Correction session principale

Partie I

Exercice n°1 : (2,5 points)

- Lire (F1, E) : Cette instruction est correcte.
- Ecrire(F, ch) : Cette instruction n'est pas correcte car ***Ch*** est de type ***chaîne***, ***F*** de type ***fent***; ***les types ne sont pas compatibles***.
- Pointer (F2,4) : Cette instruction n'est pas correcte car la fonction pointer permet d'accéder à l'enregistrement n°4 du fichier F2, or F2 est de type texte et l'accès à ce fichier est séquentiel.
- Ecrire (F1,E.age) : Cette instruction n'est pas correcte car l'instruction ***Ecrire*** permet d'écrire dans un fichier la ***totalité d'un enregistrement***, or ***E.age*** est un ***champ*** d'enregistrement.

Exercice n° 2 : (4 points)

a) U_n est défini à partir de U_{n-1} et U_{n-2}, l'ordre de récurrence est donc 2.

b) Analyse :

```
Def FN Det-rang (p,k : entier) : entier
Resultat = det-rang ← r
r = [U1 ← 2, U2 ← 1]                    (*u2 : Un-2 et u1 : Un-1*)
        Si (p < 0) alors r← -1
        sinon
            Si (p=u2) alors r← 1
            Sinon
                si (p=u1) alors r← 2
                Sinon
                        r  ← -1
                        cpt ← 2
                        Repeter
                                Cpt ← cpt+1
                                Un ←u1 + k*u2
                                U2 ← u1
                                U1 ← Un

                        Jusqu'à (Un >=p)
                        Si Un=p alors
                                r← cpt
                        finsi
                Finsi
        Finsi
Fin def Det-rang
```

c) Algorithme :

```
(0) Debut Det_Rang
(1) U1 = 2
(2) U2 = 1                  (*u2 : Un-2 et u1 : Un-1*)
(3) Lire(p)
(4) repeter
                lire(k)
    jusqu'à (k>0)
(5) Si (p < 0) alors r← -1
    sinon
            Si (p=u2) alors r← 1
            Sinon
                    si (p=u1) alors r← 2
```

```
                    Sinon
                        r  ← -1
                        cpt ← 2
                        Repeter
                            Cpt ← cpt+1
                            Un ←u1 + k*u2
                            U2 ← u1
                            U1 ← Un
                            Si Un=p alors
                                r← cpt
                            finsi
                        Jusqu'à (Un >=p)
                    Finsi
                Finsi
    (6) Fin Det_Rang
```

Exercice n°3 : (3,5 points)

```
DEF Proc Convert-Bin-Hex (var nbin : entier ; var nbhex :chaine)
Resultat = écrire("la conversion de ", nbin,"en hexadécimal est : ", nbhex)
Nbhex = traitement
Traitement =  [convch(nbin,nbinch)
              nbhex ← ‘’
              Nbint ← fn ajout-zero(nbinch)          (* nbint : nombre binaire auquel
                                                        on a rajouté des 0
              l ← long(nbint)
              Nbt ←l div 4]                          nbt : nombre de tranches de 4
              Pour cpt de 1 à nbt faire              chiffres dans nbint*)
                  Nbat ← souschaine(nbint, l-4*cpt+1,4)
                  Val(nbat,bloc,e)
                  Nbhex ← fn conv_bloc(bloc) + nbhex
              Finpour
Nbin = valid_bin(nbin)
Fin conv-bin-hex
```

Tableau de déclaration des objets

Nom	Type/nature	Rôle
nbin	Variable / entier long	Nombre binaire
Nbhex,nbat,nbinch,nbint	Variable / chaine	Nombre hexadécimal et variables intermédiaires
Bloc,e,dif,cpt,nbt,l	Variable / entier	Variables intermédiaires

```
DEF PROC valid_Bin (var nbin : entierlong)
Résultat = nbin
Nbin = []
   Répéter
      Nbin = Donnée("donner un nb binaire")
      str(nbin,nbinch)   cpt ←0  valid ← vrai
      Répéter
         Cpt←cpt+1 Si non (nbinch[cpt] dans ['0','1']) alors valid ← faux
      Jusqu'à (non valid ou (cpt = long(nbinch)))
   Jusqu'à valid
Fin Valid_Bin
```

Tableau de déclaration des objets

Nom	Type/nature	Rôle
valid	Variable / logique	Variable intermédiaire
nbinch	Variable / chaine	Variable intermédiaire
cpt	Variable / entier	Variable intermédiaire

```
DEF FN ajout-zero (nbinch :chaine) : chaine
Resultat = ajout-zero ← nbint
Nbint = [nbint ← nbinch
          Dif ← long(nbinch) mod 4]
        Pour cpt de 1 à (4-dif) faire
              Nbint ← '0' + nbint
        FinPour
Fin ajout-zero
```

Tableau de déclaration des objets

Nom	Type/nature	Rôle
nbint	Variable / chaine	Variables intermédiaires
dif,cpt	Variable / entier	Variables intermédiaires

Partie II

Analyse du programme principal
Nom : Jeu
Resultat = Les-gagnants
Les-gagnants = proc afficher(nt,fabonne)
nt = proc remplir-nt(foot,nt,nb-nt)
foot = proc remplir-foot(foot,fsms,nbsms)
(fsms,nbsms) = proc remplir-fsms(fsms,n,nbsms)
N = proc valider-n(n)
Fin jeu

Tableau de déclaration des objets globaux

Objet	Nature / Type	Rôle
Fabonne	Fichier / fichAbonne	Fichier contenant les abonnés des différents opérateurs téléphoniques (fichier considéré comme saisi)
Nbsms	Variable / entier	Nombre de SMS valides (taille du fichier fsms)
Foot	Variable / TabFoot	Vecteur de nbsms enregistrements contenant pour chaque SMS reçu, le n° tel, la proposition du joueur (validée), son score
Fsms	Fichier / fichSms	Fichier d'enregistrements contenants tous les SMS validés pour ce jeu.
N,nb_nt	Variable / entier	Nombre maximum de sms pouvant être reçus pour le jeu (n <= 5000) Nb_Nt : Nombre d'éléments dans le tableau NT
nt	Variable / TabNt	Vecteur de Nb_Nt entier long, représentant les numéros des téléphones des gagnants sans

		redondance.
Procédure afficher	Procédure	Affichage du nom des gagnants
Remplir-nt	Procédure	Remplissage du tableau NT
Remplir-foot	Procédure	Remplissage du tableau foot
Remplir-fsms	Procédure	Remplissage du fichier fsms et détermination du nombre de sms valides (nbsms)
Valider-n	Procédure	Validation du nombre maximum de sms pouvant être pris en charge par le jeu

Tableau de déclaration des nouveaux types

Nom
RecFoot = enregistrement Numtel : entier long Clas : chaîne de 4 caractères Score : entier Fin tab TabNT = tableau[1..5000] d'entier longs TabFoot = tableau [1..5000] de type recFoot RecSms = enregistrement Numtel : entier long Clas : chaîne de 4 caractères Fin RecSms RecAbonne = enregistrement Numtel : entier long Nom: chaine Prenom : chaine Fin RecAbonne
FichSms = Fichier de type RecSms FichAbonne = Fichier de type RecAbonne

Algorithme du programme principal

0) Début Jeu
1) Proc valider-n (n)
2) Proc remplir-fsms(fsms,n,nbsms)
3) Proc remplir-foot(foot,fsms)
4) Proc remplir-NT(foot,nt,nb-nt)
5) Proc afficher(nt,fabonne)
6) Fin jeu

Analyse de la procédure **valider-n**	Algorithme de la procédure **valider-n**
Def proc valider-n (var n : entier)	0) Début valider-n (var n : entier)
Resultat = n	1) Répéter
n= []	n = donnée
Répéter	2) Jusqu'à n dans [1..5000]
n = donnée	3) Fin valider-n
Jusqu'à n dans [1..5000]	
Fin valider-n	

Analyse de la procédure **remplir-Fsms**	Algorithme de la procédure **remplir-Fsms**
Def proc **remplir- Fsms** (var fsms : fichsms ; n : entier ; var nbsms :entier)	0) Début Proc **remplir- Fsms** (var fsms : fichsms ; n : entier ; var nbsms :entier)
Resultat = fsms-rempli Fsms-rempli = [init(fsms) Nbsms ← 0] Pour cpt de 1 à n faire Nt = donnée('n° de téléphone) Cl : donnée('Classement') Si (long(cl) =4) alors Debut Si fn(valid-choix(cl) alors Nbsms ← nbsms+1 Avec enreg-sms faire Numtel ← nt Clas ← cl Finavec Ecrire(fsms,enregsms) finsi Finsi Finpour Fermer(fsms) Fin remplir-fsms	1) init(fsms) 2) Nbsms ← 0 3) Pour cpt de 1 à n faire Lire(Nt) Lire(Cl) Si (long(cl) =4) alors Debut Pour j de 1 à 4 faire cl[j] ← majus(cl[j]) FinPour Si fn(valid-choix(cl) alors Nbsms ← nbsms +1 Avec enreg-sms faire Numtel ← nt Clas ← cl Finavec Ecrire(fsms,enregsms) Finsi finSi Finpour 4) Fermer(fsms)

Tableau de déclaration des objets locaux

Nom	Type/nature	Rôle
Enregsms	Variable / Recsms	Enregistrement contenant la participation d'un joueur
Cpt,j	Variable / Entier	Compteur
Cl	Variable / chaine[4]	Classement d'un joueur
Nt	Variable / entierlong	Numéro de téléphone d'un joueur

Analyse de la procédure **init**	Algorithme de la procédure **init**
Def proc Init (var fsms : fichsms)	0) Début Proc Init (var fsms : fichsms)
Resultat = fsms-ouvert Fsms-ouvert = associer(fsms, "c:\sms.dat") Recréer(fsms) Fin init	1) Associer (fsms, "c:\sms.dat") 2) Recréer(fsms) 3) Fin init

Analyse de la fonction **valid-choix**	Algorithme de la fonction **valid-choix**
Def fn valid-choix (var c:ch4) : logique	Début fn **valid-choix** (var c:ch4) : logique
Resultat = valid-choix ← ch-valid Ch-valid = [ch-valid ← vrai] Si ((pos('I',c) = 0) ou (pos('F',c) = 0) ou (pos('A',c) = 0) ou (pos('P',c) = 0) ou (long(c) <> 4)) alors ch-valid← faux Finsi Fin valid-choix	0) ch-valid ← vrai 1) Si ((pos('I',c) = 0) ou (pos('F',c) = 0) ou (pos('A',c) = 0) ou (pos('P',c) = 0) ou (long(c) <> 4)) alors ch-valid← faux Finsi 2) valid-choix ← ch-valid 3) Fin valid-choix

Tableau de déclaration des objets locaux

Nom	Type/nature	Rôle
Ch-valid	logique	Validation d'un classement
Cpt	Variable / entier	Compteur

Analyse de la procédure **remplir-foot**	Algorithme de la procédure **remplir-foot**
Def proc **remplir-foot** (var foot:tab; fsms:fichsms ; nbsms :entier)	Début proc **remplir-foot** (var foot:tabFoot; fsms:fichsms)
Resultat = foot-formé Foot-formé = [ouvrir(fsms),cpt←0] Tantque (non fdf(fsms)) faire Lire(fsms,enreg_Sms) Avec enreg_sms faire Cpt ← cpt+1 Foot[cpt].numtel ← enregSMS.numtel Foot[cpt].clas ← enregSMS.clas Foot[cpt].score ← fn calcul-score(enregSMS.clas) FinAvec FinTantque Fermer(fsms) Fin Remplir-foot	0) ouvrir(fsms)] cpt ←0 1) Tantque (non fdf(fsms)) faire Lire(fsms,enreg_Sms) Avec enreg_sms faire Cpt ← cpt+1 Foot[cpt].numtel ← enregSMS.numtel Foot[cpt].clas ← enregSMS.clas Foot[cpt].score ← fn calcul-score(enregSMS.clas) FinAvec FinTantque 2) Fermer(fsms) 3) Fin Remplir-foot

Tableau de déclaration des objets locaux

Nom	Type/nature	Rôle
Enreg_SMS	Variable / recSMS	Enregistrement du fichier Fsms
Cpt	Variable / entier	Compteur

Analyse de la fonction **calcul-score**	Algorithme de la fonction **calcul-score**
Def fn calcul-score (c:ch4) : entier	Début fn calcul-score (c:ch4) : entier
Resultat = (calcul-score ← total) total = [total ← 0] Pour i de 1 à 4 faire Si c[i] = sol[i] alors Total ← total + 25 Finsi Finpour Fin calculer-score	0) total ← 0 1) Pour i de 1 à 4 faire Si c[i] = sol[i] alors Total ← total + 25 Finsi Finpour 2) calcul-score ← total 3) fin calculer-score

Tableau de déclaration des objets locaux

Nom	Type/nature	Rôle
Sol	Constante = 'IFAP'	Classement final de la coupe du monde
I	Variable / entier	Compteur
Total	Variable / entier	Variable contenant le calcul du score d'un joueur

Analyse de la procédure **remplir-NT**	Algorithme de la procédure **remplir-NT**
Def proc **remplir-NT** (foot :tabFoot ; VAR nt :tabNT ;VAR nb-nt :entier)	Début proc **remplir-NT** (foot :tabFoot ; VAR nt :tabNT; VAR nb-nt :entier)
Resultat = affichage affichage = [max ← recherche-max(foot, nbsms) nb-nt ← 0] pour cpt de 1 à nbsms faire Si foot[cpt].score = max alors Si non existe(foot[cpt].numtel,nt,nb-nt) alors Ajouter-nt(foot[cpt].numtel,nt,nb-nt) finsi finPour Fin Remplir-NT	0) max ← recherche-max(foot, nbsms) 1) nb-nt ← 0 2) pour cpt de 1 à nbsms faire Si foot[cpt].score = max alors Si non existe(foot[cpt].numtel,nt,nb-nt) alors Ajouter-nt(foot[cpt].numtel,nt,nb-nt) finsi finPour 3) Fin Remplir-NT

Tableau de déclaration des objets locaux

Nom	Type/nature	Rôle
Recherche-max	fonction	Recherche du score maximum dans le tableau foot.
Cpt,max	Variable / entier	Compteur, score maximum obtenu par les participants
Existe	Fonction	Recherche de l'existence d'un numéro de téléphone dans le tableau NT
Ajouter	Procédure	Ajout d'un nouveau numéro de téléphone dans le tableau NT.

Analyse de la fonction **recherche-max**	Algorithme de la fonction **recherche-max**
Def fn **recherche-max** (foot :tab; nbsms :entier) :entier	Début fn **recherche-max** (foot :tabFoot; nbsms :entier) :entier
Resultat = recherche-max ← maxnt maxnt = [maxnt ← foot[1].score] pour cpt de 2 a nbsms faire si foot[cpt].score > maxnt alors maxnt ← foot[cpt].score finsi finpour Fin Recherche-max	0) Maxnt ← foot[1].score 1) pour cpt de 2 a nbsms faire si foot[cpt].score > maxnt alors maxnt ← foot[cpt].score finsi 2) Finpour 3) recherche-max ← maxnt 4) Fin recherche-max

Tableau de déclaration des objets locaux

Nom	Type/nature	Rôle
Max-nt	Variable / entier	Score maximum dans le tableau foot
Cpt	Variable / entier	Compteur

Analyse de la fonction **existe**	Algorithme de la fonction **existe**
Def fn **existe** (num : entierlong ; nt :tabnt ; nb-nt :entier) : logique	Début fn **existe** (num : entierlong ; nt :tabnt ; nb-nt :entier) : logique
Resultat = existe ← trouve trouve = [trouve ← faux ; cpt←0] si nb_nt <>0 alors Répéter Cpt ← cpt +1 Si nt[cpt] = num alors trouve ← vrai finsi Jusqu'à ((trouve) ou (cpt=nb-nt)) Existe ← trouve Sinon Existe ← faux Finsi Fin existe	0) trouve ← faux 1) cpt←0 2) si nb_nt <>0 alors Répéter Cpt ← cpt +1 Si nt[cpt] = num alors trouve ← vrai finsi Jusqu'à ((trouve) ou (cpt=nb-nt)) Existe ← trouve Sinon Existe ← faux finsi 3) Fin existe

Tableau de déclaration des objets locaux

Nom	Type/nature	Rôle
Trouve	Variable / logique	Variable intermédiaire
Cpt	Variable / entier	Compteur

Analyse de la procédure **ajouter-nt**	Algorithme de la procédure **ajouter-nt**
Def proc **ajouter-nt** (num : entierlong ; var nt :tabnt ; var nb-nt :entier)	Début proc **ajouter-nt** (num : entierlong ; var nt :tabnt ; var nb-nt :entier)
Resultat = (nt,nb-nt) (nt,nb-nt) = [nb-nt ← nb-nt +1] Nt[nb-nt] ← num Fin ajouter-nt	0) nb-nt ← nb-nt +1 1) Nt[nb-nt] ← num 2) Fin ajouter-nt

Analyse de la procédure **Afficher**	Analyse de la procedure **Afficher**
Def proc **Afficher** (nt : Tabnt; fabonne : fichabonne)	Debut procédure **Afficher** (nt : Tabnt; var fabonne : fichabonne)
Resultat = affichage affichage = [associer(fabonne, "c:\abonnes.dat")] Pour cpt de 1 à nb-nt faire ouvrir(FAbonne) trait← faux Tant que non ((trait) ou fin-fichier(fabonne)) faire Lire (fabonne, enreg-fabonne) Si enreg-fabonne.numtel = nt[cpt] alors Ecrire(enreg-fabonne.nom) Ecrire(enreg-fabonne.prenom) Trait ← vrai Finsi Fintantque FinPour Fermer(fabonne) fin afficher	0) associer(fabonne, "c:\abonnes.dat"), 1) Pour cpt de 1 à nb-nt faire ouvrir(FAbonne) trait← faux Tant que non ((trait) ou fin-fichier(fabonne)) faire Lire (fabonne, enreg_fabonne) Si enreg_fabonne.numtel = nt[cpt] alors Ecrire(enreg-fabonne.nom) Ecrire(enreg-fabonne.prenom) Trait ← vrai Finsi Fintantque Finpour 2) Fermer(fabonne) 3) fin afficher

Tableau de déclaration des objets locaux

Nom	Type/nature	Rôle
trait	Variable / logique	Variable intermédiaire
Cpt	Variable / entier	Compteur
Enreg-fabonne	Variable / recabonne	Enregistrement du fichier fabonne