Correction TD 4
Fichiers

Question 1

On commence par déclarer un enregistrement pour représenter l'employé :

```
Employe : ENREGISTREMENT
   matricule : entier
   nom : tableau de 30 caractères
   prenom : tableau de 30 caractères
   total_heures : entier #on ne compte pas les demi-heures
   taux_horaire : réel
FIN ENREGISTREMENT
```

Pour cet exercice, on travaillera sur le fichier "employe.dat".
C'est l'algorithme principal qui, à chaque lancement, vérifiera si le fichier existe ou non. Dans le cas où le fichier n'existe pas, c'est l'algorithme principal qui se chargera de le créer.
Fonction d'ajout :

```
ALGORITHME ajout
   ENVIRONNEMENT
      Entree :
         néant
      Sortie :
         néant
      Internes :
         e : Employe
         ref_fichier : Fichier d'Employes
   TRAITEMENT
   DEBUT
      # On initialise e champ par champ
      ECRIRE "matricule de l'employe?"
      e.matricule ← lire
      ECRIRE "nom de l'employe?"
      e.nom ← saisir(e.nom)
      ECRIRE "prénom de l'employe?"
      e.prenom ← saisir(e.prenom)
      #puisque l'employé arrive dans la base
      e.total_heures ← 0
      ECRIRE "taux horaire de l'employé?"
      e.taux_horaire ← LIRE
      # On peut désormais l'enregistrer
      ref_fichier ← ouvrir_en_ajout("employe.dat")
      ecrire(ref_fichier, e)
      fermer(ref_fichier)
   FIN
FIN ALGORITHME
```

Question 2

On commence par créer une fonction pour afficher un seul employé, afin de pouvoir la réutiliser si besoin (et éviter de devoir tout recoder à chaque fois qu'on veut en afficher un)

```
ALGORITHME aff_employe
   ENVIRONNEMENT
      ENTREE :
         e : Employe
      SORTIE :
```

```
         néant
      INTERNES :
         néant
   TRAITEMENT
      DEBUT
         ECRIRE "matricule : ", e.matricule
         ECRIRE "nom : "
         afficher(e.nom)
         ECRIRE "prénom : "
         afficher(e.prenom)
         ECRIRE "total heures travaillées : " , e.total_heure
         ECRIRE "taux horaire de l'employé : " e.taux_horaire
      FIN
FIN ALGORITHME
```

Affichage de tous les employés :

```
ALGORITHME aff_all_employes
   ENVIRONNEMENT
      ENTREE :
         néant
      SORTIE :
         néant
      INTERNE :
         ref_fichier : Fichier d'Employes
         e : Employe # On pourrait faire sans cette variable, en mettant l'appel à la fonction "lire"
     directement dans aff_employe
   TRAITEMENT
      DEBUT
         # On commence par ouvrir le fichier
         ref_fichier ← ouvrir_en_lecture("employe.dat")

         # On parcourt le fichier employé par employé
         TANT QUE (NON eof(ref_fichier)) FAIRE
            # Récupération de l'employé qui est situé au niveau du curseur (et déplacement automatique du
     curseur sur l'employé suivant)
            e ← lire(ref_fichier)

            # Affichage de l'employé
            aff_employe(e)
         FIN TANT QUE

         # On n'oublie pas de fermer le fichier !
         fermer(ref_fichier)
      FIN
FIN ALGORITHME
```

Question 3

```
ALGORITHME suppr_employe
   ENVIRONNEMENT
      ENTREE :
         matricule_recherche : entier
      SORTIE :
         néant
      INTERNE :
         ref_fichier : Fichier d'Employes
         tab : tableau de 1000 Employe #on réserve l'espace temporaire en RAM
         e : Employe
         compteur : entier
         index : entier
   TRAITEMENT
      DEBUT
         compteur ← 0
         # On commence par ouvrir le fichier
         ref_fichier ← ouvrir_en_lecture("employe.dat")
         TANT QUE (NON eof(ref_fichier)) FAIRE
            e ← lire(ref_fichier)
            SI (e.matricule != matricule_recherche) FAIRE #on ne copiera pas l'employé recherché
               tab[compteur] ← e
               compteur ← compteur + 1 #on veut écrire dans la case suivante au prochain Employe
```

```
            FIN SI
         FIN TANT QUE
         fermer(ref_fichier)
         #on peut maintenant écraser le fichier original pour y écrire le contenu du tableau
         ref_fichier ← ouvrir_en_creation("employe.dat")
         POUR index ALLANT DE 0 A (compteur - 1) PAR PAS DE 1 FAIRE
            ecrire(ref_fichier, tab[index])
         FIN POUR
         fermer(ref_fichier)
      FIN
FIN ALGORITHME
```

Question 4

```
ALGORITHME maj_h_employe
   ENVIRONNEMENT
      ENTREE :
         matricule_recherche : entier
      SORTIE :
         néant
      INTERNES :
         ref_fichier : Fichier d'Employes
         tab : tableau de 1000 Employe #on réserve l'espace temporaire en RAM
         e : Employe
         compteur : entier
         index : entier
   TRAITEMENT
      DEBUT
         compteur ← 0
         #On commence par ouvrir le fichier
         ref_fichier ← ouvrir_en_lecture("employe.dat")
         TANT QUE (NON eof(ref_fichier)) FAIRE
            e ← lire(ref_fichier)
            SI (e.matricule = matricule_recherche) FAIRE
               ECRIRE "nombre d'heures actuel : " , e.total_heures
               ECRIRE "nouveau nombre d'heures ?"
               e.total_heures ← LIRE
            FIN SI
            tab(compteur) ← e
            compteur ← compteur + 1
         FIN TANT QUE
         fermer(ref_fichier)
         #on peut maintenant écraser le fichier original pour y écrire le contenu du tableau
         ref_fichier ← ouvrir_en_creation("employe.dat")
         POUR index ALLANT DE 0 A (compteur - 1) PAR PAS DE 1 FAIRE
            ecrire(ref_fichier, tab(index))
         FIN POUR
         fermer(ref_fichier)
      FIN
FIN ALGORITHME
```

Question 5

```
ALGORITHME moy_salaire
   ENVIRONNEMENT
      ENTRÉES :
         matricule_recherche : entier
      SORTIE :
         réel #la moyenne des salaires
      INTERNES :
         ref_fichier : Fichier d'Employes
         e : Employe
         compteur : entier
         somme_salaire : réel
   TRAITEMENT
      DEBUT
```

```
        compteur ← 0
        somme_salaire ← 0
        #On commence par ouvrir le fichier
        ref_fichier ← ouvrir_en_lecture("employe.dat")
        TANT QUE (NON eof(ref_fichier)) FAIRE
           e ← lire(ref_fichier)
           somme_salaire  ← somme_salaire + e.taux_horaire × e.total_heures
           compteur ← compteur + 1
        FIN TANT QUE
        fermer(ref_fichier)
        #on peut maintenant calculer la moyenne et la renvoyer
        moy_salaire ← somme_salaire / compteur
     FIN
FIN ALGORITHME
```

Question 6

On réalise une première fonction qui trie les n premiers éléments d'un tableau de Employe de façon décroissante selon le nombre d'heures.
Pour ceci, on trouve le maximum et on le met à la première case, puis on passe à la suivante, etc.
NB : ce tri n'est pas performant (sa complexité algorithmique est mauvaise)

```
ALGORITHME tri_decroissant
    ENVIRONNEMENT
        ENTRÉES :
            tab : tableau de Employe # modifié
            n : nombre de cases
        SORTIE :
            néant
        INTERNE :
            i , j : entier
            e_tmp : Employe
    TRAITEMENT
        DEBUT
            POUR i ALLANT DE 0 A n-1 FAIRE
                # Si un des éléments suivants est plus grand, on le met à la place
                POUR j ALLANT DE i+1 A n-1 FAIRE
                    SI (tab[i].total_heures  <  tab[j].total_heures) FAIRE
                        # Permutation des valeurs
                        e_tmp <- tab[i}
                        tab[i] <- tab[j]
                        tab[j] <- e_tmp
                    FIN SI
                FIN POUR
            FIN POUR
        FIN
FIN ALGORITHME
```

Fonction qui affiche les employés du fichier dans l'ordre décroissant des heures travaillées :

```
ALGORITHME aff_h_decroissant
    ENVIRONNEMENT
        ENTRÉES :
            matricule_recherche : entier
        SORTIE :
            néant
        INTERNES :
            ref_fichier : Fichier d'Employes
            tab : tableau de 1000 Employe #on réserve l'espace temporaire en RAM
            compteur : entier
            index : entier
    TRAITEMENTS
        DEBUT
            compteur <- 0

            # On commence par ouvrir le fichier
            ref_fichier <- ouvrir_en_lecture("employe.dat")
```

```
            # On copie son contenu dans un tableau
            TANT QUE (NON eof(ref_fichier)) FAIRE
                tab[compteur] <- lire(ref_fichier)
                compteur <- compteur + 1
            FIN TANT QUE

            fermer(ref_fichier)

            #On trie le tableau
            tri_decroissant(tab , compteur - 1)

            # On affiche le tableau trié
            POUR index ALLANT DE 0 A compteur - 1 PAR PAS DE 1 FAIRE
                aff_employe(tab[index])
            FIN POUR

        FIN
FIN ALGORITHME
```