moyenne.
- 2- Ecrire une action paramétrée **CreerL** permettant de créer la liste 'Moyenne'.
- 3- Ecrire une action paramétrée **SupprimeL** supprimant les moyennes inférieures à une valeur **X** dans cette liste.

SCHI

Exemple :**Exercice 3 : Répondre sur la feuille.**

Bonne Chance

EXERCICE 1 :

```

TYPE TETUDIANT = ENREGISTREMENT
| MATRICULE : ENTIER ;
| NOM , PRENOM : CHAINE [ 20 ] ;
| SEXE : CARACTERE ;
| MOYENNE : REEL ;
| FIN ;
FETUD = FICHIER DE TETUDIANT ;
VAR FS , FG , FF : FETUD ;

DEBUT ASSIGNER ( FS , " SECTION . DAT " );
ASSIGNER ( FG , " GARCONS . DAT " );
ASSIGNER ( FF , " FILLES . DAT " );

PROCEDURE DELIB ( E / FS : FETUD ; E-S / FG , FF : FETUD ) : { Solution 1 }
VAR ENR : TETUDIANT ;
DEBUT { ASSIGNATION DES FICHIERS DANS L'ALGORITHME APPELANT }
RELIRE ( FS ); REECRIRE ( FG ); REECRIRE ( FF );
TANTQUE NON FDF ( FS )
FAIRE
    LIRE ( FS , ENR );
    SI ENR . MOYENNE ≥ 10
        ALORS CAS ENR . SEXE VAUT
            | 'M' : ECRIRE ( FG , ENR );
            | 'F' : ECRIRE ( FF , ENR );
            FCAS ;
    FSI ;
FAIT ;
FERMER ( FG ); FERMER ( FF );
FIN ;

PROCEDURE DELIB ( E / FS : FETUD ) ; { Solution 2 }
VAR ENR : TETUDIANT ;
FG , FF : FETUD ;
DEBUT { ASSIGNATION DES FICHIERS DANS LA PROCEDURE }
ASSIGNER ( FG , " GARCONS . DAT " ); REECRIRE ( FG );
ASSIGNER ( FF , " FILLES . DAT " ); REECRIRE ( FF );
RELIRE ( FS );
TANTQUE NON FDF ( FS )
FAIRE
    LIRE ( FS , ENR );
    AVEC ENR FAIRE
        SI MOYENNE ≥ 10
            ALORS SI SEXE = 'M'
                ALORS ECRIRE ( FG , ENR )
                SINON ECRIRE ( FF , ENR );
            FSI ;
    FSI ;
FAIT ;
FERMER ( FG ); FERMER ( FF );
FIN ;

```

EXERCICE 2 :

TYPE PTR = ^ CELLULE;
 CELLULE = ENREGISTREMENT
 | MOY : REEL ;
 | SVT : PTR ;
 FIN ;

VAR TFE = FICHIER D'ENTIER ;
 MOYENNE : PTR ; FP , FI : TFE ;

DEBUT ASSIGNER (FP , "PAIR.DAT");
 ASSIGNER (FI , "IMPAIR.DAT");

FONCTION CREERL (E / FP , FI : TFE) : PTR ;
VAR T , P , Q : PTR ; E1 , E2 : ENTIER ;
DEBUT T \leftarrow NIL ;
 RELIRE (FP) ; RELIRE (FI) ;
 TANTQUE NON FDF (FP) ET NON FDF (FI)
 FAIRE
 | LIRE (FP , E1) ; LIRE (FI , E2) ;
 | ALLOUER (P) ;
 | P ^ . MOY \leftarrow (E1 + E2) / 2 ;
 | P ^ . SVT \leftarrow NIL ;
 | SI T = NIL
 | | ALORS T \leftarrow P
 | | SINON Q ^ . SVT \leftarrow P ;
 | | FSI ;
 | | Q \leftarrow P ;
 FAIT ;
 CREERL \leftarrow T ;
 FIN ;

{ Solution 1 }

Copie

Soushill

X

FONCTION CREERL (E / FP , FI : TFE) : PTR ;
VAR T , P : PTR ; E1 , E2 : ENTIER ;
DEBUT T \leftarrow NIL ;
 RELIRE (FP) ; RELIRE (FI) ;
 TANTQUE NON FDF (FP) ET NON FDF (FI)
 FAIRE
 | LIRE (FP , E1) ; LIRE (FI , E2) ;
 | SI T = NIL
 | | ALORS ALLOUER (P) ; T \leftarrow P
 | | SINON ALLOUER (P ^ . SVT) ;
 | | P \leftarrow P ^ . SVT ;
 | | FSI ;
 | | P ^ . MOY \leftarrow (E1 + E2) / 2 ;
 FAIT ;
 SI T \neq NIL
 | ALORS P ^ . SVT \leftarrow NIL ;
 FSI ;
 CREERL \leftarrow T ;
 FIN ;

{ Solution 2 }

PROCEDURE SUPPRIMEL(E-STL.TIN, L, X);

VAR PR, P : PTR;

DEBUT

 PR ← NIL;

 P ← L;

 TANTQUE P ≠ NIL

 FAIRE

 SI P ^ . MOY < X

 ALORS

 SI

 PR = NIL

 ALORS L ← P ^ . SVT ;

 LIBERER (P);

 P ← L

 SINON PR ^ . SVT ← P ^ . SV

 LIBERER (P);

 P ← PR ^ . SVT ;

 FSI

 PR ← P ;

 P ← P ^ . SVT ;

 FSI ;

 FAIT ;

FIN ;



EXERCICE 3 :

```
#include <stdio.h>
#include <stdlib.h>
void donnerEntree()
{ int choix;
printf("Entrer:\n");
printf("1: pour Afficher 'Rouge'\n");
printf("2: pour Afficher 'Bleu'\n");
printf("3: pour Afficher 'Vert'\n");
printf("0: pour Quitter le Programme ... \n");
scanf("%d", &choix);
return choix;
}
int main()
{
char ent;
while (0) ;
{
ent = donnerentree()
if (ent == 0)
break;
}
switch(ent)
{
case 1
printf("Rouge \n");
case 2:
printf("Bleu \n");
break;
case 3:
printf("Vert \n");
break;
default:
printf("Erreur Mauvais choix ... \n");
}
system("pause");
return 0;
}
```

```
int donnerEntree()
{
scanf("%d", &choix);

int ent;
while (1) { En plus, à enlever ' ; ' }

ent = donnerEntree();
if (ent == 0)

{ En plus, à enlever }

case 1:
break;

printf("Erreur Mauvais choix ... \n");
```

Épreuve Finale - Semestre 02

Exercice 01 : (06pts)

Algorithme EX01;

Type

elt=Enregistrement

MO : chaîne; d, F : entier;

Fin;

Var

PH : chaîne; MOT : tableau [10] de elt;
N, i, j : entier;

Début

Lire(PH); i←0; j←-1;

Tant que (i<longueur(PH))

Faire

si (PH[i] ≠ '') alors j←j+1; avec MOT[i]

Faire d←i; mo←"";

Tant que (i<longueur(PH)) et (PH[i] ≠ '')

Faire

mo←mo+PH[i]; i←i+1;

Fait;

sinon i←i+1;

X 501

Fsl;

Fait;

Pour i←0 à j

Faire

avec MOT[i]

Faire

écrire(mo, d, f);

Fait;

Fait;

Fin.

Copie

Barème Ex01 :

Déclarations	1
initialisation	0.5
Calcul de D et F	2
Construction du mat	1
Construction du vecteur MOT	1
Affichage du résultat	0.5

Exercice 02 : (07pts)

Procédure EPURER();

Var

Dem, A, R : Fichier de Demandeur; D: demandeur; B : bénéficiaire;
Ben : Fichier de bénéficiaire; T : Booléen ;

Début

assigner (Dem, "demandeurs.dat"); lire(Dem);
assigner (Ben, "bénéficiaires.dat"); lire(Ben);
assigner (A, "A.dat"); écrire(A);
assigner (R, "R.dat"); écrire(R);

Tant que non fdf(Dem)

Faire

Prendre (Dem, D);

Flire (Ben); T ← faux;

Tant que non fdf (Ben) et non T

Faire

Prendre (Ben, B);

Si (B.num_idf=D.num_idf) alors T ← vrai;

Fsi;

Fait;

si T alors mettre (R, D);

sinon mettre (A, D);

Fsi;

Fait;

Fermer(Dem); Fermer(A); Fermer(R);

Flire(A); Flire(R); Fécrire(Dem);

Tant que non fdf(A)

Faire

Prendre (A, D);

Mettre (Dem, D);

Fait;

Tant que non fdf(R)

Faire

Prendre (R, D);

Mettre (Ben, D);

Fait;

Fermer(Dem); Fermer(R); Fermer(A); supprimer(R); supprimer(A); Fermer(Ben);

Fin;

Barème Exo2-1 :

Déclarations	0.5
Construction de la partie acceptée	1
Construction de la partie rejetée	0.5

:Concaténation des deux parties	0.5
Manipulation des fichiers (suppression)	0.5

2-

Procédure MAJ (E/num_dos : entier; E/salaire : réel; E/ nbr_enfant : entier);
 Var

Dem, TMP : Fichier de Demandeur; D, D1:demandeur; B : bénéficiaire;
 Ben : Fichier de bénéficiaire;
 Fonction Comparer_dossier (S1, S2 : réel; N1, N2 : entier) :booléen;

Début

C←faux;

si [(S1>S2) ou ((S1=S2) et (N1≥N2))] alors

Comparer_dossier←vrai;

sinon

Comparer_dossier←faux.

Fsi;

Fin;

Début

Assigner (Dem,"demandeurs.dat"); flire(Dem);

Assigner (Ben,"bénéficiaires.dat"); flire(Ben);

Assigner (TMP,"TMP.dat"); sécrire(TMP);

T←faux;

Tant que non fdi(Dem) et non T

Faire

Prendre(Dem, D);

si (num_dos=D.num_dos)

alors

D1.num_dos←num_dos; D1.nom←D.nom; D1.situation←D.situation;

D1.num_idf←D.num_idf;

D1.prénom←D.prénom; D1.salaire←salaire;

D1.nbr_enfant←nbr_enfant;

B sinon si (D.salaire>salaire) ou ((D.salaire=salaire) et (D.nbr_enfant≥nbr_enfant))

//comparer_dossier (salaire, D.salaire, nbr_enfant, D.nbr_enfant)

alors

mettre(TMP, D)

sinon

flire(Ben);

Tant que non fdi(Ben) et non T

Faire

Prendre(Ben, B);

Si (B.num_idf=D1.num_idf) alors T←vrai;

Fsi;

Fait;

si T alors

écrire("dossier rejeté");

sinon

mettre(TMP, D1);

Tant que non fdi(Dem)

Faire
 Fait;
 Prendre(Dem, D);
 mettre(TMP, D)
 Fsi;
 Fsi;
 Fait;
 Fermer(Ben); Fermer(TMP);
 si non T alors lire(TMP); écrire(Ben); Tant que non fdi(Tmp)
 Faire
 Prendre(TMP, D);
 mettre(Ben, D)
 Fait;
 Fermer(Ben); Fermer(TMP);
 Fsi;
 Supprimer (TMP);
 Fin;

COPIE

Barème Exo2-2 :

Déclarations	0.5
Comparaison des salaires et nbr_enfant	0.5
Suppression et sauvegarde du dossier	0.5
Recherche de l'emplacement	1
Vérification du dossier	1
Ajout du dossier	0.5

Exercice 03 : (07pts)

Type

Ptr1=^Enregistrement
 val , seq : entier;
 Fin;
 PtrF=^eltF;
 eltF=Enregistrement
 info : Ptr1;
 svt : PtrF;
 Fin;
 Ptr=^elt;
 elt=Enregistrement
 Info : entier;
 svt : Ptr;
 Fin;

Procédure FREQUENCE(E/ L :Ptr; S/LF : PtrF);

Var

P, Q : ptrF; R : ptr;

Début

alloquer(P^1 , svt); $P^1.svt \leftarrow$ nil;
alloquer($P^1.info$); $R^1 \leftarrow P^1.info$;
 $R^1.val \leftarrow L^1.info$; $R^1.freq \leftarrow 1$;

Fst;

$L \leftarrow L^1.svt$;

Fait;

sinon $LF \leftarrow$ nil;

Fst;

Fin;

Barème Exo3 :

Déclarations des types + en-tête	1
Parcours de la liste L	1
Recherche de l'élément de L dans LF	1
Création de la tête	1
Création du corps	1.5
L'optimisation	1.5

Corrigé de l'examen du module Algorithmique 02 et structures de données pour la section 11

Exercice 1 : 7pts

1) Définition du type AMI :

Constantes

Max. = 50;

Type

AHI := tableau[Max, Max] entier;

Remarque :

- Toute déclaration du AMI en tant qu'une matrice carrée d'entiers est acceptée.
- La matrice Ami est une matrice symétrique donc tous les éléments de la diagonale sont égaux à 1 et de ce fait si l'est ami avec j on peut déduire aussi que j est ami avec i (l'inverse est aussi correct).

2) La procédure amisEnCommun : 2pts

procédure amisEnCommun(E/ A : AMI; E/ N : entier, E/
i : entier, E/ j : entier)

début

var

k : entier ;

```
pour k ← 1 à N faire
    si(A[1, k] = 1 et A[j, k] = 1) alors
        écrire(k);
    fin;
```

fait;

fin.

Remarque : avoir AMI en tant que variable globale est aussi accepté.

3) Déclaration du type listeAmis : 3pts

Type

```
listeAmis = noeud;
noeud = enregistrement;
val : entier;
svt : listeAmis;
finenregistrement.
```

a) fonction créerListeAmis : 2pts

```
fonction créerListeAmis(E/ A : AMI, E/ N : entier,
E/ i : entier) : listeAmis
début
    var
        L : listeAmis;
        k : entier;
```

var

L : listeAmis;

k : entier;

```
ajoutFin(L, i) ;
pour k ← 1 à N faire
    si (A[i,k] = 1) alors
        ajoutFin(L, k)
    fin;
    fait;
    retourner L;
fin.
```

5) La fonction nbAmis : 2pts

```
fonction nbAmis(E/ L : listeAmis) : entier
début
    var
        p : listeAmis ;
        cpt : entier ;
        p ← L ; cpt ← 0 ;
        tantque( p <> Nil) alors
            cpt ← cpt + 1 ;
            p ← ^p.svt ;
        fait ;
    retourner cpt-1 ;
fin.
```

Exercice 2 : 7 pts

1) La fonction estMersenne :

```
fonction estMersenne(E/ n : entier) : booléen
début
    var
        p : entier ;
        p ← log2(N - 1) + 1 ;
    retourner estPremier(p) ;
fin
```

Rémarque :

- Les autres solutions justes sont acceptables.

2) La procédure fichierMersenne :

```
Procédure fichierMersenne(E/ nomEx1 : chaîne, E/
nomEx2 : chaîne)
```

X Var
F1, F2 : Fichier entier;
x : entier;
Ouvrir(F1, nomEx1, "lecture");
Ouvrir(F2, nomEx2, "écriture");
tantque(non fdf(F1)) faire
 lire(F1, x);
 si (estMersenne(x) = vrai) alors

écrire(F2, x);

fin;

fait;

fermer(F1);

fermer(F2);

fin.

✓ T C

Exercice 3 : bpts

Voir le fichier descripteur.c

Examen Final

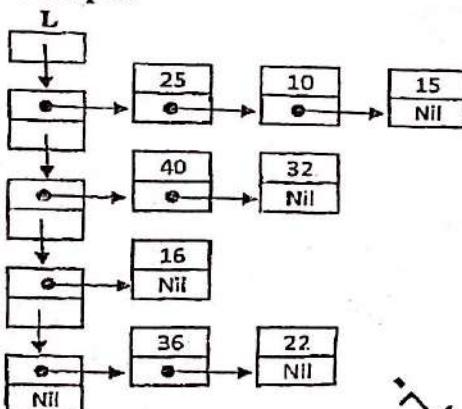
Durée : 01H30.

Exercice 1 : (10 Pts)

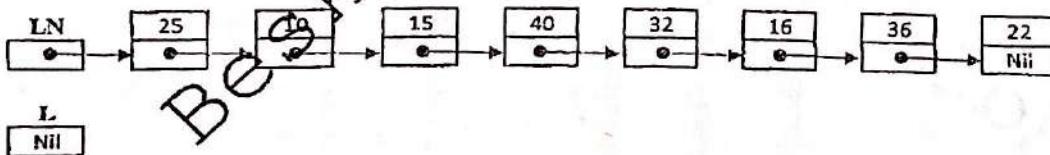
Soit L une liste chainée dont les éléments sont des pointeurs vers des listes de nombres entiers positifs.

- 1) Ecrire une action paramétrée Lineaire permettant de transformer (sans allocation) la liste L en une liste linéaire LN (voir exemple).
- 2) Ecrire une action paramétrée Nombre1 permettant de donner le nombre de 1 utilisé dans la représentation binaire d'un entier positif.
- 3) En utilisant l'action paramétrée Nombre1, écrire une action paramétrée permettant de créer une liste LB contenant pour chaque élément de LN , le nombre de 1 résultant de la conversion binaire.
- 4) Ecrire une action paramétrée MaxP permettant de déterminer la première occurrence de la valeur maximale et sa position dans une liste linéaire.
- 5) En utilisant l'action paramétrée MaxP, déterminer le max de LN et de LB et leurs positions respectives, puis conclure ?

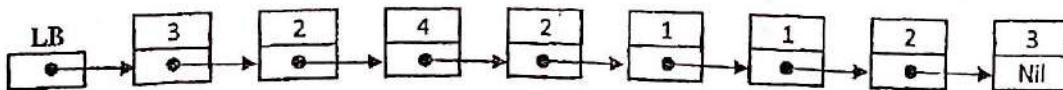
Exemple:



Après transformation :



Valeur	25	10	15	40	32	16	36	22
En binaire	11001	1010	1111	101000	100000	10000	100100	10110
Nombre 1	3	2	4	2	1	1	2	3



Exercice 2 : (10 Pts)

Une entreprise dispose de deux catégories de clients. Soient deux fichiers existants ayant pour noms physiques CliCat1.Dat et CliCat2.Dat, représentant respectivement les clients de la catégorie 1 et les clients de la catégorie 2, identifiés respectivement par F1 et F2. Les deux fichiers F1 et F2 ont la même structure (NomCL et ChiffreAF) où NomCL est le nom du client et ChiffreAF est le chiffre d'affaires du client exprimé en DA. On veut accorder des réductions à certains clients en fonction de leurs chiffres d'affaires.

Questions :

- 1) Ecrire une action paramétrée MOYCA permettant de calculer la moyenne des chiffres d'affaires d'une catégorie de clients;
- 2) On veut regrouper tous les clients ayant un chiffre d'affaires supérieur ou égal à la moyenne de sa catégorie dans un fichier F3 nommé SelMoy.Dat; Ecrire une action paramétrée REGROUPEZ permettant de réaliser ce regroupement;
- 3) Ecrire une action paramétrée MAXCA permettant de déterminer le client ayant le chiffre d'affaires le plus élevé;
- 4) A partir du fichier F3, on veut sélectionner et classer par ordre de mérite les dix (10) clients ayant un chiffre d'affaires supérieur ou égal à la moyenne des deux moyennes des deux catégories. Ecrire un algorithme utilisant les actions paramétrées définies précédemment pour afficher la liste des clients bénéficiaires des réductions.

NB: ne pas utiliser de structures de données autres que les fichiers.

Souhly

Bon Courage

Correction Examen Final S2 : 2016

Exercice 1 : (10P)

Type : PlateauListe,
 ListeV-Eregistrement
 Val entier ; Sauv : Plateau;
 Fin.
 Vliseur : ListeV;
 ListeV-Eregistrement
 Haute : Plateau ; Vliseur : Vlisee.
 Fin.

1. Procedure LireEntier(LS/L,Vlisee : S/ LN : Plateau);
 Var P,Q : Plateau , LV : Vlisee.
 Debut

LN<=Nd ;
 Si L<=Nd Alors
 LN <=LAutre , LN <=LVautre .
 Liberer(L) ; L <=LY ;
 Si LN>Nil Alors
 Q<=LN ;
 Tantque Q<=Q "Suiv">Nil
 Faire Q<=Q "Suiv" ; Fai .
 Fai ;
 Tantque L<=Nil Faire
 P<=L ; Lautre , LN <=L "Vsliv" ;
 Liberer(L) ; L <=LY ;
 Si LN>Nil Alors
 Q "Suiv" <=P ;
 Tantque Q<=Q "Suiv">Nil
 Faire Q<=Q "Suiv" ; Fai .
 Si LN>Nil Alors
 Q <=LN ;
 Tantque Q<=Q "Suiv">Nil
 Faire Q<=Q "Suiv" ; Fai .
 Fai .

Fai .

Fai .

2- Fonction Nombre(X : entier) entree ;
 Var NB : entier ;
 Debut

NB<=0 ,
 Tantque X>=0 Faire
 Si X MOD 2 = 1 Alors NB <=NB+1 Fai ;
 X <=X DIV 2 .
 Fai ,
 Nombre1 <=NB ;
 Fin :

3. Procedure LBio(E/LN : Plateau ; S/ LB : Plateau) :
 Var P,Q : Plateau .
 Debut
 LB<=Nil ;
 Si LN>Nil
 Alors
 Allouer(LB).
 LB^,Val<=NombreI(LN^,Val) ;
 Q<=LB ; LN <=LN^"Suiv" ;
 Tantque LN>Nil Faire
 Allouer(P) .
 P^,Val<=NombreI(LN^,Val) ;
 Q<=Suiv<=P ; Q<=P .
 LN <=LN^"Suiv" ;
 Fai ;
 Q "Suiv" <=Nil ;
 Fai .

4. Procedure MaxP(E/ L : Plateau ; S/ Max,Pos : entier) ;
 Var I : entier ;
 Debut

Max<=0 ,Pos<=0 ,I<=0 ;

Tantque L<=Nil Faire

I<=I+1 .

Si L^,Val>Max Alors Max<=L^,Val ; Pos<=I Fai ;
 I <=L^,Sauv ;

Fai .

I <=I-1 .

5.

Algorithm MaxPos .

Type

Var MaxN,MaxB,PosN,PosB : entier ;

Procedure MaxP(E/ L : Plateau ; S/ Max,Pos : entier) ;

Debut

MaxP(LN,MaxN,PosN) ;

MaxP(LN,MaxB,PosB) ;

Ecrire("Max LN","MaxN","Pos",PosN) .

Ecrire("Max LB","MaxB","Pos",PosB) ;

Fai .

Conclusion :

PosN n'est pas forcément égale à PosB, d'où un nombre de l'est max ne veut pas dire que le nombre correspondant est aussi max.

Exercice 2 (10 pts)

Type Client-Empreinte

Nom Cl chaine[5]; Client recet.

Fini.

FC : Fichier de Client.

Fonction MOYCA(F1) recet

Var S recet; N entier; C: Client;

Début

S=0; N=1; Recette1;

Tant que Non F1=N et N<=10 faire

 Liref1.Cli; N=N+1; S=S+Cl.Chiffre1;

Fin

Si N>0 Alors MOYCA=S/N Sinon MOYCA=0 Fini;

Fermement;

Fini

Procédure REGROUPE(F1,F2,F3,F4,F5, FC);

Var Moy recet; Cl Client;

Début

Moy=MOYCA(F1); Recette1; Recette2=0;

Tant que Non F1=N et N<=10 faire

 Liref1.Cli;

 Si Cl.Chiffre1 < Moy Alors Recette1=F1.F1;

 Fini; Fermement 1;

 Moy=MOYCA(F2); Recette2;

 Tant que Non F2=N et N<=10 faire

 Liref2.Cli;

 Si Cl.Chiffre1 < Moy Alors Recette2=F2.F1;

 Fini; Fermement 2; Fermement 1;

Fin

Procédure MAXCLI(S,F1,FC,N,CMax,Client)

Var Cl Client;

Début

CMax=N; Client=Cl; Client.F1=S; Client.F2=0;

N=N+1; Recette1;

Alors Liref1.Cli; Client.F1=;

Tant que Non F1=N et N<=S faire

 Liref1.Cli;

 Si Cl.Chiffre1 < Client.F1 Alors Client.F1=;

 Alors Client.F2=;

 Fin;

Fin

Fermement;

Fin;

4.

Algorithmie Selection

Type

Var F1,F2,F3,F4; Client; Moy recet; I entier;

Fonction MOYCA(...);

Procédure REGROUPE(...);

Procédure MAXCLI(...);

Début

 Attribut1.'Chiffre1 Dat'); Attribut2.'Chiffre2 Dat');

 Attribut3.'Chiffre3 Dat'); Attribut4.'Imer');

 REGROUPE(F1,F2,F3),

 Moy=MOYCA(F1),MOYCA(F2);

 Recette3),I+1,

 Tant que Non F3=F3 et I<=10 faire

 MAXCLI(F3,CMax),

 Si CMax.Chiffre1 < Moy

 Alors Client.I=Max.Nom.Client+Recette3;

I+1,I;

 Supprimer ce élément;

 Recette3); Recette3;

 Tant que Non I>F3 faire

 Liref3.Cli;

 Si CMax.Chiffre1 Alors Erreur(F1,Cli,F3);

I+1;

 Lement3; Lement3;

 Copier F3.datas3;

 Recette3), Recette3;

 Tant que Non I>F3 faire

 Liref3.Cli; Lement3.Cli

 Fin

 Fermement3; Fermement 3; Recette3;

 N=I+1;

 Fin

 I+1;

 Recette3 de la liste'; Fermement3;

Fin

 I+1;

 I+1;

Épreuve Finale - Semestre 02

Exercice 01 (07 pts) :

Soit *F_nom* un fichier de chaînes de caractères contenant les noms des étudiants de la première année MI.

Écrire une action paramétrée *Transformer* qui transforme une chaîne de caractères fournie en entrée, en une autre chaîne en supprimant toutes les voyelles qu'elle contient.

En utilisant l'action paramétrée *Transformer*, écrire un algorithme qui construit un fichier *F_nom_trans* contenant les noms transformés du fichier *F_nom* (déjà rempli).

Exemple : 'mohammed' après transformation devient 'mhmmnd'

Exercice 02 (07 pts) :

Dans ce qui suit, on veut étendre la procédure *lire* définie en cours appliquée sur un fichier de caractères, en ajoutant un troisième paramètre qui représente le nombre de caractères qu'on peut lire en une seule opération (K), en suivant la syntaxe :

lire_bloc(<fichier_logique>, <idf_chaine>, <nombre_car>); tels que *<fichier_logique>*, *<idf_chaine>*, et *<nombre_car>* représentent respectivement, le nom logique du fichier, le nom de la variable tampon(intermédiaire) qui est de type chaîne de caractères, et le nombre de caractères qu'on peut lire à la fois.

1-En utilisant la procédure *lire* classique (à deux paramètres), écrire la procédure *lire_bloc()*.

2-On suppose maintenant que le nombre de caractères K=3, en utilisant la procédure *lire_bloc()*, écrire une action paramétrée *recherche* qui retourne la position de la première occurrence d'un caractère donnée C s'il existe, et zéro(0) sinon.

Exemple

Soit *Fcar* un fichier de caractères



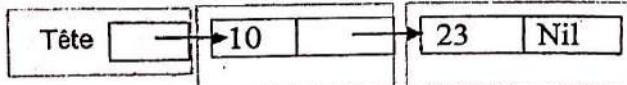
Nombre (K)	1 ^{ère} lecture	2 ^{ème} lecture
3	'Exa'	'men'
4	'Exam'	'en'
>=6	'Examen'	

Pour la recherche, K=3, si C='e', la recherche retourne la position (5), si C='A', la recherche retourne la position (0).

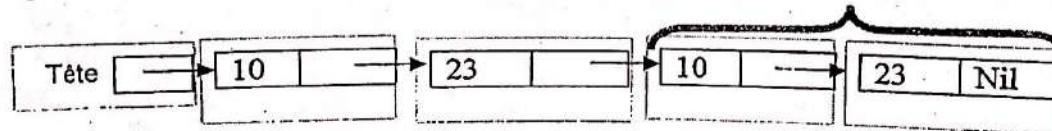
Exercice 03 : (06 pts)

Écrire une action paramétrée qui duplique le contenu d'une liste entière (ajoute le même contenu à la fin).

Exemple



après duplication, on obtient la liste :



Bon courage

Correction Epreuve Finale

Exercice 01

Fonction transformer_mot (mot:chaîne);

Var I,T :entier ;

Début

T:=taille(mot); i:=1;Transformer:=";

Tant que (i<=T)

Faire

cas mot[i] vaut

'a','e','i','o','u','y','A','E','I','O','U',Y' : i:=i+1

sinon Transformer:= Transformer+ mot[i];
i:=i+1;

Fcas;

Fait;

Fin;

Algorithme TranFile;

var

F,G :fichier de chaîne ;

Cnom :chaine;

Trouve :booleen ;

Fonction transformer_mot (mot:chaîne);

Exercice 02

Procédure lire_bloc(f: fichier de caractère;
bloc:chaîne ; k:entier) ;

Var c:car ;

Début

bloc:="";

Tant que non fdf(F) et (k>0)

Faire

lire(F, c); bloc:=bloc+c; k :=k-1 ;

Fait ;

Fin;

Fonction Recherche (E-S/f: fichier de caractère;

c:caractère):Entier;

Var i,t,b:entier ;

Début

Exercice 03

Type

ptr=[^]elt;

elt=Enregistrement

info:entier;

svt:ptr;

Fin;

Procédure dupliquer(T:ptr);

Var

P,Q,L,D : Ptr;

Début

Si T \neq nil

Alors allouer(L);

L[^].info:=T[^].info;

Q:=L;

Début

assigner(F, 'f_nom'); relire(F) ;

trouve :=faux ;

assigner(G,'f_nom_tran.dat'); réécrire(Gs) ;

Tant que non fdf(F)

Faire

lire(F, Cnom);

Cnom :=Transformer(Cnom)

Si Cnom \neq "

Alors Si Non Trouve

Alors assigner(G,'f_nom_trans');

réécrire(fG_trans) ;

Trouve :=Vrai ;

Fsi

écrire(G, Cnom);

Fsi ;

Fait ;

Fermer(F);

Si Trouve Alors Fermer(G) ; Fsi ;

Fin.

Recherche:=0; b:=0 ;

Tant que non fdf(f) et (Recherche=0)

Faire

lire_bloc(f, bloc, 3) ; i:=1 ; t:=taille(bloc)

Tant que (i<=T) et ((Recherche=0)

Faire

si (bloc[i]=c)

alors Recherche:=3*b+i

sinon i:=i+1

Fsi;

Fait;

b:=b+1;

Fait;

Fin;

Tant que (T[^].svt \neq nil)

Faire

T:=T[^].svt;

allouer(D);

D[^].info:=T[^].info;

Q[^].svt:=D;

Q:=D ;

Fait;

Q[^].svt:=nil;

T[^].svt:=L;

Fsi ;

Fin;

Révision :EPREUVE FINALE (2^e Semestre)- 2010/11Exercice n°1 : (7 pts)

Soient les types suivants :

Type Date = Enregistrement Jour, Mois, Année : Entier ; FinEnreg ;
 Code = Enregistrement Année : 2003 .. 2070 ; Num : 0 .. 9999 ; Spécialité : 1 .. 20 ;
 FinEnreg ;
 Etudiant = Enregistrement

Matricule : Cod. ; Nom, Prénom : Chaîne[30] ; Date_Nais : Date ;
 Note : tableau[1..12] de réel ;
 Coef : tableau[1..12] de Entier ;
 MoyG : Réel ;

FinEnreg ;

Soit T(N) un vecteur d'étudiants d'une section L1-MI (N<=140).

1° Déclarer le type du vecteur T, puis tracer la hiérarchie sémantique d'un étudiant T[i].

2° Ecrire une procédure qui calcule la moyenne générale MoyG de chaque étudiant T[i] comme suit :

$$T[i].MoyG = \left(\sum_{j=1}^{12} T[i].Note[j] * T[i].Coef[j] \right) / \left(\sum_{j=1}^{12} T[i].Coef[j] \right)$$

3° Ecrire une procédure qui réalise :

- L'appel de la procédure précédente ;
- Affiche les champs : Matricule, Nom, Prénom et Date_Nais des étudiants dont la moyenne générale est maximale.

Exercice n°2 : (8 pts)

Soit 'FTrié.Do1' le nom physique d'un fichier non vide de nombres entiers triés dans l'ordre croissant. Ecrire l'algorithme qui permet de :

- Ouvrir le fichier existant 'FTrié.Do1' avec comme nom logique F.
- Insérer une valeur entière Val dans F telle que F restera trié après insertion de Val.
- Afficher F après insertion.

Exemple :

Etat initial : Val = 9

Etat Final : Val = 9

F	3	6	12	18	51	?
---	---	---	----	----	----	---

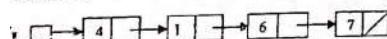
F	3	6	9	12	18	51	?
---	---	---	---	----	----	----	---

Exercice n°3 : (5 pts)

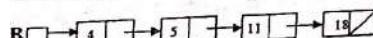
Soit L une liste de nombres entiers. Ecrire une procédure Codage qui code la liste L, selon un parcours "gauche → droite", en une liste R comme suit :

$$U1(R) = U1(L); \quad U2(R) = U1(R) + U2(L); \quad U3(R) = U2(R) + U3(L) \quad \dots \quad Un(R) = Un-1(R) + Un(L).$$

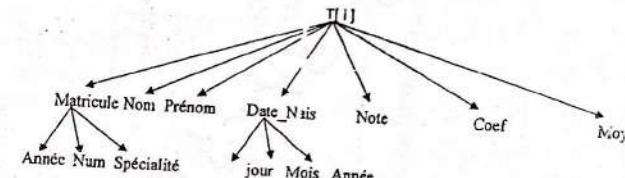
Où : Uj : désigne le j-ième élément d'une liste ; et n désigne le nombre d'éléments d'une liste.

Exemple :

On obtient R :

CORRIGE DE L'EPREUVE FINALE (2^e Semestre)Exercice n°1 : (7 pts)

1° Type vect = tableau [1.. 140] de Etudiant ;



2° Procédure P(E_S / T : Vect ; E/N : Entier) ;
 Var i, j, S : Entier ;
 Début

Pour i := 1 à N
 Faire T[i].MoyG := 0; S := 0;
 Pour j := 1 à 12 Faire T[i].MoyG := T[i].MoyG + T[i].Note[j] * T[i].Coef[j];
 S := S + T[i].Coef[j];
 Fait;
 T[i].MoyG := T[i].MoyG / S;

Fin;

3° Procédure Affiche(E_S / T : Vect ; E/N : Entier) ;
 Var i : Entier; Max : Réel ;
 Procédure P(E_S / T : Vect ; E/N : Entier) ; IDEM Fin;

Début
 P(T, N); Max := T[1].MoyG;
 Pour i := 2 à N Faire Si T[i].MoyG > Max Alors Max := T[i].MoyG; Si; Fait;
 Ecrire(" Etudiant de moyenne générale max= ", Max, " sont : ");
 Pour i := 1 à N Faire
 Si (T[i].MoyG = Max) Alors Ecrire(T[i], Date_Nais) Faire
 Ecrire(" Matricule : ", Matricule, Année, Matricule.Num, Matricule.Spec);
 Ecrire(" Nom et Prénom : ", Nom, " ", Prénom);
 Ecrire(" Né(e) le : ", jour, "/", Mois, "/", Année); Fait;
 Fsi;
 Fait;

Fin;

Exercice n°2 : (8 pts)

Algorithme Exo2 ;
 Var F G : Fichier de Entier ; V, Val : Entier ; Inséré : Booléen ;
 Début
 As F := ' >1' ; Relire(F) ; Assigner(G, 'Tempo') ; Récrire(G) ;
 Ecrire('Donner Val : ') ; Lire(Val) ; Inséré := Faux ;
 Tant que Non Fdf(F) Et (inséré=Faux) Faire Lire(F, V) ;
 Si V <= Val Alors Ecrire(G, V) ;
 Sinon Ecrire(G, Val) ; Ecrire(G, V) ; inséré := Vrai ;
 Fsi ;
 Fait ;
 /* (inséré = Faux) Alors Ecrire(G, Val) /* insertion en fin du fichier */
 Sinon Tant que Non fdf(F) Faire Lire(F, V) ; Ecrire(G, V) ; Fait ;
 Fsi ;
 /* recopier G dans F */ Relire(G) ; Récrire(F) ;
 Tant que Non fdf(G) Faire Lire(G, v) ; Ecrire(F, V) ; Fait ; Effacer(G) ;
 Ecrire(' F après insertion de ', Val, ' est : ') ; Relire(F) ;
 Tant que Non fdf(F) Faire Lire(F, V) ; Ecrire(V, ' ') ; Fait ; Fermer(F) ;
 Fin.

Exercice n°3 : (5 pts)

Type Liste = Tcellule ; cellule=Enregistrement info : Entier ; Suivant : Liste ; FinEnreg ;
 Procédure Codage(E/L : Liste ; S/R : Liste) ;
 Var p,q,t : Liste ;
 Début
 t:=Nil Alors R:=Nil
 Sinon t:=L ; Allouer(p) ; p.info:=t.info ; R:=t ; t:=t.suivant ;
 tant que (t > Nil)
 Faire Allouer(q) ; q.info:=p.info + t.info ;
 P.t.suivant:=q ; p:=q ;
 t:=t.suivant ;
 Fait ;
 P.t.suivant:=Nil ;
 Fsi ;
 Fin ;

Epreuve de Rattrapage (2^eSemestre)-2010/11Exercice N°1 : (8 pts)

Soient les types suivants :
 Type Date = Enregistrement Jour, Mois, Année : Entier ; FinEnreg ;
 Personne = Enregistrement Nom , Prénom : Chaîne[30] ;
 Date_Nais : Date ; FinEnreg ;
 Livre = Enregistrement
 Titre : Chaîne[120] ; Auteur : Personne ;
 Année_Pub : 1900 .. 2070 ; Prix : Réel ; Code , NbrPage : Entier ;
 FinEnreg ;

Soit F un Fichier non vide de type "Livre".
 1°_Déclarer le type du fichier F, puis tracer la hiérarchie sémantique d'un élément V de F.
 2°_Ecrire une procédure qui, à partir de F, Construit (calcule) un fichier G de type "Personne" contenant les auteurs des livres dont :

- > Le nom de l'auteur Commence par la lettre 'B' ;
- > L'année de naissance de l'auteur dépasse l'an 1950 ;
- > Le prix du livre est inférieur à 300 DA ;
- > L'année de publication du livre "Année_Pub" dépasse 2010 ;
- > Et le nombre de pages du livre "NbrPage" ne dépasse pas 200.

Exercice n°2 : (7 pts)

Soit F un Fichier non vide de nombres entiers ; et soient Val1 et Val2 deux valeurs entières données.
 Ecrire une procédure qui insère Val2 après chaque occurrence de Val1 dans le fichier F.

Exemple :
 État initial : Val1 = 4 et Val2 = 7 État Final : Val1 = 4 et Val2 = 7
 F [4 | 9 | 4 | 5 | 1 | 3 | 4 | 7] F [4 | 7 | 9 | 4 | 7 | 5 | 1 | 3 | 4 | 7]

Exercice n°3 : (5 pts)
 Soit L une liste de nombres entiers strictement positifs. Ecrire une procédure qui, à partir de la liste L, construit une liste R selon un parcours "droite-gauche" contenant les éléments de L qui sont multiples de 5.

Exemple :
 L [] → [20] → [8] → [12] → [25] → [30] → []
 On obtient R :
 R [] → [30] → [25] → [20] → []

ÉPREUVE FINALE - SEMESTRE 2

(DURÉE : 1h 30mn)

Exercice n°1 : (7 points)

Soient les types suivants :

Type : Personne = Enregistrement Nom, Prénom : Chaîne [25];
 Activité : Chaîne ; Nationalité : Chaîne [45] ; FinEnreg ;

Ouvrage = Enregistrement Titre : Chaîne ; Auteur : Personne ;
 Année : Entier ; FinEnreg ;

Soit T(N) un tableau d'ouvrages ($N \leq 100$), et soit V(M) un tableau de personnes ($M \leq 200$). Ecrire une procédure qui, pour chaque personne (Nom et Prénom) du tableau V qui est aussi auteur d'un ouvrage du tableau T, affecte le champ "Activité" de cette personne de V et le champ "Activité" de cet auteur de T par la constante chaîne 'ÉCRIVAIN'.

Exercice n°2 : (8 points)

Type : E = Enregistrement Mot : chaîne [30] Long : entier ; FinEnreg ;

Soit F un fichier de type E, écrire l'algorithme qui :

1. Saisit uniquement le champ "Mot" de chaque élément de F (arrêt dès lecture d'un mot vide) ;

Exemple de l'étape 1 :

F	Hakim	Ali	Amel	Houda	Karim	Adel	
			X				

2. Calcule, pour chaque élément de F (si F est non vide), le champ "Long" qui représente la longueur du "Mot" correspondant (sans utiliser une structure de données intermédiaire : tableau, liste ou fichier).

Exemple de l'étape 2 :

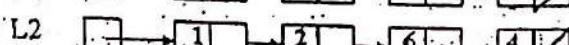
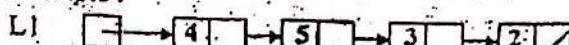
F	Hakim	Ali	Amel	Houda	Karim	Adel	
	5	3	4	5	5	4	

3. Affiche l'état final du fichier F.

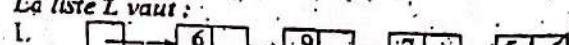
Exercice n°3 : (5 points)

Soient L1 et L2 deux listes non vides de nombres entiers et ayant le même nombre d'éléments. Ecrire la procédure qui construit une liste L, selon un parcours "droite -> gauche", dont un élément vaut la somme d'un élément de L1 et d'un élément de L2 qui sont de même rang (position).

Exemple :



La liste L vaut :



CORRIGÉ DE L'ÉPREUVE FINALE (SEMESTRE 2)

Exercice n°1 : (7 points)

Soient les types suivants :

Type Vect = tableau [1 .. 100] de ouvrage;
Tab = tableau [1 .. 200] de personne;

Procédure P (E_S / T : Vect ; E_S / V : Tab ; E / N, M : Entier);

Var i, j : entier;

Début

Pour i := 1 à M

Faire Pour j := 1 à N

Faire Si (V[i].Nom = T[j].Auteur.Nom) ET

(V[i].Prénom = T[j].Auteur.Prénom)

Alors

V[i].Activité := 'ECRIVAIN';

T[j].Auteur.Activité := 'ECRIVAIN';

Fsi;

Fait;

Fait;

Fin.

Exercice n°2 : (8 points)

Algorithm Exo2 ;

type E = Enregistrement mot : Chaîne[30]; long : entier; FinEnreg;

var F : fichier de E;

V, W : E; i, k : entier;

Début

Assigner (F, 'essai.emp'); Récrire(F);

Ecrire(' donner un mot du fichier F : '); Lire(V.mot);

Si (V.mot = '') Alors Ecrire(' Pas de calcule: F vide ')

Sinon Tant que (V.mot <> '')

Faire Ecrire(F; V);

Ecrire(' donner un mot du fichier F : '); Lire(V.mot); Fait ;

Relire(F); k := 0;

Tant que non fdf(F)

Faire Lire(F, V); k := k + 1; V.long := longueur(V.mot);

Relire(F); Pour i := 1 à (k - 1) Faire Lire(F, W); Fait ;

Ecrire(F, V);

Fait ;

Ecrire(' F devient : '); Relire(F);

Tant que non fdf(F) Faire Lire(F, V); Ecrire(V.mot, ' ', V.long); Fait ;

Fsi;

Fermier (F);

Fin.

Exercice n°3 : (5 points)

Type Ptent = ↑ élément ;
élément = Enregistrement
 info : Entier ;
 Suivant : Ptent ;
FinEnreg ;

Procédure SomListes(E/ L1, L2 : Ptent; S/ L : Ptent);
Var p, q, k : Ptent;
Début
 P:=L1; q:=L2; L:=Nil;
 Tant que (p<> Nil) /* ou bien . Tantque (p<>Nil) Et(q<>Nil) */
 Faire Allouer(k);
 k↑.info := p↑.info + q↑.info ;
 k↑.suivant := L;
 L:=k;
 P:=p↑.suivant; q:=q↑.suivant;
 Fait;
Fin;

Scanné avec CamScanner

EPREUVE FINALE
(Semestre 2)

Exercice 1 : (7 pts)

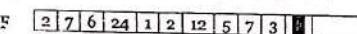
1. Soit Mot une chaîne de caractères. Ecrire une fonction qui vérifie si Mot est un palindrome (2 pts).
2. Soit F un fichier non vide de nombres entiers. Ecrire une fonction qui vérifie si F est trié dans l'ordre croissant (2,5 pts).
3. Soit M une chaîne de caractères (30 caractères au maximum). Ecrire une procédure qui construit un enregistrement R composé du mot M, la longueur de M et la première lettre de M (2,5 pts).

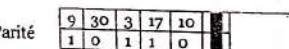
Exercice 2 : (8 pts)

Soit F un fichier d'entiers strictement positifs. Ecrire un algorithme qui :

1. Lit les éléments de F (arrêt de lecture dès lecture d'un nombre ≤ 0).
2. Si le nombre d'éléments du fichier F est pair, construit le fichier d'enregistrements Parité dont un élément représente la somme des éléments de F pris deux à deux et la parité de cette somme (pair=0 ; impair=1).
3. Supprime les éléments pairs du fichier Parité.

Exemple :

1. F 

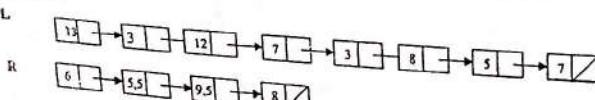
2. Parité 

3. Parité 

Exercice 3 : (5 pts)

Soit L une liste de nombres entiers non vide dont le nombre d'éléments est pair. Ecrire une action paramétrée qui, à partir de L, construit selon un parcours "droite-gauche" une liste R dont les éléments représentent la moyenne des éléments de L pris deux à deux.

Exemple :



CORRIGE DE L'EPREUVE FINALE
(Semestre 2)

Exercice 1 : (7 pts)

1°/ Type Ch = Chaîne[30] ;

Fonction Palind (E / Mot : Ch) : Booléen ;

Var B : Booléen ; lg, i : Entier ;

Début

B := Vrai ; lg := Longueur(Mot) ; i := 1 ;

Tant que (i <= lg div 2) Et (B = Vrai)

Faire Si (Mot[i] <> Mot[lg-i+1]) Alors B := Faux

Sinon i := i+1 ;

Fsi ;

Fait ;

Palind := B ;

Fin ;

2°/ Type Fich = Fichier de Entier ;

Fonction Vérif(E_S / F : Fich) : Booléen ;

Var v, w : entier ; B : booleen ;

Début

Relire (F) ; Lire(F, v) ; B := vrai ;

Tant que Non Fdf(F) Et (B = vrai)

Faire Lire(F, w) ;

Si (v > w) alors B := Faux

Sinon v := w ;

Fsi ;

Fait ;

Vérif := B ;

3°/ Type ch = chaîne[30] ;

E = Enregistrement

Mot : ch ; N : entier ; FinEnreg ;

Pcar : caractère ; Procédure P(E / M : ch ; S / R : E) ;

Début

R.Mot := M ; R.N := longueur(M) ;

Si (R.N = 0) Alors R.Pcar := "

Sinon R.Pcar := M[1] ;

Fsi ;

Fin ;

Exercice 2 : (8 pts)

Alg: nom: Exo2;
 Type E = Enregistrement X:Entier; ' : o..1; FinEnreg;
 Var F : Fichier de Entier;
 ' ; Parité : Fichier de E;
 ' ; nbr : Entier; w : E;
 Début
 Assigner(F, 'F.doi'); Récrire(F); Lire(v);
 Si v <= 0 Alors Ecrire(' Pas de calcul : F vide')
 Sinon nbr := 0;
 Tant que (v>0)
 Faire Ecrire(F, v); nbr := nbr+1;
 Lire (v);
 Fait;
 Si (nbr mod 2 = 1)
 Alors Ecrire('pas de calcul du fichier Parité : nbr elts de F impair')
 Sinon Assigner(Parité, 'Parité.doi'); Récrire(Parité);
 Relire(F);
 Ecrire('Elts du fichier Parité sont : ') ;
 Tant que Non fdf(F) Faire Lire(F, v); Lire(F, z);
 w.x := v + z;
 w.P := (w.x mod 2); Ecrire(Parité, w);
 Ecrire(w.x, ' ', w.P);
 Fait;
 /* Suppression des elts pairs */
 Relire(Parité); Assigner(G, 'tempo.doi'); Récrire(G);
 Tant que Non fdf(Parité)
 Faire Lire(Parité, w);
 Si (w.x = 1) alors Ecrire(G, w); Fsi;
 Fait;
 Relire(G); Récrire(Parité);
 Tant que Non fdf(G)
 Faire lire(G, w); Ecrire(Parité, w); Fait;
 Effacer(G);
 Ecrire('Elts de parité après suppression sont : ') ;
 Relire(Parité);
 Tant que Non fdf(Parité)
 Faire Lire(Parité, w); Ecrire(w.x, ' ', w.P); Fait;
 Fermer(F); fermer(Parité);
 Fsi;
 "si;
 Fin.

Exercice 3 : (5 pts)

Type Ptr1 = ↑ Elément;
 Elément = Enregistrement info:Entier; suivant:Ptr1; FinEnreg;
 Ptr2 = ↑ Cellule;
 Cellule = Enregistrement info:Entier; suivant:Ptr2; FinEnreg;
 Procédure P(E/ L:Ptr1; S/ R:Ptr2);
 Var p, q:Ptr1; t:Ptr2;
 Début
 R:=Nil; P:=L;
 Tant que (P<> Nil)
 Faire q := p↑.suivant; allouer(t);
 t↑.info := (p↑.info + q↑.info) / 2;
 t↑.suivant := R;
 R := t;
 P := q↑.suivant;
 Fait;
 Fin;

ÉPREUVE FINALE – SEMESTRE 2
(Durée : 1h 30mn)

Exercice n° 1 : (10 points)

Suivent les types suivants.

Type Personne - Enregistrement Nom, Prénom : chaîne[20];
Année naissance : 1900 .. 2050;
Adresse : chaîne [50]; FinEnreg;

Type Employé - Enregistrement Identité : Personne;
Fonction : Chaîne[100];
NbreEnfant : 0 .. 20;
Salaire : Réel; FinEnreg;

Soit T un vecteur de N ($N \leq 100$) employés.

- Ecrire une procédure qui ajoute une prime de 5000 DA à tous ceux des employés dont le salaire ne dépasse pas 10.000 DA et dont le nombre d'enfants est supérieur à 2. (5 pts)
- Etant donnée une année courante Année_C, écrire une procédure qui affiche les informations (Nom, prénom et adresse) des employés dont l'âge est égal à 60 ans. (5 pts)

Exercice n° 2 : (6 points)

Soit F un fichier d'entiers strictement positifs.

- Etant donné le fichier F, écrire une procédure Inverse qui crée le fichier G contenant les éléments de F dans l'ordre inverse. G est le fichier inverse de F.

Exemple :

F: 11 20 33 14 45 51 ...

Le fichier G inverse de F est: G 45 14 33 20 11 ...

- En utilisant la procédure précédente, écrire l'algorithme principal qui:
 - > Lit les éléments du fichier F (arrêt de lecture dès la saisie d'un nombre ≤ 0).
 - > Affiche son fichier inverse G.

Exercice n° 3 : (4 points)

Ecrire une procédure qui calcule la valeur maximale des éléments d'une liste L d'entiers strictement positifs ainsi que sa position (son rang dans la liste).

Si la liste L est vide alors la procédure retourne zéro pour la valeur maximale et zéro pour sa position.

**CORRECTION DE
L'ÉPREUVE FINALE - SEMESTRE 2**

Exercice n° 1 : (10 points)

Soit le type suivant :

Type Vect=tableau [1.. 100] de Employé ;

1/

Procédure Prime(E-S/ T : Vect ; E/ N : Entier) ;

Var i : Entier ;

Début

Pour i := 1 à N

Faire Avec T[i] Faire

SI (Salaire <= 10 000) Et (NbreEnfant > 2)

Alors Salaire := Salaire + 5000

Fsi ;

Fait ;

Fait ;

Fin ;

2/

Soit le type suivant :

Type Année = 1900 .. 2050

Procédure AffichInfos(E/ Année_C : Année ; E/ T : Vect ; E/ N : Entier) ;

Var i : Entier ;

Début

Ecrire('Employé âgés de 60 ans : ')

Pour i := 1 à N

Faire Avec T[i].Identité

Faire SI (Année_C - Année_naissance = 60)

Alors écrire('Nom : ', Nom)

écrire('Pénom : ', Prénom)

écrire('Adresse : ', Adresse)

Fsi ;

Fait ;

Fait ;

Fin ;

Algorithm inverseFile (v pour v)

Algorithme inverseFile ;
Type fichent = fichier de Entier;
var F, G : Fichent ;
v : Entier ;

P:rocédure inverse(E-S/ F, G : Fichent);
var N, j : Entier ;

Début

Relire(F);

/* non fdf(F) */

Alors récrire(G);

/* compter nbre éléments de F */

N:=0;

Tantque non fdf(F) Faire N:=N+1; Lire(F,v); Fait;

/* inverser F dans G */

Tantque (N>0) Faire

Relire(F);

Pour j := 1 à N Faire Lire(F,v); Fait;

écrire(G,v); N:=N-1;

Fait;

Fsi;

Fin;

DEBUT

Assigner(F,'InvFile.d01'); assigner(G,'invFile.d02');
récrire(F); récrire(G);

Ecrire('--- Saisie de F - Arrêt dès lecture d'un nbre <=0 :---');

Ecrire('Donner un nbre >0 : '); Lire(v);

Tantque(v>0) Faire

écrire(F,v);

écrire('Donner un nbre >0 : '); Lire(v);

Fait;

Ecrire('----- Le fichier inverse de F est : -----');

***** Appel de la procédure : */ inverse(F,G);

Relire(G);

Tantque non fdf(G)

Faire Lire(G,v); Ecrire(v, ' '); Fait;

Fermer(F); Fermer(G);

FIN.

Exercice n° 3 : (4 points)

Soient les types suivants :

Type PtrEnt = ↑ élément;

élément = Enregistrement

 Info : entier ;

 suivant : PtrEnt ;

FinEnreg ;

procedure MaxPos(E/ L : PtrEnt; S/ max, pos : Entier);

var p : PtrEnt; i : Entier;

Début

 p := L;

 max := 0; pos := 0;

 i := 0;

 Tantque (p <> NIL) Faire

 i := i + 1;

 Si (max < p^.info) alors max := p^.info;

 pos := i;

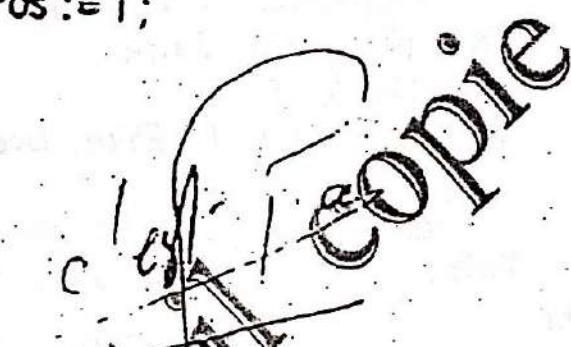
 Fsi;

 P := p^.suivant;

 Fait;

 P := NIL;

Fin;



Rattrapage

Exercice 1 : (pts)

Durée : 01H30

- × Ecrire un algorithme qui calcule pour un nombre réel x , la somme $S(x)$ d'ordre N définie comme suit :

$$S(x) = \sum_{i=0}^N \frac{(-1)^i x^i}{i!}$$

Remarque : on n'utilise que les opérateurs de base (sans l'utilisation de l'opérateur de puissance).

Exercice 2 : (pts)

- × Ecrire une fonction **Carre** vérifiant si un nombre entier naturel est un carré (en utilisant seulement les opérateurs de base). X est un carré s'il existe un entier i tel que $X = i * i$.
- 2) Soit T un tableau de N entiers naturels ($N \leq 50$). Ecrire un algorithme qui lit (remplit) T et affiche tous ses carrés (en utilisant la fonction définie en 1). Puis, il génère un autre tableau C contenant seulement les carrés encadrés (précédés et suivis) par des valeurs paires.

Exemple :

$$\begin{array}{cccccc} T : & 4 & 16 & 36 & 12 & 81 & 25 \\ C : & \underline{16} & \underline{36} & \underline{6} & & & \end{array}$$

Exercice 3 : (pts)

Définition : Une matrice symétrique est une matrice dont la partie inférieure à la diagonale principale est symétrique à la partie supérieure à la diagonale principale (voir exemple).

- 1) Ecrire une action paramétrée permettant de vérifier si une matrice est symétrique.
- 2) Soit une matrice $(N \times N)$ d'entiers avec $N \leq 20$. Ecrire un algorithme qui lit (remplit) une matrice et vérifie si elle est symétrique en utilisant l'action paramétrée, et, dans ce cas, affiche les valeurs non dupliquées ainsi que leurs positions respectives.

Exemple :

	1	2	3	4	5	6
1	3	7	5	2	0	
2	3	-1	2	1	-2	4
3	7	2	-2	6	0	9
4	5	1	6	8	-5	10
5	2	-2	0	-5	-2	-3
6	0	4	9	10	-3	2

Bonne Chance

Épreuve de rattrapage - Semestre 02

Exercice 01 : (03-04 pts)

- 1- Écrire une fonction **palindrome** qui vérifie si un mot donné en entrée, est un mot palindrome.
- 2- Soit **Fmot** un fichier de chaînes de caractères contenant des mots de taille inférieure strictement à 50, écrire un algorithme qui affiche le plus court mot palindrome contenu dans le fichier **Fmot** (déjà rempli).

Exercice 02 : (02-03-02 pts)

Soient les fichiers suivants : **Liste** et **Résultat** de types respectifs **Etudiant** et **Note**, tels que : **Etudiant** =Enregistrement mat: chaîne[12]; prénom, nom: chaîne[30] Fin;
Note=Enregistrement mat: Chaîne[12]; module: chaîne[10]; moy:réel Fin; Non remplir

Le type **Note** représente la moyenne obtenue par l'étudiant **mat** dans le module **module**.

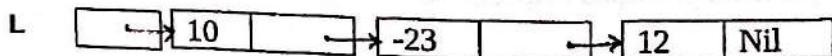
- 1- Écrire une fonction **Moyenne** qui retourne la moyenne générale d'un étudiant **E** de type **Etudiant**, étant donné le fichier **Résultat** (déjà rempli).
- 2- Écrire une procédure **Création** qui crée un fichier **admis** contenant uniquement les étudiants **admis** (**moyenne générale** $>= 10$), tel que chaque étudiant est décrit par son matricule, son nom, son prénom, et sa moyenne générale.

- 3- Écrire une procédure **Statistiques** qui affiche le pourcentage des étudiants ayant eu le module **algorithmique** ('ALGO2')

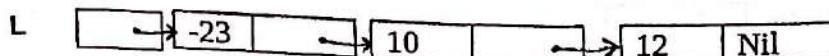
Exercice 03 : (06 pts)

Écrire une action paramétrée **Trier** qui trie une liste linéaire d'entiers dans l'ordre croissant et sans création d'une nouvelle liste.

Exemple



Après le tri, on obtient la liste :



Bon courage

Epreuve Finale (Semestre 2)

Exercice 1 : (6 pts)

Soit le type suivant :

```
Type Produit = Enregistrement Code : Entier ;
          Désignation : Chaîne[80];
          Prix : Réel;
          FinEnreg;
```

Soit F un fichier de produits. Ecrire une fonction qui vérifie si les éléments de F sont triés par ordre croissant de leur code.

Exercice 2 : (7 pts)

L'utilisation des téléphones portables permet de stocker le répertoire des contacts dans deux fichiers :

- Un fichier 'TEL.DAT', enregistré sur la mémoire du téléphone ;
- Un fichier 'SIM.DAT', enregistré sur la mémoire de la carte SIM.

Chaque fichier contient des informations d'un contact regroupant : un nom, un prénom et un numéro de téléphone.

1°_ Donnez la syntaxe d'assignation et d'ouverture des deux fichiers.

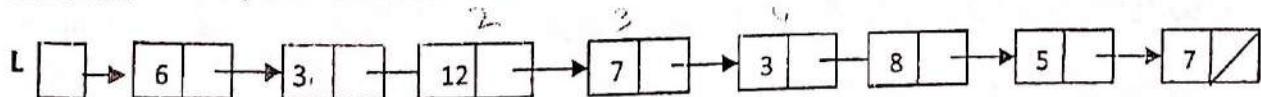
2°_ Ecrire une procédure qui permet de stocker dans un nouveau fichier, les éléments des deux fichiers précédents qui ont le même numéro de téléphone.

Exercice 3 : (7 pts)

Soit L une liste de nombres entiers. Ecrire une procédure qui supprime *N éléments successifs* de L, à partir d'une position K ; si possible.

Si le nombre des éléments restants à partir de K est inférieur à N, la procédure doit supprimer ces éléments restants.

Exemple : Si N=4 , k=2 et la liste l. vaut :



Après suppression, la liste L devient :

