UNIVERSITE DE OUAGADOUGOU

ECOLE SUPERIEURE D'INFORMATIQUE

DELEGATION GENERALE

A L'INFORMATIQUE

RAPPORT DE PROGRAMMATION

ANNEE ACADEMIQUE 1993-1994

Présenté par M.DJIGUIMDE Marc .T

Etudiant à ESI

Sous la direction de :

Maître de stage
M.NOMBRE Lassané
Ingénieur à la DEL.G.I

Superviseur ESI
M.ZONGO Sylvain
Enseignant à l'ESI

UNIVERSITE DE OUAGADOUGOU
ECOLE SUPERIEURE D'INFORMATIQUE

01 BP 7021 OUAGADOUGOU 01
TEL :30 16 81

DELEGATION GENERALE
A L'INFORMATIQUE(DEL.G.I)

RAPPORT

DE

PROGRAMMATION

ANNEE UNIVERSITAIRE 1993-1994

Présenté par M.**DJIGUIMDE MARC .T**
Etudiant 2è année ESI

Sous la direction de:

Maître de stage
M.**Lassané NOMBRE**
Ingénieur à la DEL.G.I

Superviseur ESI
M.**Sylvain ZONGO**
Enseignant à l'ESI

SOMMAIRE

INTRODUCTION

Comme nous le signalions dans notre rapport d'insertion nous entamons actuellement la phase de programmation.

Ce rapport a pour but de présenter les procédures fonctionnelles que nous aurons à programmer;lesquelles constituent une partie de l'application développée par les étudiants de 3ème annee de la première promotion de l'école supérieure d'informatique dans le cadre de leur projet de fin d'étude.Lequel projet a pour but l'informatisation de la direction des affaires académiques et scolaires de l'université de ouagadougou (DAAS).

Pour des contraintes de temps nous fusionnons le rapport de la prémière phase avec celui de la deuxième.C'est un bon esprit ouvert à apprentissage.c'est donc avec une grande volonté d'accomplir de façon satisfaisante le travail qui nous a été demandé.

I.**PRÉSENTATION GENERALE DE L'APPLICATION**

Vu que le nombre des étudiants de l'université de ouagadougou devient pléthorique au fil des années,leur gestion devient de plus en plus difficile.Pour palier ce problème l'outil informatique est le plus indiqué.C'est donc dans cet optique que la conception de l'application qui vous sera présentée a vu jour.

La D.A.A.S charpente centrale de l'université de ouagadougou a pour attributions:

-D'assurer l'organisation,l'information et le suivi administratif et académique de la scolarité de tous les étudiants.

-D'aider à orienter les étudiants vers les filières de formation.

-La supervision technique des activités de toutes les scolarités de l'université.

Cette application doit faciliter la gestion des affaires académiques et scolaires des étudiants;car elle permettra :

- L'enregistrement de toutes les données concernant chaque étudiant ou un éventuel étudiant.

-Le tirage des certificats d'inscription ,des attestations de succès , des relevés de votes ,des cartes d'étudiant.

-La publication des bulletins statistiques annuels.

-L'archivage des dossiers de personnes ayant fini leurs études à l'université.

-Le suivi des frais d'inscription.

-Communication permanente des informations aux autres structures chargée également de gérer les étudiants.

Cette application est développée sous un système de gestion de base de données relationnel(SGBDR) associé à un langage de programmation (ORACLE associé à PRO*C).L'application est organisée en postes de travail qui assurent les mises à jour et les consultations de la base de données.Le mode réseau a été choisi pour des raisons qui sont entre autres la rapidité et quasi simultanéité des opérations.

Il y'a sept postes de travail qui sont:

-**Le poste service accueil** qui s'occupe des demandes d'accès à l'université ,de la saisie des informations de départ et de la gestion des résultats des demandes.

-**Le poste inscription et réinscription** qui s'occupe du complément d'informations pour ceux dont la demande a reçu un avis favorable à une inscription effective à des programmes et de la gestion de leurs résultats.

-**Le poste titres et diplômes** qui s'occupe de la délivrance des diplômes.

-**Le poste gestion des programmes d'enseignement** qui précise les programmes par filière et par année et de gérer les enseignants et leurs horaires.

-**Le poste direction** chargé du contrôle de tout l'ensemble du système de gestion.

-**Le poste statistique** chargé de faire les états statistiques des inscriptions et des résultats suivant des paramètres donnés.

II.**RÈGLES DE GESTION**

Il y'a une trentaine de règles de gestion et d'organisation.Mais celles relatives aux procédures que nous devons programmer sont les suivantes:

-En dehors des bacheliers et étudiants boursiers de l'état,tous ceux désireux de s'inscrire pour la premières fois à l'université de ouagadougou doivent déposer au rectorat un dossier de demande d'inscription.

-Les candidats non titulaires du baccalauréat doivent obligatoirement passer l'examen spécial d'entrée à l'université.

-L'admission à l'examen spécial donne droit à une attestation de succès au dit examen.

-Tout étudiant inscrit a droit à une carte d'étudiant obligatoire pour l'accès en salle d'examen.

-Dès la premières inscription ,l'étudiant reçoit un numéro matricule pour toute sa scolarité à l'université de ouagadougou .

-Tout étudiant inscrit peut demander une attestation d'inscription qui lui est délivrée.

-l'admission à l'examen spécial donne droit à une attestation de réussite signée par le directeur de la D.A.A.S.

-Tout étudiant qui passe un examen reçoit de la scolarité de son établissement un relevé de notes.

-Le succès à l'examen donne droit à une attestation de succès.

-Un étudiant peut passer plusieurs examens (pendant la durée de ses études);ces examens pouvant être composés de plusieurs matières et faisant l'objet d'un procès verbal.

-Un étudiant a un et un seul statut (Boursier / non Boursier/ Salarié) lié à son inscription une année scolaire donnée.

-Après admission à un examen de fin d'année ,l'étudiant reçoit :

*Un relevé de notes délivré par son établissement et signé par le président de jury.

*Une attestation de succès délivrée au service d'inscription et de réinscription et signée par le secrétaire général.

III.DESCRIPTION DES PROCÉDURES FONCTIONNELLES A PROGRAMMER

1.LES EDITIONS COURANTES

a.*Tirage des attestations définitives de succès (de tous les étudiants admis à un programme d'inscription donné)suivies d'un bordereau des attestations pour émargement.*

Cette unité de traitement a pour but d'éditer(édition batch):

*Les attestations définitives de succès de tous les étudiants admis à un programme d'inscription donné.Le numéro de pv ou le code programme saisi interactivement permet de sélectionner le programme auquel ils sont admis.

*Un bordereau qui doit contenir le nom ,le prénom,le matricule,date retrait et la date de naissance de chaque étudiant admis à ce programme pour émargement lors du retrait de l'attestation.

Maquette d'édition (Voir annexe1).

b.*Tirage des procès verbaux vierges ou renseignés des inscrits à un programmes donné.*

Cette unité à pour but d'éditer:

*Les pv non renseignés(sans les différentes notes et observations) de tous les inscrits à un programme sélectionné interactivement.

*Les pv renseignés (avec les notes par matière et les observations) de tous les étudiants inscrits à un programme sélectionné interactivement.

Maquette d'édition(Voir annexe2).

c.*Edition des résultats(admis) de concours ,examen spécial ou pour études de dossiers.*

Il s'agit d'éditer la liste des candidats admis à un concours ,un examen spécial ou pour étude de dossiers.

Un menu permet de préciser s'il s'agit d'un concours ,d'un examen spécial ou d'une étude de dossiers.

Maquette (Voir annexe3).

d.*Tirage d'une fiche diplôme contenant les informations à retracer sur un diplôme vierge ainsi que les informations attestant le droit de l'étudiant à ce diplôme.*

Maquette (Voir annexe4).

e.*Edition du cursus universitaire d'un étudiant à partir du numéro matricule.*

Il s'agit d'éditer le cursus universitaire d'un étudiant dont le numéro matricule est saisi interactivement à partir un menu.

Maquette (Voir annexe5).

f.*Etat des frais d'inscription (étudiants non à jour)par établissement,par département, avec rupture de séquences.*

Il s'agit d'éditer pour chaque établissement suivant ses départements les noms, puis le montant dû, le montant payé et le montant restant des étudiants no à jour de leurs frais d'inscription.

Maquette (Voir annexe6).

2.**ETAT STATISTIQUE**

Edition pour chaque établissement et pour une année donnée du nombre d'inscrits, des admis, du pourcentage des admis par niveau d'étude, par sexe et par pays.

Maquette (Voir annexe7).

IV.***DESCRIPTION DES DIFFERENTES TABLES DE LA BASE DE DONNEES UTILISEE***

(Voir annexe8)

V.**METHODE GENERALE EMPLOYEE**

La méthode employée pour la programmation de l'unité de traitement présentée plus haut est la suivante:

- Décomposer chaque état en trois(3) parties:
  1. ENTETE.
  2. CORPS.
  3. BAS DE PAGE.
- Utiliser une boucle d'itération pour itérer sur les trois(3) parties ou sur le corps, suivant que c'est le même document qui est à édité plusieurs fois ou que dans le même document on doit faire ressortir plusieurs lignes contenant des informations du même genre mais différant d'une ligne à l'autre.

De ce fait la plus part des programmes sont décomposent en trois(3) procédures essentielles qui sont:

- ENTETE.
- AFFICHER_CORPS.
- BAS_DE_PAGE.

TABLE DES IDENTIFICATEURS

IDENTIFICATEURS	TYPE	CORRESPOND
fp	pointeur sur fichier	utilisé pour diriger la sortie vers l'imprimante
cpt	entier	compteur
mat	chaine de caractères	matricule de l'étudiant
année	chaine de caractères	année académique
te_mt_paye	flottant	total montant payé par établissement
td_mt_paye	" "	total montant payé par département
tg_mt_paye	" "	total général montant payé
te_mt_du	entier	total montant dû par établissement
td_mt_du	" "	total montant dû par département
tg_mt_du	" "	total général du montant dû
mt_du	entier	montant dû
td_mt_reste	flottant	total reste à verser par département
tg_mt_reste	" "	total général du reste à verser
choix	chaine de caractères	choix
lib_prog	pointeur sur une chaine de caractères	libellé du programme
reste	flottant	reste à verser
mt_quit	flottant	montant de la quittance

ALGORITHMES

Il est à préciser que la récupération des données de la base de données est faite sous forme de requettes SQL.Ipso facto vous ne verrez pas de raffinement pour certaines procédures qui ne contiennent que des requettes et l'affichage des données sélectionnées.Cependant la sélection sera indiquée s'il s'agit d'un curseur.De plus pour éviter de surcharger les algorithmes il a été volontairement omises les procédures de détournement en cas d'érreur ou de données non existantes dans la base de données, eut égard du fait que des commandes SQL permettent de le faire.

ATTESTATION

```
a) RESULTAT
   Edition des attestations de succès directement sur
         imprimante.
b)DONNEES
      Base de données.
      code_prog.
c)VARIABLES DE TRAVAIL
      (voir table des identificateurs)
d)ENCHAINEMENT DES OPERATIONS
  SI not(CONNEXION) ALORS
     exit
  SINON
     effacer_ecran
     code_prog<-lire_code_programme
      SI TEST<0 ALORS
       aucun étudiant n'a droit à une attestation.
     SINON
       ouv_fich_imprimante
       select_num_pv_correspdt_au_code_prog_par_curs_prog
       ouvrir_curs_prog
       TANTQUE curs_prog_non_vide REPETER
          select_un_num_pv_du_curseur
          select_etdiant_admis_le_num_pv_par_curs_etudiant
          ouvrir_curs_etudiant
```

```
            TANTQUE non_fin_curs_etudiant REPETER
                select_un_etudiant_du_curseur
                ENTETE
                AFFICHER_CORPS
                BAS_DE_PAGE
            FINTANTQUE
            fermer_curs_etudiant
          FINTANTQUE
          fermer_curs_prog
        ferme_fich_imprimante
      FINSI
   FINSI.


  RAFFINEMENT AFFICHER_CORPS
       selection_données
        afficher_données
  FINRAFFINEMENT.




 RAFFINEMENT ENTETE
     afficher_entête_d'une_attestation
 FINRAFFINEMENT


 RAFFINEMENT BAS_DE_PAGE
     selection_date_système
     afficher_date_du_jour
     afficher_bas_de_page_attestation
 FINRAFFINEMENT.


RAFFINEMENT TEST
   select_et_compter_nb_etudiant_admis_dans_programme
    SI nb=0 ALORS
       retourné -1
    SINON
       retourné 0
    FINSI
FINRAFFINEMENT
```

BORDEREAU DES ATTESTATIONS

a) RESULTAT
Edition d'un bordereau des attestations pour emargement
b) DONNEES
Base de données
code_prog
c) VARIABLES DE TRAVAIL
(voir table des identificateurs)
d) ENCHAINEMENT DES OPERATIONS

```
SI not(CONNEXION) ALORS
    exit
 SINON
    effacer_ecran
    code_prog<-Lire_code_programme
    SI TEST <0 ALORS
       afficher"AUCUN ETUDIANT N'EST ADMIS "
    SINON
       ouv_fich_imprimante
       effacer_ecran
      select_num_pv_correspdt_au_code_prog_par_curs_prog
       ouvrir_curs_prog
       TANTQUE non_fin_curs_prog REPETER
           select_un_num_pv_du curseur
          select_nom_etdt_admis_sur_pv_par_curs_nom_etdt
           ouvrir_curs_nom_etudiant
           SI TEST2>0 ALORS
               ENTETE
               AFFICHER_CORPS
               BAS_DE_PAGE
           FINSI
       FINTANTQUE
       fermer_curs_prog
       fermer_fich_imprimante
    FINSI
FINSI.


RAFFINEMENT ENTETE
    Selection_données_utiles_à_ENTETE
    afficher_données
FINRAFFINEMENT
```

```
RAFFINEMENT AFFICHER_CORPS

    TANTQUE curs_nom_etudiant_non_vide REPETER
        selectionner_une_ligne_du curseur
        afficher_données_selectionnées
    FINTANTQUE
    fermer_curs_nom_etudiant
FINRAFFINEMENT.
```

PROCES-VERBAL RENSEIGNE

a) RESULTAT
   Edition des pv renseignés des inscrits à un programme donné.
b) DONNEES
   Base de données
   code_prog
c) VARIABLES DE TRAVAIL
   (voir table des identificateurs)
d) ENCHAINEMENT DES OPERATIONS

```
code_prog<-Lire_code_programme
SI not(CONNEXION) ALORS
   exit
SINON
  SI TEST<0 ALORS
     code programme inexistant
  SINON
     effacer_ecran
     ouv_fich_imprimante
     Select_num_pv_correspdt_au_programme_par_curs_pv
     ouvrir_curs_pv
      TANTQUE curs_pv_non_vide REPETER
       select_un_num_pv_du_curseur
       ENTETE
       AFFICHER_CORPS
       BAS_DE_PAGE
     FINTANTQUE
     fermer_curs_pv
     fermer_fich_imprimante
  FINSI
FINSI.
```

```
RAFFINEMENT ENTETE
   selectionner_nom_matière_par_curs_cours
   selection_autres_données_pour_entête
   afficher_autres_données_selectionnées
   ouvrir_curs_cours
   TANTQUE curs_cours_non_vide REPETER
       afficher_nom_matière
   FINTANTQUE
   fermer_curs_cours
FINRAFFINEMENT


RAFFINEMENT AFFICHER_CORPS
   selectionner_matricule_etudiant_sur_pv_par_curs_mat
   ouvrir_curs_mat
   cpt<-0
    TANTQUE curs_mat_non_vide REPETER
      select_une_du_curseur
      selectionner_notes
      afficher_notes
      ouvrir_curs_cours
      TANTQUE curs_cours_non_vide REPETER
          selectionner_coef_matière
          selectionner_moyenne_/_matière
          calculer_total
          afficher_notes
          afficher_total
      FINTANTQUE
      fermer_curs_cours
      selectionner_moyenne_generale,resultat,rang
      afficher_données_selectionnées
      cpt<-cpt+1
   FINTANTQUE
   fermer_curs_mat
FINRAFFINEMENT.

RAFFINEMENT BAS_DE_PAGE
   nb_admis<-select_compte_nb_etudiant_admis
   nb_ajournes<-select_compte_nb_etudiant_ajournés
   nb_absents<-select_compte_nb_etudiant_absents
   nb_présents<-cpt -nb_absents
   afficher_resultats
FINRAFFINEMENT
```

PROCES-VERBAL NON RENSEIGNE

```
 a)RESULTAT
   Edition des pv non renseignés des inscrits à un
    programme donné.
 b)DONNEES
   Base de données
   code_prog
 c)VARIABLES DE TRAVAIL
         (voir table des identificateurs)
 d)ENCHAINEMENT DES OPERATIONS
    code_prog<-Lire_code_programme
    SI not(CONNEXION) ALORS
       exit
    SINON
      SI TEST<0 ALORS
         code programme inexistant
      SINON
         effacer_ecran
         ouv_fich_imprimante
         Select_num_pv_correspdt_au_programme_par_curs_pv
         ouvrir_curs_pv
          TANTQUE curs_pv_non_vide REPETER
           select_un_num_pv_du_curseur
           ENTETE
           AFFICHER_CORPS
           BAS_DE_PAGE
         FINTANTQUE
         fermer_curs_pv
         fermer_fich_imprimante
      FINSI
    FINSI.
```

```
RAFFINEMENT ENTETE
   selectionner_nom_matière_par_curs_cours
   selection_autres_données_pour_entête
   afficher_autres_données_selectionnées
   ouvrir_curs_cours
   TANTQUE curs_cours_non_vide REPETER
       afficher_nom_matière
   FINTANTQUE
   fermer_curs_cours
FINRAFFINEMENT
```

```
RAFFINEMENT AFFICHER_CORPS
   selectionner_matricule_etudiant_sur_pv_par_curs_mat
   ouvrir_curs_mat
   cpt<-0
   TANTQUE curs_mat_non_vide REPETER
      select_identite_de_etudiant_du_curseur
      afficher_nom_prenom_matricule_selectionnés
      cpt<-cpt+1
   FINTANTQUE
   fermer_curs_mat
FINRAFFINEMENT.
```

CURSUS UNIVERSITAIRE

```
a)RESULTAT
    Edition du cursus universitaire d'un étudiant à partir de
     son numéro matricule
b)DONNEES
    Base de données
    mat
c)VARIABLES DE TRAVAIL
    (voir table des identificateurs)
 d)ENCHAINEMENT DES OPERATIONS
    mat<-lire_numéro_matricule
    SI not(CONNEXION) ALORS
       exit
    SINON
       effacer_ecran
     SI TEST <0 ALORS
           Matricule inexistant
       SINON
           SI TEST2 <0 ALORS
              Etudiant jamais inscrit à un programme
           SINON
              ouv_fich_imprimante
              ENTETE
              AFFICHER_CORPS
              BAS_DE_PAGE
              fermer_fich_imprimante
           FINSI
       FINSI
    FINSI.
```

```
RAFFINEMENT TEST
   select_etudiant_dont_matricule=mat
   SI non_trouvé ALORS
       retourné -1
   SINON
       retourné 1
   FINSI
FINRAFFINEMENT

RAFFINEMENT TEST2
   selectionner_numéros_pv_contenant_mat
   SI non_trouvé ALORS
      retourné -1
   SINON
      retourné 1
   FINSI
FINRAFFINEMENT


RAFFINEMENT ENTETE
   select_identite_etudiant
   afficher_entête
FINRAFFINEMENT


RAFFINEMENT AFFICHER_CORPS
   select_programmes_par_curs_cursus
   ouvrir_curs_cursus
   TANTQUE curs_cursus_non_vide REPETER
       select_un_programme_du_curseur
       select_autres_données
       afficher_données_selectionnées
   FINTANTQUE
   fermer_curs_cursus
FINRAFFINEMENT



RAFFINEMENT BAS_DE_PAGE
   selectionner_date_système
   afficher_date_du_jour
FINRAFFINEMENT
```

FICHE DIPLÔME

```
a)RESULTAT
  Edition d'une fiche diplôme contenant des informations
   necessaires à la calligraphie d'un diplôme et attestant
     le droit de l'étudiant à ce diplôme.
b)DONNEES
  Base de données
c)VARIABLES DE TRAVAIL
   (voir table des identificateurs)
d)ENCHAINEMENT DES OPERATIONS
  SI not(CONNEXION) ALORS
      exit
  SINON
     select_par_curs_prog_les_programmes_à_un_diplôme
     ouv_fich_imprimante
     ouvrir_curs_prog
     TANTQUE non_fin_curs_prog REPETER
         select_un_programme_du_curseur
         select_par_curs_etu_les_etudts_admis_au_progme
         ouvrir_curs_etu
      TANTQUE non_fin_curs_etu REPETER
             select_un_etudiant
             ENTETE
             AFFICHER_CORPS
             BAS_DE_PAGE
          FINTANTQUE
          fermer_curs_etu
     FINTANTQUE
     fermer_curs_prog
     fermer_fich_imprimante
  FINSI .


RAFFINEMENT ENTETE
   select-année
   afficher_entête_de_la_fiche
FINRAFFINEMENT


RAFFINEMENT AFFICHER_CORPS
   selectionner_informations_necessaires
   afficher_informations
FINRAFFINEMENT
```

```
RAFFINEMENT BAS_DE_PAGE
    selectionner_date_système
    afficher_date_courante
FINRAFFINEMENT
```

STATISTIQUES DES INSCRITS & ADMIS PAR ETABLISSEMENT, PAR PAYS, PAR NIVEAU D'ETUDE ET PAR SEXE.

a) RESULTAT
Edition d'état statistique par établissement en une années donnée.

b) DONNEES
Base de données
annee

c) VARIABLES DE TRAVAIL
(voir table des identificateurs)

d) ENCHAINEMENT DES OPERATIONS

```
SI not(CONNEXION) ALORS
   exit
SINON
   effacer_ecran
   ouv_fich_imprimante
   select_tous_les_etablissements_par_curs_etb
   ouvrir curs_etb
   TANTQUE annee=""  REPETER
      lire_annee
   FINTANTQUE
   TANTQUE non_fin_curs_etb REPETER
       select_un_etablissement_du_curseur
       ENTETE
       AFFICHER_CORPS
       BAS_DE_PAGE
   FINTANTQUE
   fermer_curs_etb
   fermer_fich_imprimante
FINSI.

RAFFINEMENT AFFICHER_CORPS
   select_tous_les_code_pays_par_curs_pays
   ouvrir_curs_pays
   TANTQUE non_fin_curs_pays REPETER
      select_un_code_pays_du_curseur
      select_nom_du_pays
      afficher_nom_pays
      select_niveaux_etude_par_curs_niveau
      ouvrir_curs_niveau
```

```
        TANTQUE non_fin_curs_niveau REPETER
            select_premier_niveau_contenu_dans_curseur
            compter_nb_garçons_inscrits
            compter_nb_garçons_admis
            compter_nb_filles_inscrites
            t_inscrits_pays<-nb_filles_inscrites
                            + nb_garçons_inscrits
            compter_nb_filles_admises
            t_admis_pays<-nb_filles_admises+nb_garçons_admis
            t_g_inscrits<-t_g_inscrits + nb_g_inscrits
            t_f_inscrites<-t_f_inscrites + nb_f_inscrites
            T_inscrits_pays<-T_inscrits_pays +t_inscrits_pays
            T_admis_pays<-T_admis_pays + t_admis_pays
            SI t_inscrits_pays<>0 ALORS
               p_admis_pays<-t_admis_pays/t_inscrits_pays
            FINSI
            afficher_resultats
        FINTANTQUE
        fermer_curs_niveau
    FINTANTQUE
    fermer_curs_pays
FINRAFFINEMENT


RAFFINEMENT BAS_DE_PAGE
     SI T_inscrits_pays<>0 ALORS
         p_T_admis_pays<-T_admis_pays/T_inscrits_pays
     FINSI
     afficher_resultats_totaux_col
FINRAFFINEMENT
```

ETAT DES FRAIS D'INSCRIPTION

a) RESULTAT
   Edition de l'état des frais d'inscription(étudiant non à jour) par établissement, département, avec rupture de sequences.
b) DONNEES
   Base de données
c) VARIABLES DE TRAVAIL
   (voir table des identificateurs)

```
d)ENCHAINEMENT DES OPERATIONS
  SI not(CONNEXION) ALORS
     exit
  SINON
     effacer_ecran
     ouv_fich_imprimante
     ENTETE
     AFFICHER_CORPS
     BAS_DE_PAGE_GENERAL
     fermer_fich_imprimante
  FINSI.


 RAFFINEMENT AFFICHER_CORPS
      select_tous_les_etablissements_par_curs_etb
      ouvrir_curs_etb
      TANTQUE non_fin_curs_etb REPETER
         initialiser_var
         select_un_etablissement_du_curseur
         select_tous_les_dépmts_de_l'étb_par_curs_dep
         ouvrir_curs_dep
         TANTQUE non_fin_curs_dep REPETER
             initialiser_var
             select_un_dep_du_curseur
             select_Mle_etudt_non_à_jour_par_curs_frais
             ouvrir_curs_frais
             ENTETE2
             test=1
             TANTQUE non_fin_curs_frais REPETER
                  select_autres_données
                  calculer_reste_à_payer
                  afficher_données_selectionnées
             FINTANTQUE
             fermer_curs_frais
             BAS_DE_PAGE_DEP
             te_mt_du<-te_mt_du + td_mt_du
             te_mt_paye<-te_mt_paye + td_mt_paye
         FINTANTQUE
         fermer_curs_dep
         tg_mt_du<-tg_mt_du + te_mt_du
         tg_mt_paye<-tg_mt_paye + te_mt_paye
         SI test=0 ALORS
            ENTETE2
            BAS_DE_PAGE_dep
         FINSI
         BAS_DE_PAGE_etb
      FINTANTQUE
      fermer_curs_etb
 FINRAFFINEMENT
```

```
RAFFINEMENT BAS_DE_PAGE_dep
     td_mt_reste<-td_mt_du - td_mt_paye
     afficher_totaux_dep
FINRAFFINEMENT


RAFFINEMENT BAS_DE_PAGE_etb
     te_mt_reste<-te_mt_du - te_mt_paye
     afficher_totaux_etb
FINRAFFINEMENT


RAFFINEMENT BAS_DE_general
     tg_mt_reste<-tg_mt_du - tg_mt_paye
     afficher_totaux_generaux
FINRAFFINEMENT
```

LISTE DES ADMIS A UN CONCOURS,UN EXAMEN SPECIAL OU A UNE ETUDE DE DOSSIERS

a) RESULTAT
Edition des résultats(admis) de concours, examen spécial ou pour étude de dossiers.

b) DONNEES
Base de données
code_prog

c) VARIABLES DE TRAVAIL
(voir table des identificateurs)

d) ENCHAINEMENT DES OPERATIONS

```
SI not(CONNEXION) ALORS
   exit
SINON
   effacer_ecran
   ouv_fich_imprimante
   TANTQUE choix <> '4' REPETER
      cpt<-0
      AFFICHER_MENU
       lire_menu
```

```
        DECIDER ENTRE
          cas choix=1:AFFICHER_MENU2
                      lire_code_prog
                      CONCOURS
          cas choix=2:AFFICHER_MENU2
                      lire_code_prog
                      EXAMEN
          cas choix=3:AFFICHER_MENU2
                      lire_code_prog
                      DOSSIER
          cas choix=4:quitter
        FINDECIDER
      BAS_DE_PAGE
  FINTANTQUE
FINSI.



RAFFINEMENT AFFICHER_MENU
   effacer_ecran
   afficher"1.PAR CONCOURS"
   afficher"2.PAR EXAMEN SPECIAL"
   afficher"3.PAR DOSSIER"
   afficher"4.ABANDON"
   afficher"Choix"
   lire_type_accès
FINRAFFINEMENT


RAFFINEMENT AFFICHER_MENU2
   effacer_ecran
   afficher"  CODE DU PROGRAMME"
   lire_code_prog
FINRAFFINEMENT


RAFFINEMENT CONCOURS
   selection_libellé_programme
   ENTETE
   AFFICHER_CORPS1
FINRAFFINEMENT


RAFFINEMENT EXAMEN
   selection_libellé_programme
   ENTETE
   AFFICHER_CORPS2
FINRAFFINEMENT
```

```
RAFFINEMENT DOSSIER
    selection_libellé_programme
    ENTETE
    AFFICHER_CORPS3
FINRAFFINEMENT


RAFFINEMENT AFFICHER_CORPS1
  select_identités_étudiants_par_curs_candidat1
         (étudiants reçus par concours)
  ouvrir_curs_candidat1
  TANTQUE non_fin_curs_candidat1 REPETER
      select_identité_d'un_étudiant_du_curseur
      afficher_données_selectionnées
      cpt<-cpt + 1
  FINTANTQUE
  fermer_curs_candidat1
FINRAFFINEMENT


 RAFFINEMENT AFFICHER_CORPS2
  select_identités_étudiants_par_curs_candidat2
         (étudiants reçus par concours)
  ouvrir_curs_candidat2
  TANTQUE non_fin_curs_candidat2 REPETER
      select_identité_d'un_étudiant_du_curseur
      afficher_données_selectionnées
      cpt<-cpt + 1
  FINTANTQUE
  fermer_curs_candidat2
FINRAFFINEMENT

RAFFINEMENT AFFICHER_CORPS3
  select_identités_étudiants_par_curs_candidat3
         (étudiants reçus par concours)
  ouvrir_curs_candidat3
  TANTQUE non_fin_curs_candidat3 REPETER
      select_identité_d'un_étudiant_du_curseur
      afficher_données_selectionnées
      cpt<-cpt + 1
  FINTANTQUE
  fermer_curs_candidat3
FINRAFFINEMENT


RAFFINEMENT BAS_DE_PAGE
   selectionner_date_système
   afficher_date_courante_et_information_de_bas_de_page
FINRAFFINEMENT
```

LISTING DES PROGRAMMES

I.attestation

```
#include<stdio.h>

/*----------------declaration des variables hotes-------*/
EXEC SQL BEGIN DECLARE SECTION;
  VARCHAR username[20];
  VARCHAR password[20];
  char c;
  VARCHAR h_num_pv[7];
  VARCHAR nom_etudiant[26];
  VARCHAR prenom_etudiant[41];
  VARCHAR  date_n[9];
  VARCHAR lieu_n[21];
  VARCHAR mat[8];
  VARCHAR code[6];
  VARCHAR mention[3];
  VARCHAR date_signe_pv[9];
  VARCHAR code_programme[6];
  VARCHAR code_cycle[2];
  VARCHAR code_option[4];
  VARCHAR niveau[3];
  VARCHAR code_filiere[7];
  VARCHAR lib_cycle[16];
  VARCHAR sigle_filiere[11];
  VARCHAR lib_filiere[41];
  VARCHAR intitule_diplome[41];
  VARCHAR lib_option[31];
  VARCHAR date_courante[9];
  VARCHAR h_sexe[2];
EXEC SQL END DECLARE SECTION;
  FILE *fp;
  char code_prog[6];
EXEC SQL INCLUDE SQLCA;

/*************************************************************/
/************DECLARATION DES  PROCEDURES******************/
/*************************************************************/

int CONNEXION();

int TEST();

void ERREUR2();

void ENTETE();

void AFFICHER_CORPS();
```

```
void BAS_DE_PAGE();

void NOMORE();


/*********************************************************/
/*--------------PROCEDURE PRINCIPALE--------------------*/
/*********************************************************/

main()

{
if (CONNEXION() < 0)
   exit(-1);
else
{
  system("clear");
  printf("\n\n\n\n\n\n\n\n\n\n\n");
  printf("                CODE DU PROGRAMME: ");
  scanf("%s",&code_prog);
  strcpy(code_programme.arr,code_prog);
  code_programme.len=strlen(code_programme.arr);
  if (TEST() < 0)
    {
      printf ("AUCUN ETUDIANT N'A DROIT A UNE ATTESTATION");
      scanf("%c",&c);
    }
  else
  {
   fp=fopen("/dev/lp0","w");
   EXEC SQL DECLARE curs_prog CURSOR FOR
                SELECT num_pv
                FROM pv
                WHERE code_programme=:code_programme
                ORDER BY num_pv;
  EXEC SQL OPEN curs_prog;
  EXEC SQL WHENEVER NOT FOUND GOTO fin_curs_prog;
  do
  {
    EXEC SQL FETCH curs_prog INTO:h_num_pv;

    EXEC SQL DECLARE curs_etudiant CURSOR FOR
               SELECT matricule,mention
               FROM INS_PROG
               WHERE (resultat_obt = 'AD') AND (num_pv=:h_num_pv)
                  ORDER BY  matricule;
    EXEC SQL OPEN curs_etudiant;
    EXEC SQL WHENEVER NOT FOUND GOTO fin_curs_etudiant;
    do
    {
      EXEC SQL FETCH curs_etudiant INTO :mat,
                           :mention;
```

```
      ENTETE();
      AFFICHER_CORPS();
      BAS_DE_PAGE();
    }while(1);
    fin_curs_etudiant:EXEC SQL CLOSE curs_etudiant;
    EXEC SQL WHENEVER NOT FOUND CONTINUE;
  }while(1);
  fin_curs_prog:EXEC SQL CLOSE curs_prog;
  fclose(fp);
 }
scanf("%c",&c);

}
}


/*************************************************************/
/*******************CORPS DES PROCEDURES**********************/
/*************************************************************/


/*************************************************************/
/*------------PROCEDURE DE CONNECTION ORACLE-------------*/
/*************************************************************/

int CONNEXION()

{
  strcpy(username.arr,"system");
  username.len=strlen(username.arr);
  strcpy(password.arr,"manager");
  password.len=strlen(password.arr);
  EXEC SQL WHENEVER SQLERROR goto erreur;
  EXEC SQL CONNECT :username IDENTIFIED BY :password;
  return 0;
  erreur:printf("error on connection\n");
         return (-1);
}


/*************************************************************/
/*-----------------PROCEDURE ERREUR2--------------------*/
/*************************************************************/

void ERREUR2()

{
   printf("erreur");
   exit(0);
}


/*************************************************************/
/*-----------------PROCEDURE ENTETE---------------------*/
/*************************************************************/
```

```
 void ENTETE()

{
  fprintf(fp,"\n\n\n\n\n\n\n");
  fprintf(fp,"%s","          MINISTERE");
  fprintf(fp,"%+50s","BURKINA FASO \n");
  fprintf(fp,"%s","  DES ENSEIGNEMENTS SECONDAIRE,");
  fprintf(fp,"%+38s","------------ \n");
  fprintf(fp,"%s","  SUPERIEUR ET DE LA RECHERCHE");
  fprintf(fp,"%+45s","La Patrie ou la Mort,\n");
  fprintf(fp,"%s","          SCIENTIFIQUE ");
  fprintf(fp,"%+48s","    Nous Vaincrons\n");
  fprintf(fp,"%s","           ---------\n ");
  fprintf(fp,"%s"," UNIVERSITE DE OUAGADOUGOU \n");
  fprintf(fp,"%s","      ---------------\n");
  fprintf(fp,"%s","  No -----------------");
  fprintf(fp,"%s","/UO/SG/SGS\n\n\n\n\n\n\n");
  fprintf(fp,"%+45s", "ATTESTATION\n");
  fprintf(fp,"%+45s","   -------  \n");
  fprintf(fp,"%s","\n\n                    Le Secretaire General de");
  fprintf(fp,"%s","  l'UNIVERSITE DE OUAGADOUGOU,");
  fprintf(fp,"%s","\n\n   atteste que  M. ");
}


/*********************************************************/
/*--------------PROCEDURE NOMORE-----------------------*/
/*********************************************************/


void NOMORE()

{
  fprintf(fp," \n");
  exit(0);
}

/*********************************************************/
/*------------------PROCEDURE AFFICHER_CORPS------------*/
/*********************************************************/

void AFFICHER_CORPS()

{
   EXEC SQL WHENEVER SQLERROR DO ERREUR2();
   EXEC SQL SELECT  nom,prenom,TO_CHAR(date_naiss,'DD-MM-YY'),
               lieu_naiss
            INTO    :nom_etudiant,
                 :prenom_etudiant,
                 :date_n,
                 :lieu_n
         FROM    candidat
         WHERE   code_candidat IN (SELECT code_candidat
                           FROM etudiant
                          WHERE matricule = :mat);
```

```
  EXEC SQL SELECT TO_CHAR( date_signe_pv,'DD-MM-YY'),
                        code_programme
                  INTO  :date_signe_pv,
                        :code_programme
                  FROM   pv
                 WHERE   num_pv =:h_num_pv;
     EXEC SQL SELECT  code_cycle,code_option,niveau,code_filiere
              INTO  :code_cycle,
                    :code_option,
                    :niveau,
                    :code_filiere
                 FROM  programme
               WHERE  code_programme = :code_programme;
     EXEC SQL SELECT  lib_cycle
              INTO  lib_cycle
              FROM  cycle
                WHERE  code_cycle = :code_cycle;
     EXEC SQL SELECT  sigle_filiere,lib_filiere,intitule_diplome
              INTO  :sigle_filiere,
                    :lib_filiere,
                    :intitule_diplome
                 FROM  filiere
               WHERE  code_filiere = : code_filiere;
     EXEC SQL SELECT  lib_option
              INTO  :lib_option
              FROM  Options
             WHERE  code_option = :code_option;
     EXEC SQL  CLOSE  curs_etudiant;


/*****REMPLISSAGE DES CASES CORRESPONDANT AUX INFO
RECUPEREES******-*/

     nom_etudiant.arr[nom_etudiant.len]='\0';
     prenom_etudiant.arr[prenom_etudiant.len]='\0';
     fprintf(fp,"%s%s%s",nom_etudiant.arr,"
",prenom_etudiant.arr);
     fprintf(fp,"%s","\n\n   ne(e) le ");
     fprintf(fp,"%s",date_n.arr);
     fprintf(fp,"%s","   a   ");
     fprintf(fp,"%s",lieu_n.arr);
     fprintf(fp,"%s","\n\n   matricule");
     fprintf(fp,"%s%s","  ",mat.arr);
     fprintf(fp,"%s","\n\n   a satisfait le   ");
     fprintf(fp,"%s",date_signe_pv.arr);
     fprintf(fp,"\n\n");
     fprintf(fp,"%s","   aux epreuves sanctionnant la  ");
     fprintf(fp,"%s",niveau.arr);
     fprintf(fp,"%s","  annee du  ");
     fprintf(fp,"%s",lib_cycle.arr);
     fprintf(fp,"%s"," cycle");
     fprintf(fp,"%s","\n\n   de               ");
     fprintf(fp,"%s",lib_filiere.arr);
```

```
      fprintf(fp,"%s","\n\n   et a ainsi obtenu le  ");
      fprintf(fp,"%s",intitule_diplome.arr);
      fprintf(fp,"\n\n\n");
      fprintf(fp,"%+54s","Mention:");
      fprintf(fp,"%s",mention.arr);
      fprintf(fp,"\n\n");
      fprintf(fp,"%+53s","Option:");
      fprintf(fp,"%s",lib_option.arr);
      fprintf(fp,"\n\n");
      fprintf(fp,"%+59s","Appreciation:");
      fprintf(fp,"%s","\n\n\n   En fois de quoi, il lui delivre");
      fprintf(fp,"%s"," la presente attestation pour ");
      fprintf(fp,"%s","\n\n   servir et valoir ce que de
droit.\n\n\n");
}


/*******************************************************/
/*---------------PROCEDURE BAS_DE_PAGE-----------------*/
/*******************************************************/

void BAS_DE_PAGE()

{
  fprintf(fp,"\n\n");

  EXEC SQL SELECT TO_CHAR (sysdate,'DD-MM-YY')
           INTO :date_courante
           FROM dual;

  fprintf(fp,"%+55s","OUAGADOUGOU, le ");
  fprintf(fp,"%s",date_courante.arr);
  fprintf (fp,"\n\n");
  fprintf(fp,"%+60s","Le Secretaire general");
  fprintf(fp,"\n\n\n\n\n\n\n\f");
}

/*******************************************************/
/*---------------PROCEDURE TEST------------------------*/
/*******************************************************/
/*Cette procedure permet de tester s'il y a au moins un */
/*etudiant admis ;auquel cas on peut lancer le programme*/
/*autrement le programme ne peut etre execute.----------*/


int TEST()

{
  EXEC SQL WHENEVER NOT FOUND goto non_trouve;
```

```
  EXEC SQL SELECT matricule
          INTO :mat
             FROM INS_PROG
            WHERE (resultat_obt = 'AD') AND (num_pv IN
          (SELECT num_pv
          FROM pv
          WHERE code_programme=:code_programme));
  return(0);
  non_trouve:return(-1);
}
```

BORDEREAU

```
#include "/home/daas/prodaas/daas_fonctions/fonc.c"
#include<stdio.h>

/*----------------declaration des variables hotes-------*/
EXEC SQL BEGIN DECLARE SECTION;
  VARCHAR username[20];
  VARCHAR password[20];
  char c;
  VARCHAR h_num_pv[7];
  VARCHAR nom_etudiant[26];
  VARCHAR prenom_etudiant[41];
  VARCHAR mat[8];
  VARCHAR date_signe_pv[9];
  VARCHAR code_programme[6];
  VARCHAR code_cycle[2];
  VARCHAR niveau[3];
  VARCHAR code_filiere[7];
  VARCHAR lib_cycle[16];
  VARCHAR sigle_filiere[11];
  VARCHAR lib_filiere[41];
EXEC SQL END DECLARE SECTION;
  FILE *fp;
  int cpt;
  char code_prog[6];
EXEC SQL INCLUDE SQLCA;
```

```
/*************************************************************/
/*************DECLARATION DES  PROCEDURES*********************/
/*************************************************************/
int TEST();

int TEST2();

int CONNEXION();

void ERREUR2();

void AFFICHER_CORPS();

void ENTETE();

void NOMORE();


/*************************************************************/
/*---------------PROCEDURE PRINCIPALE---------------------*/
/*************************************************************/


main()
{
if (CONNEXION() < 0)
   exit(-1);
else
{
  system("clear");
  printf("\n\n\n\n\n\n\n\n\n\n\n");
  printf("                CODE DU PROGRAMME: ");
  scanf("%s",&code_prog);
  strcpy(code_programme.arr,code_prog);
  code_programme.len=strlen(code_programme.arr);
  if (TEST() < 0)
    {
      printf ("\n\n\n  AUCUN ETUDIANT N'EST ADMIS A CE
PROGRAMME\n\n");
    }
  else
  {
   fp=fopen("/dev/lp0","w");
   system("clear");
   EXEC SQL DECLARE curs_prog CURSOR FOR
                SELECT num_pv
                FROM pv
                WHERE code_programme=:code_programme
                ORDER BY num_pv;
  EXEC SQL OPEN curs_prog;
```

```
   EXEC SQL WHENEVER NOT FOUND GOTO fin_curs_prog;
   do
   {
     EXEC SQL FETCH curs_prog INTO:h_num_pv;

     EXEC SQL DECLARE curs_nom_etudiant CURSOR FOR
              SELECT DISTINCT candidat.nom,candidat.prenom,matricule
            FROM    candidat,etudiant
            WHERE  (candidat.code_candidat=etudiant.code_candidat)
            AND    (candidat.code_candidat
            IN (SELECT code_candidat
                  FROM etudiant
                 WHERE matricule
              IN
               (SELECT DISTINCT matricule
               FROM ins_prog
               WHERE (resultat_obt = 'AD') AND (num_pv
=:h_num_pv))))
                   ORDER BY matricule;

     EXEC SQL OPEN curs_nom_etudiant;
     if(TEST2() > 0)
     {
        ENTETE();
        AFFICHER_CORPS();
        BAS_DE_PAGE();
     }
    EXEC SQL WHENEVER NOT FOUND CONTINUE;
   }while(1);
   fin_curs_prog:EXEC SQL CLOSE curs_prog;
   fclose(fp);
  }
scanf("%c",&c);
}
}

/*****************************************************************/
/*******************CORPS DES PROCEDURES**********************/
/*****************************************************************/


/*****************************************************************/
/*------------PROCEDURE DE CONNECTION ORACLE-------------*/
/*****************************************************************/

int CONNEXION()

{
  strcpy(username.arr,"system");
  username.len=strlen(username.arr);
  strcpy(password.arr,"manager");
  password.len=strlen(password.arr);
  EXEC SQL WHENEVER SQLERROR goto erreur;
```

```
  EXEC SQL CONNECT :username IDENTIFIED BY :password;
  return 0;
  erreur:printf("error on connection\n");
         return (-1);
}


/*********************************************************/
/*-----------------PROCEDURE ERREUR2--------------------*/
/*********************************************************/

void ERREUR2()

{
   printf("erreur");
   exit(0);
}

/*********************************************************/
/*-------------------PROCEDURE NOMORE--------------------*/
/*********************************************************/

void NOMORE()

{
  fprintf(fp,"\n");
  exit(0);
}




/*********************************************************/
/*-----------------PROCEDURE ENTETE----------------------*/
/*********************************************************/

void ENTETE()


{

/*declaration de deux curseur pour recuperer des info dans la
base*/
/*Mais pour tester le programme une valeur arbitraire est affectee
-*/
/*-----------------------a cette
variable---------------------------*/

/* utilisation d'une variable VARCHAR num_pv qui doit etre
---------*/

/*initialisee lors de la selection du
programme.--------------------*/

  EXEC SQL WHENEVER SQLERROR DO ERREUR2();
```

```
 EXEC SQL SELECT TO_CHAR(date_signe_pv,'DD-MM-YY'),
            code_programme
            INTO  :date_signe_pv,
            :code_programme
            FROM   pv
          WHERE num_pv =:h_num_pv;
  EXEC SQL SELECT code_cycle,niveau,code_filiere
          INTO  :code_cycle,
             :niveau,
             :code_filiere
             FROM programme
          WHERE code_programme =:code_programme;
  EXEC SQL SELECT lib_cycle
          INTO :lib_cycle
          FROM cycle
          WHERE code_cycle =:code_cycle;
  EXEC SQL SELECT lib_filiere
          INTO  :lib_filiere
          FROM filiere
          WHERE code_filiere =:code_filiere;

  fprintf(fp,"\n\n\n\n\n");
  fprintf(fp,"%+75s","      BORDEREAU  DES  ATTESTATIONS");
  fprintf(fp,"\n");
  fprintf(fp,"%+75s","      ----------------------------");
  fprintf(fp,"\n\n\n");
  fprintf(fp,"%s%s","  Proces-Verbal no ", h_num_pv.arr);
  fprintf(fp,"%s"," du ");
  fprintf(fp,"%s",date_signe_pv.arr);
  fprintf(fp,"\n\n");
  fprintf(fp,"%s%s","  ",niveau.arr);
  fprintf(fp,"%s","  annee ");
  fprintf(fp,"%s",lib_cycle.arr);
  fprintf(fp,"%s"," cycle ");
  fprintf(fp,"%s%s"," de ",lib_filiere.arr);
  fprintf(fp,"%s","\n\n\n|");

fprintf(fp,"%s","-------------------------------------------------"
);

fprintf(fp,"%s","-------------------------------------------------
");
  fprintf(fp,"%s","-------------------------");
  fprintf(fp,"%s","\n|");
  fprintf(fp,"%s", " No ordre  |");
  fprintf(fp,"%+10s"," Matricule  |");
  fprintf(fp,"%+30s","  Nom(s)    |");
  fprintf(fp,"%+35s","  Prenom(s)  |");
  fprintf(fp,"%+10s"," Date retrait  |");
  fprintf(fp,"%s", "     signature ");
  fprintf(fp,"\n|");

fprintf(fp,"%s","-------------------------------------------------"
);
```

```
fprintf(fp,"%s","-----------------------------------------------------
");
  fprintf(fp,"%s","--------------------------");
  fprintf(fp,"%s","\n|");
}


/*****************************************************************/
/*------------------PROCEDURE AFFICHER_CORPS-------------------*/
/*****************************************************************/

void AFFICHER_CORPS()
{
  cpt=1;
  EXEC SQL WHENEVER NOT FOUND GOTO fin_curs_etudiant;
  do
  {
    EXEC SQL FETCH curs_nom_etudiant INTO :nom_etudiant,
                                :prenom_etudiant,
                           :mat;
    nom_etudiant.arr[nom_etudiant.len]='\0';
    fprintf(fp,"\n");
    fprintf(fp,"%s%d%s","|     ",cpt,"     |");
    fprintf(fp,"%+10s%s",mat.arr,"  |");
    fprintf(fp,"%-28s%s",nom_etudiant.arr," |");
    fprintf(fp,"%-33s%s",prenom_etudiant.arr," |");
    fprintf(fp,"%+10s","              |");
    fprintf(fp,"%s", "             ");
    cpt=cpt++;
 }while(1);
 fin_curs_etudiant:EXEC SQL CLOSE curs_nom_etudiant;
}



BAS_DE_PAGE()
{

fprintf(fp,"%s","\n|-----------------------------------------------
");

fprintf(fp,"%s","-----------------------------------------------------
");
  fprintf(fp,"%s","-------------------------|\n\n\n\n\n\n\n");
}

int TEST()

{
  EXEC SQL WHENEVER NOT FOUND goto non_trouve;
```

```
  EXEC SQL SELECT matricule
          INTO :mat
             FROM INS_PROG
            WHERE (resultat_obt = 'AD')
          AND (num_pv IN (SELECT num_pv
                    FROM pv
                    WHERE code_programme=:code_programme));
  return(1);
  non_trouve:return(-1);
}


int TEST2()

{
  EXEC SQL WHENEVER NOT FOUND goto non_trouve;
  EXEC SQL SELECT matricule
          INTO :mat
             FROM INS_PROG
            WHERE (resultat_obt = 'AD') AND (num_pv=:h_num_pv);
  return(1);
  non_trouve:return(-1);
}
```

CURSUS UNIVERSITAIRE

```
#include<stdio.h>
/*------------------DECLARATION DES VARIABLES-----------------*/
EXEC SQL BEGIN DECLARE SECTION;
  VARCHAR username[20];
  VARCHAR password[20];
  VARCHAR h_mat[8];
  VARCHAR h_date_courante[9];
  VARCHAR h_nom_etudiant[26];
  VARCHAR h_prenom_etudiant[41];
  VARCHAR h_code_candidat[6];
  VARCHAR h_code_prog[6];
  VARCHAR h_niveau[3];
  VARCHAR h_annee[9];
  VARCHAR h_sigle_dep[11];
  VARCHAR h_sigle_etab[11];
  VARCHAR h_lib_prog[31];
  VARCHAR h_resultat[3];
  VARCHAR h_mention[3];
  VARCHAR h_ses_sion[2];
  VARCHAR h_num_pv[7];
EXEC SQL END DECLARE SECTION;
  FILE *fp;
  char mat[8];
EXEC SQL INCLUDE SQLCA;
```

```
/*****************************************************************/
/***************DECLARATION DES PROCEDURES********************/
/*****************************************************************/
int TEST();
int TEST2();
void ENTETE();
int CONNEXION();
void AFFICHER_CORPS();
void BAS_DE_PAGE();
void erreur();


/*****************************************************************/
/****************PROCEDURE PRINCIPALE*************************/
/*****************************************************************/
main()
{
char c;
system("clear");
printf("\n\n\n\n\n\n\n\n\n\n\n");
printf("                MATRICULE: ");
scanf("%s",&mat);
strcpy(h_mat.arr,mat);

h_mat.len=strlen(h_mat.arr);
if (CONNEXION() < 0)
    exit(-1);
else
{
    if (TEST() < 0)
    {
       printf("\n\n        MATRICULE INEXISTANT \n\n");
    }
    else
    {
    if(TEST2() < 0)
    {
      printf("\n\n     ETUDIANT JAMAIS INSCRIT A UN
PROGRAMME\n\n");
    }
    else
    {
       fp=fopen("/dev/lp0","w");
       ENTETE();
       AFFICHER_CORPS();
       BAS_DE_PAGE();
    }
    }
}
}
```

```
/*************************************************************/
/***************CORPS DES PROCEDURES**************************/
/*************************************************************/


/*************************************************************/
/****************PROCEDURE DE CONNEXION ORACLE***************/
/*************************************************************/

int CONNEXION()
{
  strcpy(username.arr,"system");
  strcpy(password.arr,"manager");
  username.len=strlen(username.arr);
  password.len=strlen(password.arr);
  EXEC SQL WHENEVER SQLERROR DO erreur();
  EXEC SQL CONNECT :username IDENTIFIED BY :password;
  return 0;
}

/*************************************************************/
/*****************PROCEDURE ERREUR****************************/
/*************************************************************/

void erreur()
{
  printf("%s%s","erreur sql:",sqlca.sqlerrm.sqlerrmc);
  exit(0);
}


/*************************************************************/
/*********************PROCEDURE ENTETE************************/
/*************************************************************/

void ENTETE()
{
fprintf(fp,"%s","\n\n\n\n\n   UNIVERSITE DE OUAGADOUGOU");
fprintf(fp,"%s","\n   DIRECTION DES AFFAIRES ACADEMIQUES ET
SCOLAIRES");
fprintf(fp,"\n\n\n\n");
fprintf(fp,"%s","                              CURSUS UNIVERSITAIRE");
fprintf(fp,"\n\n\n");
EXEC SQL WHENEVER SQLERROR DO erreur();
EXEC SQL SELECT nom,prenom
          INTO :h_nom_etudiant,
               :h_prenom_etudiant
          FROM candidat
       WHERE code_candidat IN (SELECT code_candidat
                     FROM etudiant
                              WHERE matricule=:h_mat);
```

```
fprintf(fp,"%s%s"," NOM:",h_nom_etudiant.arr );
fprintf(fp,"\n");
fprintf(fp,"%s%s"," PRENOM:",h_prenom_etudiant.arr);
fprintf(fp,"\n");
fprintf(fp,"%s%s"," MATRICULE:",h_mat.arr);
fprintf(fp,"\n");
fprintf(fp,"%s","|-----------------------------------------------"
);
fprintf(fp,"%s","--------------------|");
fprintf(fp,"\n");
fprintf(fp,"%s","|ANNEE");
fprintf(fp,"%s","|S");
fprintf(fp,"%s","| ETB  ");
fprintf(fp,"%s","|DEPT    ");
fprintf(fp,"%s","|N");
fprintf(fp,"%s","| LIBELLE PROGRAMME            ");
fprintf(fp,"%s","|RES");
fprintf(fp,"%s","|MENTION|");
fprintf(fp,"\n");
fprintf(fp,"%s","|-----------------------------------------------"
);
fprintf(fp,"%s","--------------------|");

}


/*****************************************************************/
/****************PROCEDURE AFFICHER_CORPS*************************/
/*****************************************************************/

void AFFICHER_CORPS()
{
char annee[6];
char j;
EXEC SQL DECLARE curs_cursus CURSOR FOR
       SELECT num_pv,resultat_obt,mention
       FROM ins_prog
       WHERE matricule=:h_mat;
EXEC SQL OPEN curs_cursus;
EXEC SQL WHENEVER NOT FOUND GOTO fin_curs_cursus;
do
{
EXEC SQL FETCH curs_cursus INTO :h_num_pv,
                                :h_resultat,
                                :h_mention;
EXEC SQL SELECT annee,ses_sion,code_programme
         INTO :h_annee,
           :h_ses_sion,
           :h_code_prog
         FROM pv
      WHERE num_pv=:h_num_pv;
```

```
EXEC SQL SELECT libelle_programme,
          sigle_departement,
          sigle_etablessement,
          niveau
          INTO :h_lib_prog,
            :h_sigle_dep,
            :h_sigle_etab,
            :h_niveau
          FROM programme
        WHERE code_programme=:h_code_prog;
annee[0]=h_annee.arr[0];
annee[1]=h_annee.arr[1];
annee[2]='-';
annee[3]=h_annee.arr[2];
annee[4]=h_annee.arr[3];
annee[5]='\0';
h_lib_prog.arr[h_lib_prog.len]='\0';
fprintf(fp,"\n");
fprintf(fp,"%s%+5s","|",annee);
fprintf(fp,"%s%s","|", h_ses_sion.arr );
fprintf(fp,"%s%+6s","|", h_sigle_etab.arr);
fprintf(fp,"%s%s","| ",h_sigle_dep.arr);
fprintf(fp,"%s%s","|",h_niveau.arr );
fprintf(fp,"%s%+30s","|",h_lib_prog.arr );
fprintf(fp,"%s%s","|", h_resultat.arr );
fprintf(fp,"%s%+5s%s","  |",h_mention.arr ,"  |");
fprintf(fp,"\n");
fprintf(fp,"%s","|-----------------------------------------------
--");
fprintf(fp,"%s","-----------------|");
}while(1);
fin_curs_cursus: EXEC SQL CLOSE curs_cursus;
}


/*****************************************************************/
/******************PROCEDURE BAS_DE_PAGE**************************/
/*****************************************************************/

void BAS_DE_PAGE()
{
EXEC SQL WHENEVER NOT FOUND DO erreur();
EXEC SQL SELECT TO_CHAR(sysdate,'DD/MM/YY')
        INTO :h_date_courante
        FROM dual;
fprintf(fp,"\n\n\n\n");
fprintf(fp,"%s%s","                OUAGADOUGOU LE
",h_date_courante.arr);
fprintf(fp,"\n\n\n                                        LE RECTEUR");
fprintf(fp,"\n\n\n\n\n\n\n\n");
fprintf(fp,"%s","
");
fprintf(fp,"%s","                                  ");
}
```

```
int TEST()
{
EXEC SQL WHENEVER NOT FOUND GOTO nontrouve;
EXEC SQL SELECT code_candidat
       INTO:h_code_candidat
       FROM etudiant
       WHERE matricule=:h_mat;
return(1);
nontrouve:return(-1);
}

int TEST2()
{
EXEC SQL WHENEVER NOT FOUND GOTO non_inscrit;
EXEC SQL SELECT num_pv
       INTO:h_num_pv
          FROM ins_prog
       WHERE matricule=:h_mat;
return(1);
non_inscrit:return(-1);
}
```

INFORMATIONS POUR CALLIGRAPHIE DE DIPLÖMES

```
#include<stdio.h>

/*----------------declaration des variables hotes-------*/
EXEC SQL BEGIN DECLARE SECTION;
  VARCHAR username[20];
  VARCHAR password[20];
  VARCHAR h_num_pv[7];
  VARCHAR h_nom_etudiant[26];
  VARCHAR h_prenom_etudiant[41];
  VARCHAR h_code_filiere[7];
  VARCHAR h_lib_fil[41];
  VARCHAR h_mat[8];
  VARCHAR h_code_etb[3];
  VARCHAR h_lib_etb[41];
  VARCHAR h_code_prog[6];
  VARCHAR h_lib_prog[31];
  VARCHAR h_code_dep[4];
  VARCHAR h_lib_dep[41];
  VARCHAR h_code_option[4];
  VARCHAR h_date_naiss[9];
  VARCHAR h_lieu_naiss[21];
  VARCHAR h_date_creat_pv[9];
  VARCHAR h_sexe[2];
  VARCHAR h_niveau[3];
  VARCHAR h_lib_option[31];
  VARCHAR h_annee[6];
```

```
  VARCHAR h_session[2];
  VARCHAR h_date_courante[11];
EXEC SQL END DECLARE SECTION;
  FILE *fp;
EXEC SQL INCLUDE SQLCA;

/*************************************************************/
/*************DECLARATION DES  PROCEDURES*********************/
/*************************************************************/
void BAS_DE_PAGE();

int CONNEXION();


void AFFICHER_CORPS();

void ENTETE();

void BAS_DE_PAGE();

void NOMORE();
int errreur();


/*************************************************************/
/*--------------PROCEDURE PRINCIPALE--------------------*/
/*************************************************************/
main()
{
if (CONNEXION() < 0)
   exit(-1);
else
{
    EXEC SQL DECLARE curs_prog CURSOR FOR
                  SELECT code_programme,libelle_programme,
                      code_option,
                      niveau,code_filiere,
                      code_departement,code_etablissement
                     FROM programme
                  WHERE donne_dip='o';
    EXEC SQL OPEN curs_prog;
    fp=fopen("/dev/lp0","w");
    EXEC SQL WHENEVER NOT FOUND GOTO fin_curs;
    do
    {
    EXEC SQL FETCH curs_prog INTO:h_code_prog,
                    :h_lib_prog,
                             :h_code_option,
                    :h_niveau,
                    :h_code_filiere,
                    :h_code_dep,
                    :h_code_etb;
```

```
  EXEC SQL SELECT num_pv,ses_sion,date_creation_pv
          INTO:h_num_pv,
               :h_session,
               :h_date_creat_pv
           FROM pv
           WHERE code_programme=:h_code_prog;
    EXEC SQL DECLARE curs_etu CURSOR FOR
                 SELECT matricule
                 FROM ins_prog
                 WHERE num_pv=:h_num_pv
                 AND resultat_obt='AD';
    EXEC SQL OPEN curs_etu;
    EXEC SQL WHENEVER NOT FOUND GOTO fin_curs_etu;
      do
      {
         EXEC SQL FETCH curs_etu INTO:h_mat;
         system("clear");
         ENTETE();
         AFFICHER_CORPS();
         BAS_DE_PAGE();
      }while(1);
      fin_curs_etu:EXEC SQL CLOSE curs_etu;
    }while(1);
    fin_curs:EXEC SQL CLOSE curs_prog;
    EXEC SQL WHENEVER NOT FOUND CONTINUE;
    fclose(fp);
}
}

/*************************************************************/
/*******************CORPS DES PROCEDURES**********************/
/*************************************************************/


/*************************************************************/
/*------------PROCEDURE DE CONNECTION ORACLE-------------*/
/*************************************************************/

int CONNEXION()

{
  strcpy(username.arr,"system");
  username.len=strlen(username.arr);
  strcpy(password.arr,"manager");
  password.len=strlen(password.arr);
  EXEC SQL WHENEVER SQLERROR DO errreur();
  EXEC SQL CONNECT :username IDENTIFIED BY :password;

  return 0;
}
```

```
/*********************************************************/
/*-----------------PROCEDURE ENTETE--------------------*/
/*********************************************************/

void ENTETE()

{
  int i;
  char annee[6];
  EXEC SQL SELECT annee
           INTO:h_annee
         FROM pv
         WHERE code_programme=:h_code_prog;
  strcpy(annee,h_annee.arr);
  h_annee.arr[2]='-';
  h_annee.arr[3]=annee[2];
  h_annee.arr[4]=annee[3];
  fprintf(fp,"\n\n\n\n");
  fprintf(fp,"%s","\n|");

fprintf(fp,"%s","-----------------------------------------------");


fprintf(fp,"%s","-----------------------------------------------");

  fprintf(fp,"%s","---------------------------|\n");
  fprintf(fp,"%s%-90s","|      "," UNIVERSITE DE OUAGADOUGOU");
  fprintf(fp,"%-26s%s","  La Patrie ou La Mort,","|\n");
  fprintf(fp,"%s%-90s","|      "," DIRECTION DES AFFAIRES ");
  fprintf(fp,"%-26s%s","  Nous Vaincrons.","|\n");
  fprintf(fp,"%s%-115s","|       ","ACADEMIQUES ET SCOLAIRES");
  fprintf(fp,"%s","|\n");

fprintf(fp,"%s","|---------------------------------------------");


fprintf(fp,"%s","-----------------------------------------------");

  fprintf(fp,"%s","----------------------------|");
  fprintf(fp,"%s","\n\n\n\n");
  fprintf(fp,"%s%s","       ANNEE UNIVERSITAIRE :",h_annee.arr);
  fprintf(fp,"%s","\n\n\n\n");
  fprintf(fp,"%+90s"," INFORMATIONS NECESSAIRES  A LA CALLIGRAPHIE
D'UN DIPLOME");
  fprintf(fp,"%s","\n");
  fprintf(fp,"%+90s","
-----------------------------------------------------");
  fprintf(fp,"\n\n\n\n");
}
```

```
int errreur()
{
  printf("erreur sql:%s",sqlca.sqlerrm.sqlerrmc);
  exit(0);
}


/*********************************************************/
/*------------------PROCEDURE AFFICHER_CORPS------------*/
/*********************************************************/

void AFFICHER_CORPS()
{
EXEC SQL SELECT nom,prenom,date_naiss,lieu_naiss,sexe
      INTO:h_nom_etudiant,h_prenom_etudiant,
          :h_date_naiss,h_lieu_naiss,h_sexe
         FROM candidat
      WHERE code_candidat IN (SELECT code_candidat
                     FROM etudiant
                     WHERE matricule=:h_mat);
EXEC SQL SELECT libelle_etab
       INTO:h_lib_etb
       FROM etablissement
       WHERE code_etablissement=:h_code_etb;
EXEC SQL SELECT libelle_depart
       INTO:h_lib_dep
       FROM departement
       WHERE code_departement=:h_code_dep;
EXEC SQL SELECT lib_option
       INTO:h_lib_option
       FROM options
       WHERE code_option=:h_code_option;
EXEC SQL SELECT lib_filiere
       INTO:h_lib_fil
       FROM filiere
       WHERE code_filiere=:h_code_filiere;
h_nom_etudiant.arr[h_nom_etudiant.len]='\0';
h_prenom_etudiant.arr[h_prenom_etudiant.len]='\0';
h_lib_etb.arr[h_lib_etb.len]='\0';
h_lib_dep.arr[h_lib_dep.len]='\0';
h_lib_fil.arr[h_lib_fil.len]='\0';
h_lib_option.arr[h_lib_option.len]='\0';
h_lib_prog.arr[h_lib_prog.len]='\0';
fprintf(fp,"%s%s","      NOM: ",h_nom_etudiant.arr);
fprintf(fp,"%s","\n");
fprintf(fp,"%s%s","      PRENOM: ",h_prenom_etudiant.arr);
fprintf(fp,"%s","\n");
fprintf(fp,"%s%s","      MATRICULE: ",h_mat.arr);
fprintf(fp,"%s","\n");
fprintf(fp,"%s%s","      DATE DE NAISSANCE: ",h_date_naiss.arr);
fprintf(fp,"%s","\n");
fprintf(fp,"%s%s","      LIEU DE NAISSANCE: ",h_lieu_naiss.arr);
```

```
fprintf(fp,"%s","\n");
fprintf(fp,"%s%s","      SEXE: ",h_sexe.arr);
fprintf(fp,"%s","\n");
fprintf(fp,"%s%s","      ETABLISSEMENT: ",h_lib_etb.arr);
fprintf(fp,"%s","\n");
fprintf(fp,"%s%s","      DEPARTEMENT :",h_lib_dep.arr);
fprintf(fp,"%s","\n");
fprintf(fp,"%s%s","      FILIERE: ",h_lib_fil.arr);
fprintf(fp,"%s","\n");
fprintf(fp,"%s%s", "      PROGRAMME DE FORMATION:
",h_lib_prog.arr);
fprintf(fp,"%s","\n");
fprintf(fp,"%s%s","      OPTION: ",h_lib_option.arr);
fprintf(fp,"%s","\n");
fprintf(fp,"%s%s","      NUMERO PV: ",h_num_pv.arr);
fprintf(fp,"%s","\n");
fprintf(fp,"%s%s","      SESSION DE: ",h_session.arr);
fprintf(fp,"%s","\n");
fprintf(fp,"%s%s","      ANNEE CREATION PV: ",h_date_creat_pv.arr);
fprintf(fp,"%s","\n");
}

/*********************************************************
/*---------------PROCEDURE BAS_DE_PAGE-----------------*/
/*********************************************************/

void BAS_DE_PAGE()

{
EXEC SQL SELECT sysdate
      INTO:h_date_courante
      FROM dual;
fprintf(fp,"%s%+60s%s","\n\n\n\n"," OUAGADOUGOU LE : ",(char *)
h_date_courante.arr);
fprintf(fp,"%s","\n");
fprintf(fp,"%+60s","--------------   ");
fprintf(fp,"%s","\n\n\n\n\n\f");
}
```

ETAT DES FRAIS D'INSCRIPTION

```
#include "/home/daas/prodaas/daas_fonctions/fonc.c"
#include<stdio.h>

/*----------------declaration des variables hotes-------*/
EXEC SQL BEGIN DECLARE SECTION;
  VARCHAR username[20];
  VARCHAR password[20];
  VARCHAR h_num_pv[7];
  VARCHAR h_nom_etudiant[26];
  VARCHAR h_prenom_etudiant[41];
  VARCHAR h_mat[8];
  VARCHAR h_sigle_etb[10];
  VARCHAR h_code_prog[6];
  VARCHAR h_sigle_departement[10];
  VARCHAR h_code_etb[3];
  VARCHAR h_code_dep[4];
  VARCHAR h_lib_etb[41];
  VARCHAR h_lib_dep[41];
  VARCHAR h_niveau[3];
  VARCHAR h_code_candidat[6];
  VARCHAR sigle_filiere[11];
  VARCHAR lib_filiere[41];
  float   h_mt_quit,te_mt_paye,td_mt_paye,tg_mt_paye;
  int  h_mt_ins_prog,te_mt_du,tg_mt_du ,td_mt_du;
  int nb;
  int h_nb_mat;
EXEC SQL END DECLARE SECTION;
  FILE *fp;
  int i,cpt;
  char anc_mat[8];
EXEC SQL INCLUDE SQLCA;

/*************************************************************/
/*************DECLARATION DES  PROCEDURES*******************/
/*************************************************************/
void BAS_DE_PAGE();

int CONNEXION();


void AFFICHER_CORPS();

void ENTETE();
void ENTETE2();
void BAS_DE_PAGE_etb();
void BAS_DE_PAGE_dep();
void BAS_DE_PAGE_general();
int errreur();
```

```
/*********************************************************/
/*--------------PROCEDURE PRINCIPALE--------------------*/
/*********************************************************/
main()

{

if (CONNEXION() < 0)
   exit(-1);
else
{
    system("clear");
    fp=fopen("/dev/lp0","w");
    ENTETE();
    AFFICHER_CORPS();
    BAS_DE_PAGE_general();
    fclose(fp);
}
}


/*************************************************************/
/*******************CORPS DES PROCEDURES**********************/
/*************************************************************/


/*********************************************************/
/*------------PROCEDURE DE CONNECTION ORACLE-------------*/
/*********************************************************/

int CONNEXION()

{
  strcpy(username.arr,"system");
  username.len=strlen(username.arr);
  strcpy(password.arr,"manager");
  password.len=strlen(password.arr);
  EXEC SQL WHENEVER SQLERROR DO errreur();
  EXEC SQL CONNECT :username IDENTIFIED BY :password;

  return 0;
}


/*********************************************************/
/*-----------------PROCEDURE ERREUR2---------------------*/
/*********************************************************/

/*********************************************************/
/*-----------------PROCEDURE ENTETE----------------------*/
/*********************************************************/
```

```
void ENTETE()

{
  fprintf(fp,"\n\n\n\n");
  fprintf(fp,"%s","\n|");

fprintf(fp,"%s","-----------------------------------------------"
);

fprintf(fp,"%s","-----------------------------------------------"
);
  fprintf(fp,"%s","-------------------|\n");
  fprintf(fp,"%s","|");
  for(i=1;i==120;i=i+1)
    fprintf(fp,"%s"," ");
  fprintf(fp,"%s","|\n");
  fprintf(fp,"%s%-90s","|      "," UNIVERSITE DE OUAGADOUGOU");
  fprintf(fp,"%-22s","SERVICE: INSCRIPTION &");
  fprintf(fp,"%s","|\n");
  fprintf(fp,"%s%s","|      "," DIRECTION DES AFFAIRES ");
  fprintf(fp,"%-75s","ACADEMIQUES ET SCOLAIRES");
  fprintf(fp,"%-13s","REINSCRIPTION");
  fprintf(fp,"%s","|\n");

fprintf(fp,"%s","|-----------------------------------------------"
);

fprintf(fp,"%s","-----------------------------------------------"
);
  fprintf(fp,"%s","--------------------|");
  fprintf(fp,"\n\n\n\n");
  fprintf(fp,"%s","                                      ");
  fprintf(fp,"%-110s","LISTE DES ETUDIANTS NON A JOUR DE LEURS
FRAIS D'INSCRIPTION");
  fprintf(fp,"%s","\n                                      ");

fprintf(fp,"%-110s","-----------------------------------------------
-------------");
  fprintf(fp,"%s","\n");
  fprintf(fp,"%s","
");
  fprintf(fp,"%s","
");
  fprintf(fp,"%s","                      ");
  fprintf(fp,"%s","\n");
}

int errreur()
{
  printf("erreur sql:%s",sqlca.sqlerrm.sqlerrmc);
  exit(0);
}
```

```
/*********************************************************/
/*******************PROCEDURE ENTETE2*********************/
/*********************************************************/

void ENTETE2()

{
   h_lib_etb.arr[h_lib_etb.len]='\0';
   h_lib_dep.arr[h_lib_dep.len]='\0';
   fprintf(fp,"\n");

fprintf(fp,"%s","|-------------------------------------------------"
);

fprintf(fp,"%s","-------------------------------------------------"
);
   fprintf(fp,"%s","--------------------|");
   fprintf(fp,"%s","\n");
   fprintf(fp,"%s%+94s","|  ETABLISSEMENT         ","
DEPARTEMENT|");
   fprintf(fp,"\n");

fprintf(fp,"%s%-37s%+77s%s","|",h_lib_etb.arr,h_lib_dep.arr,"|\n");

fprintf(fp,"%s","|-------------------------------------------------"
);

fprintf(fp,"%s","-------------------------------------------------"
);
   fprintf(fp,"%s","--------------------|");
   fprintf(fp,"\n\n");

fprintf(fp,"%s","|-------------------------------------------------"
);

fprintf(fp,"%s","-------------------------------------------------"
);
   fprintf(fp,"%s","--------------------|\n");
   fprintf(fp,"%s","|niveau ");
   fprintf(fp,"%s","| matricule");
   fprintf(fp,"%s","| nom                    ");
   fprintf(fp,"%s","|       prenom(s)              ");
   fprintf(fp,"| montant du ");
   fprintf(fp,"| montant paye ");
   fprintf(fp,"| reste a verser|");

fprintf(fp,"%s","\n|-------------------------------------------------"
);

fprintf(fp,"%s","-------------------------------------------------"
);
   fprintf(fp,"%s","--------------------|\n");
}
```

```
/*********************************************************/
/*-----------------PROCEDURE AFFICHER_CORPS-------------*/
/*********************************************************/

void AFFICHER_CORPS()
{
 float reste,mt_quit,anc_mt_quit;
 int cpt,test,mt_du,anc_mt_ins;
/*******************DECLARATION DE CURSEURS*****************/
 EXEC SQL DECLARE curs_etb CURSOR FOR
                    SELECT code_etablissement,libelle_etab
                    FROM etablissement
                    ORDER BY code_etablissement;

 EXEC SQL OPEN curs_etb;
 anc_mat[0]='\0';
 do
 {
   test=0;
   te_mt_du=0;
   te_mt_paye=0;
   EXEC SQL WHENEVER NOT FOUND GOTO fin_curs;
   EXEC SQL FETCH curs_etb INTO  :h_code_etb,
                     :h_lib_etb;
   EXEC SQL DECLARE curs_dep CURSOR FOR
                    SELECT code_departement,libelle_depart
                    FROM departement
                    WHERE code_etablissement=:h_code_etb
                    ORDER BY code_departement;
   EXEC SQL OPEN curs_dep;

   do
   {
      td_mt_du=0;
      td_mt_paye=0;
      EXEC SQL WHENEVER NOT FOUND GOTO fin_curs2;
      EXEC SQL FETCH curs_dep INTO :h_code_dep,
                       :h_lib_dep;
      EXEC SQL DECLARE curs_frais CURSOR FOR
      select matricule,mt_quit
      from frais
      where nat_frais='in'
      or    nat_frais='IN'
      and code_departement=:h_code_dep
      order by matricule;
      EXEC SQL OPEN curs_frais;
      ENTETE2() ;
      test=1;
      do
      {
            EXEC SQL WHENEVER NOT FOUND GOTO fin_curs1;
            EXEC SQL FETCH curs_frais INTO :h_mat,
                                    :h_mt_quit;
```


```
        EXEC SQL WHENEVER NOT FOUND GOTO continuer;
        EXEC SQL SELECT mt_ins_prog
                 INTO:h_mt_ins_prog
                 FROM ins_prog
                  WHERE mode_paie='T'
                  AND   matricule=:h_mat;

        EXEC SQL SELECT code_candidat
                INTO:h_code_candidat
                FROM etudiant
                WHERE matricule=:h_mat;

        EXEC SQL SELECT nom,prenom
                INTO:h_nom_etudiant,
                    :h_prenom_etudiant
                FROM candidat
               WHERE code_candidat=:h_code_candidat;

        h_nom_etudiant.arr[h_nom_etudiant.len]='\0';
        h_prenom_etudiant.arr[h_prenom_etudiant.len]='\0';

     EXEC SQL SELECT niveau
                 INTO:h_niveau
                 FROM programme
                 WHERE code_departement=:h_code_dep;

     reste=h_mt_ins_prog - h_mt_quit;
        td_mt_du=td_mt_du + h_mt_ins_prog;
        td_mt_paye=td_mt_paye + h_mt_quit;

        fprintf(fp,"\n");
        fprintf(fp,"%s%+5s","|",h_niveau.arr);
        fprintf(fp,"%s%+8s","  |",h_mat.arr);
        fprintf(fp,"%s%-24s","  |", h_nom_etudiant.arr);
        fprintf(fp,"%s%-29s","|", h_prenom_etudiant.arr);
        fprintf(fp,"%s%10d","|",h_mt_ins_prog);
        fprintf(fp,"%s%12.0f","  |",h_mt_quit);
        fprintf(fp,"%s%15.0f%s","  |",reste,"|");
        continuer:;

   }while(1);

   fin_curs1:EXEC SQL CLOSE curs_frais;

   BAS_DE_PAGE_dep();

   te_mt_du=te_mt_du + td_mt_du;
   te_mt_paye=te_mt_paye + td_mt_paye;

}while(2);

fin_curs2:EXEC SQL CLOSE curs_dep;
          tg_mt_du=tg_mt_du + te_mt_du;
          tg_mt_paye=tg_mt_paye + te_mt_paye;
```

```
  if(test==0)
   {
      ENTETE2();
      BAS_DE_PAGE_dep();
    }

   BAS_DE_PAGE_etb();

 }while(3);

 fin_curs:EXEC SQL CLOSE curs_etb;

 fin_curs3:EXEC SQL CLOSE curs_etb;
        return ;
 }



/*******************************************************/
/*---------------PROCEDURE BAS_DE_PAGE-----------------*/
/*******************************************************/

void BAS_DE_PAGE_dep()

{
  float td_mt_reste;
  td_mt_reste=td_mt_du - td_mt_paye;

fprintf(fp,"%s","\n|-----------------------------------------------"
);

fprintf(fp,"%s","-----------------------------------------------"
);
  fprintf(fp,"%s","----------------------|\n");
  fprintf(fp,"%-78s","TOTAL DEPARTEMENT:");
  fprintf(fp,"%10d%12.0f%15.0f",td_mt_du,td_mt_paye,td_mt_reste);
  fprintf(fp,"\n\n");
}



void BAS_DE_PAGE_etb()

{
  float te_mt_reste;
  te_mt_reste=te_mt_du - te_mt_paye;

fprintf(fp,"%s","\n|-----------------------------------------------"
);

fprintf(fp,"%s","-----------------------------------------------"
);
```

```
  fprintf(fp,"%s","----------------------|\n");
  fprintf(fp,"%-78s","TOTAL ETABLISSEMENT:");
  fprintf(fp,"%10d%12.0f%15.0f",te_mt_du,te_mt_paye,te_mt_reste);
  fprintf(fp,"\n\n");
}


void BAS_DE_PAGE_general()

{
  float tg_mt_reste;
  tg_mt_reste=tg_mt_du - tg_mt_paye;

fprintf(fp,"%s","\n|-----------------------------------------------"
);

fprintf(fp,"%s","-------------------------------------------------"
);
  fprintf(fp,"%s","----------------------|");
  fprintf(fp,"%-78s","\nTOTAL GENERAL :");
  fprintf(fp,"%10d%12.0f%15.0f",tg_mt_du,tg_mt_paye,tg_mt_reste);
  fprintf(fp,"\n\n");
}
```

LISTE DES ADMIS

```
#include<stdio.h>
/*------------------DECLARATION DES VARIABLES-----------------*/
EXEC SQL BEGIN DECLARE SECTION;
  VARCHAR username[20];
  VARCHAR password[20];
  VARCHAR h_mat[8];
  VARCHAR h_date_courante[9];
  VARCHAR h_nom_etudiant[26];
  VARCHAR h_prenom_etudiant[41];
  VARCHAR h_p_conc[2];
  VARCHAR h_p_es[2];
  VARCHAR h_p_dossier[2];
  VARCHAR h_date_naiss[9];
  VARCHAR h_lieu_naiss[20];
  VARCHAR h_rang[5];
  VARCHAR h_lib_prog[31];
  VARCHAR h_code_prog[6];
  VARCHAR h_etat_civil[81];
  VARCHAR h_type[2];
EXEC SQL END DECLARE SECTION;
```

```
  FILE *fp;
  char choix[2], *lib_prog[31];
  char code_prog[6];
  int cpt;
EXEC SQL INCLUDE SQLCA;

/*****************************************************************/
/***************DECLARATION DES PROCEDURES*********************/
/*****************************************************************/
void afficher_menu2();
void afficher_menu();
void msg();
void ENTETE();
int CONNEXION();
void fin_curs_candidat1();
void fin_curs_candidat2();
void fin_curs_candidat3();
void AFFICHER_CORPS1();
void AFFICHER_CORPS2();
void AFFICHER_CORPS3();
void BAS_DE_PAGE();
void erreur();
void concours();
void dossier();
void examen();

/*****************************************************************/
/****************PROCEDURE PRINCIPALE*************************/
/*****************************************************************/
main()
{
char c;
if (CONNEXION() < 0)
       exit(-1);
else
{

        system("clear");
        fp=fopen("/dev/lp0","w");
     do
     {
         cpt=0;
      afficher_menu();
      switch(choix[0])
      {
      case '1':afficher_menu2();
            concours();break;
      case '2':afficher_menu2();
            examen();break;
      case '3':afficher_menu2();
            dossier();break;
      case '4':return;
      }
```

```
 BAS_DE_PAGE();
      }while(choix[0]!='4');
}
}


/*************************************************************/
/***************CORPS DES PROCEDURES**************************/
/*************************************************************/


/*************************************************************/
/***************PROCEDURE DE CONNEXION ORACLE*****************/
/*************************************************************/

int CONNEXION()
{
  strcpy(username.arr,"system");
  strcpy(password.arr,"manager");
  username.len=strlen(username.arr);
  password.len=strlen(password.arr);
  EXEC SQL WHENEVER SQLERROR DO erreur();
  EXEC SQL CONNECT :username IDENTIFIED BY :password;
  return 0;
}


/*************************************************************/
/*****************PROCEDURE ERREUR****************************/
/*************************************************************/


void erreur()
{
  printf("%s%s","erreur sql:",sqlca.sqlerrm.sqlerrmc);
  exit(0);
}

/*************************************************************/
/*************PROCEDURE AFFICHER_MENU*************************/
/*************************************************************/
```

```
 void  afficher_menu()
  {
   system("clear");
   printf("\n\n\n\n\n\n");
   printf("                                        MENU   PRINCIPAL
\n");
   printf("                                        ----------------
\n");
   printf("
--------------------------\n\n");
   printf("                                         1 .PAR CONCOURS
\n\n");
   printf("                                         2 .PAR EXAMEN SPECIAL
\n\n");
   printf("                                         3 .PAR  DOSSIER
\n\n");
   printf("                                         4 . ABANDON
\n\n");
   printf("
--------------------------- \n\n");
   printf("                                            VOTRE CHOIX : ");
   scanf("%s",choix);
  }


/*****************************************************************/
/****************PROCEDURE AFFICHER_MENU2*************************/
/*****************************************************************/

void  afficher_menu2()
  {
   system("clear");
   printf("\n\n\n\n\n\n\n\n\n\n\n");
   printf("                CODE DU PROGRAMME: ");
   scanf("%s",&code_prog);
   strcpy(h_code_prog.arr,code_prog);
   h_code_prog.len=strlen(h_code_prog.arr);
  }

/*****************************************************************/
/*******************PROCEDURE MSG*********************************/
/*****************************************************************/


void msg()
{
char c;
printf("%s","\n\n Ce code ne correspond a aucun programme du type
choisi\n");
scanf("%c",&c);
exit(0);

}
```

```
/*************************************************************/
/*****************PROCEDURE CONCOURS**************************/
/*************************************************************/


void concours()
{
EXEC SQL WHENEVER NOT FOUND  DO msg();
EXEC SQL SELECT f1.libelle_programme
       INTO :h_lib_prog
       FROM programme f1,resultats_acces f2
       WHERE f1.code_programme=:h_code_prog
       AND f2.type_acces='c';
ENTETE();
AFFICHER_CORPS1();
}

/*************************************************************/
/*****************PROCEDURE DOSSIER***************************/
/*************************************************************/


void dossier()
{
EXEC SQL WHENEVER NOT FOUND  DO msg();
EXEC SQL SELECT f1.libelle_programme
       INTO :h_lib_prog
       FROM programme f1,resultats_acces f2
       WHERE f1.code_programme=:h_code_prog
       AND f2.type_acces='D';
ENTETE();
AFFICHER_CORPS2()  ;
}

/*************************************************************/
/*****************PROCEDURE EXAMEN****************************/
/*************************************************************/

void examen()

{
EXEC SQL WHENEVER NOT FOUND  DO msg();
EXEC SQL SELECT f1.libelle_programme
       INTO :h_lib_prog
       FROM programme f1,resultats_acces f2
       WHERE f1.code_programme=:h_code_prog
       AND f2.type_acces='E';
ENTETE();
AFFICHER_CORPS3();
}
```

```
/*************************************************************/
/*********************PROCEDURE ENTETE************************/
/*************************************************************/

void ENTETE()
{
fprintf(fp,"%s","\n\n\n\n\n\n\n");
fprintf(fp,"%s","|-----------------------------------------------"
);
fprintf(fp,"%s","-----------------------------------------------")
;
fprintf(fp,"%s","-----------------------------------------------")
;
fprintf(fp,"%s","-------------------------|");
fprintf(fp,"%s","\n|  UNIVERSITE DE OUAGADOUGOU");
fprintf(fp,"%s","\n|  DIRECTION DES AFFAIRES ACADEMIQUES ET
SCOLAIRES");
fprintf(fp,"\n\n\n\n");
fprintf(fp,"%s%+50s%s","|","LISTE DES RESULTATS A
",h_lib_prog.arr);
fprintf(fp,"%s%+80s","\n|","-------------------------------------"
);
fprintf(fp,"\n\n\n");
fprintf(fp,"%s","|-----------------------------------------------"
);
fprintf(fp,"%s","-----------------------------------------------")
;
fprintf(fp,"%s","-----------------------------------------------")
;
fprintf(fp,"%s","-------------------------|");
fprintf(fp,"%s%s","\n| Ordre de merite","|           NOM
");
fprintf(fp,"%s","|                PRENOM(S)              " );
fprintf(fp,"%s","| DATE & LIEU DE NAISSANCE  ");
fprintf(fp,"%s","|  ETAT CIVIL  |");
fprintf(fp,"\n");
fprintf(fp,"%s","|-----------------------------------------------"
);
fprintf(fp,"%s","-----------------------------------------------")
;
fprintf(fp,"%s","-----------------------------------------------")
;
fprintf(fp,"%s","-------------------------|");
fprintf(fp,"\n");
}


/*************************************************************/
/****************PROCEDURE AFFICHER_CORPS1********************/
/*************************************************************/
```

```
void AFFICHER_CORPS1()
{
EXEC SQL DECLARE curs_candidat1 CURSOR FOR
       SELECT f3.nom,f3.prenom,
           f3.etat_civil,f3.date_naiss,
           f3.lieu_naiss,f2.rang
          FROM programme f1,resultats_acces f2,candidat f3
       WHERE (f1.code_programme=f2.code_programme)
       AND  (f2.type_acces='c')
       AND  (f2.code_candidat=f3.code_candidat)
       ORDER BY f2.rang;
EXEC SQL OPEN curs_candidat1;
EXEC SQL WHENEVER NOT FOUND GOTO fin_curs1;
do
{
EXEC SQL FETCH curs_candidat1 INTO :h_nom_etudiant,
                      :h_prenom_etudiant,
                      :h_etat_civil,
                      :h_date_naiss,
                      :h_lieu_naiss,
                                  :h_rang;
h_nom_etudiant.arr[h_nom_etudiant.len]='\0';
h_prenom_etudiant.arr[h_prenom_etudiant.len]='\0';
h_etat_civil.arr[h_etat_civil.len]='\0';
h_date_naiss.arr[h_date_naiss.len]='\0';
h_lieu_naiss.arr[h_lieu_naiss.len]='\0';
fprintf(fp,"%s%+10s","|",h_rang.arr);
fprintf(fp,"%s%-24s","      |",h_nom_etudiant.arr);
fprintf(fp,"%s%-37s","|",h_prenom_etudiant.arr);
fprintf(fp,"%s%-13s","|",h_date_naiss.arr);
fprintf(fp,"%+14s%s",h_lieu_naiss.arr,"|");
fprintf(fp,"%+14s%s",h_etat_civil.arr,"|\n");
cpt=cpt+1;
}while(1);
fin_curs1:fin_curs_candidat1();
}


/*********************************************************/
/****************PROCEDURE AFFICHER_CORPS2****************/
/*********************************************************/


void AFFICHER_CORPS2()
{
EXEC SQL DECLARE curs_candidat2 CURSOR FOR
       SELECT f3.nom,f3.prenom,
           f3.etat_civil,f3.date_naiss,
           f3.lieu_naiss,f2.rang
          FROM programme f1,resultats_acces f2,candidat f3
       WHERE (f1.code_programme=f2.code_programme)
       AND  (f2.type_acces='D')
       AND  (f2.code_candidat=f3.code_candidat)
       ORDER BY f2.rang;
```

```
EXEC SQL OPEN curs_candidat2;
EXEC SQL WHENEVER NOT FOUND GOTO fin_curs2;
do
{
EXEC SQL FETCH curs_candidat2 INTO :h_nom_etudiant,
                      :h_prenom_etudiant,
                      :h_etat_civil,
                      :h_date_naiss,
                      :h_lieu_naiss,
                                  :h_rang;
h_nom_etudiant.arr[h_nom_etudiant.len]='\0';
h_prenom_etudiant.arr[h_prenom_etudiant.len]='\0';
h_etat_civil.arr[h_etat_civil.len]='\0';
h_date_naiss.arr[h_date_naiss.len]='\0';
h_lieu_naiss.arr[h_lieu_naiss.len]='\0';
fprintf(fp,"%s%+10s","|",h_rang.arr);
fprintf(fp,"%s%-24s","      |",h_nom_etudiant.arr);
fprintf(fp,"%s%-37s","|",h_prenom_etudiant.arr);
fprintf(fp,"%s%-13s","|",h_date_naiss.arr);
fprintf(fp,"%+14s%s",h_lieu_naiss.arr,"|");
fprintf(fp,"%+14s%s",h_etat_civil.arr,"|\n");
cpt=cpt+1;
}while(1);
fin_curs2:fin_curs_candidat2();
}

/*************************************************************/
/*****************PROCEDURE AFFICHER_CORPS3*******************/
/*************************************************************/

void AFFICHER_CORPS3()
{
EXEC SQL DECLARE curs_candidat3 CURSOR FOR
       SELECT f3.nom,f3.prenom,
           f3.etat_civil,f3.date_naiss,
           f3.lieu_naiss,f2.rang
          FROM programme f1,resultats_acces f2,candidat f3
       WHERE (f1.code_programme=f2.code_programme)
       AND  (f2.type_acces='E')
       AND  (f2.code_candidat=f3.code_candidat)
       ORDER BY f2.rang;
EXEC SQL OPEN curs_candidat3;
EXEC SQL WHENEVER NOT FOUND GOTO fin_curs3;
do
{
EXEC SQL FETCH curs_candidat3 INTO :h_nom_etudiant,
                      :h_prenom_etudiant,
                      :h_etat_civil,
                      :h_date_naiss,
                      :h_lieu_naiss,
                                  :h_rang;
```

```
h_nom_etudiant.arr[h_nom_etudiant.len]='\0';
h_prenom_etudiant.arr[h_prenom_etudiant.len]='\0';
h_etat_civil.arr[h_etat_civil.len]='\0';
h_date_naiss.arr[h_date_naiss.len]='\0';
h_lieu_naiss.arr[h_lieu_naiss.len]='\0';
fprintf(fp,"%s%+10s","|",h_rang.arr);
fprintf(fp,"%s%-24s","      |",h_nom_etudiant.arr);
fprintf(fp,"%s%-37s","|",h_prenom_etudiant.arr);
fprintf(fp,"%s%-13s","|",h_date_naiss.arr);
fprintf(fp,"%+14s%s",h_lieu_naiss.arr,"|");
fprintf(fp,"%+14s%s",h_etat_civil.arr,"|\n");
cpt=cpt+1;
}while(1);
fin_curs3:fin_curs_candidat3();
}
/*******************FIN PROCEDURE**********************/



void fin_curs_candidat1()
{
EXEC SQL CLOSE curs_candidat1;
return;
}




void fin_curs_candidat2()
{
EXEC SQL CLOSE curs_candidat2;
return;
}



void fin_curs_candidat3()
{
EXEC SQL CLOSE curs_candidat3;
}

/*************************************************************/
/**************PROCEDURE BAS_DE_PAGE**************************/
/*************************************************************/

void BAS_DE_PAGE()
 {
  EXEC SQL WHENEVER NOT FOUND CONTINUE;
  EXEC SQL WHENEVER SQLERROR DO erreur();
  EXEC SQL SELECT TO_CHAR (sysdate,'DD-MM-YY')
           INTO :h_date_courante
           FROM dual;
```

```
fprintf(fp,"%s","\n|-----------------------------------------------
-");

fprintf(fp,"%s","-------------------------------------------------"
);

fprintf(fp,"%s","-------------------------------------------------"
);
 fprintf(fp,"%s","----------------|\n");
 fprintf(fp,"%80s%d%s","Arrete la presente liste a ",cpt,"
nom(s).");
 fprintf(fp,"%70s%s","\n Ouagadougou le ",h_date_courante.arr);
 fprintf(fp,"%70s","\n Le president du jury");
 fprintf(fp,"%s","\n\n\n\n\f");
 }
```

PV NON RENSEIGNE

```
#include "/home/daas/prodaas/daas_fonctions/fonc.c"
#include<stdio.h>

/*----------------declaration des variables hotes-------*/
EXEC SQL BEGIN DECLARE SECTION;
  VARCHAR username[20];
  VARCHAR password[20];
  char c;
  VARCHAR h_num_pv[7];
  VARCHAR h_nom_etudiant[26];
  VARCHAR h_prenom_etudiant[41];
  float   h_moy_obt;
  VARCHAR h_resultat_obt [3];
  VARCHAR h_mat[8];
  VARCHAR h_sigle_etb[10];
  VARCHAR h_rang[5];
  VARCHAR h_code_prog[6];
  VARCHAR h_sigle_departement[10];
  VARCHAR h_code_option[4];
  VARCHAR h_niveau[3];
  VARCHAR h_code_candidat[6];
  VARCHAR h_lib_option[31];
  VARCHAR h_session[2];
  int nb;
  int h_nb_mat;
EXEC SQL END DECLARE SECTION;
  FILE *fp;
  int i,cpt;
  char code_prog[6];
EXEC SQL INCLUDE SQLCA;
```

```
/*************************************************************/
/*************DECLARATION DES  PROCEDURES*******************/
/*************************************************************/
void BAS_DE_PAGE();

int CONNEXION();


void AFFICHER_CORPS();

void ENTETE();

void BAS_DE_PAGE();
void NOMORE();
int errreur();


/*************************************************************/
/*--------------PROCEDURE PRINCIPALE--------------------*/
/*************************************************************/
main()

{

system("clear");
printf("\n\n\n\n\n\n\n\n\n\n\n");
printf("                CODE DU PROGRAMME: ");
scanf("%s",&code_prog);
strcpy(h_code_prog.arr,code_prog);
h_code_prog.len=strlen(h_code_prog.arr);
if (CONNEXION() < 0)
   exit(-1);
else
{
  system("clear");
   fp=fopen("/dev/lp0","w");
   EXEC SQL DECLARE curs_pv CURSOR FOR SELECT
                            ses_sion,
                            num_pv
                     FROM  pv
                     WHERE code_programme=:h_code_prog;
    EXEC SQL OPEN curs_pv;
do
{
    EXEC SQL WHENEVER NOT FOUND GOTO fin_curs_pv;
    EXEC SQL FETCH curs_pv INTO
                     :h_session,
                     :h_num_pv;
    ENTETE();
    AFFICHER_CORPS();
    BAS_DE_PAGE();
    EXEC SQL WHENEVER NOT FOUND CONTINUE;
}while(1);
```

```
  fin_curs_pv:EXEC SQL CLOSE curs_pv;
    fclose(fp);

}
}


/*************************************************************/
/*******************CORPS DES PROCEDURES**********************/
/*************************************************************/


/*************************************************************/
/*------------PROCEDURE DE CONNECTION ORACLE-------------*/
/*************************************************************/

int CONNEXION()

{
  strcpy(username.arr,"system");
  username.len=strlen(username.arr);
  strcpy(password.arr,"manager");
  password.len=strlen(password.arr);
  EXEC SQL WHENEVER SQLERROR DO errreur();
  EXEC SQL CONNECT :username IDENTIFIED BY :password;

  return 0;
}



/*************************************************************/
/*-----------------PROCEDURE ENTETE--------------------*/
/*************************************************************/

void ENTETE()

{
  EXEC SQL WHENEVER SQLERROR DO errreur();
  EXEC SQL SELECT code_option,
            niveau,
            sigle_etablessement,
            sigle_departement
            INTO  :h_code_option,
            :h_niveau,
            :h_sigle_etb,
            :h_sigle_departement
            FROM  programme
         WHERE code_programme IN (SELECT code_programme
                         FROM pv
                         WHERE num_pv=:h_num_pv);
```

```
 EXEC SQL SELECT lib_option
          INTO :h_lib_option
          FROM options
          WHERE code_option =:h_code_option;
  fprintf(fp,"\n\n\n\n");
  fprintf(fp,"%s","\n|");

fprintf(fp,"%s","-----------------------------------------------------"
);

fprintf(fp,"%s","-----------------------------------------------------"
);

fprintf(fp,"%s","-----------------------------------------------------"
);
  fprintf(fp,"%s","------------");
  for(i=1;i==133;i=i+1)
    fprintf(fp,"%s"," ");
  fprintf(fp,"%s","|\n");
  fprintf(fp,"%s%-127s","|     "," UNIVERSITE DE OUAGADOUGOU");
  fprintf(fp,"%s","|\n");
  fprintf(fp,"%s%s","|     "," DIRECTION DES AFFAIRES ");
  fprintf(fp,"%-103s","ACADEMIQUES ET SCOLAIRES");
  fprintf(fp,"%s","|\n");

fprintf(fp,"%s","|---------------------------------------------------"
);

fprintf(fp,"%s","-----------------------------------------------------"
);

fprintf(fp,"%s","-----------------------------------------------------"
);
  fprintf(fp,"%s","------------|");
  fprintf(fp,"\n");
  fprintf(fp,"%s%-20s%-30s","|     ","SCOLARITE DE
:",h_sigle_etb.arr);
  fprintf(fp,"%-20s%-10s","PROCES_VERBALE No : ",h_num_pv.arr);
  fprintf(fp,"%-14s%-10s","SESSION DE: ",h_session.arr);
  fprintf(fp,"%s","|\n");
  fprintf(fp,"%s%-50s","|     ","------------");
  fprintf(fp,"%-29s","----------------- ");
  fprintf(fp,"%-44s","---------- ");
  fprintf(fp,"%s","                |");
  fprintf(fp,"%s","\n|");
  fprintf(fp,"%s","
");
  fprintf(fp,"%s","
");
  fprintf(fp,"%s","
");
```

```
  fprintf(fp,"%s","             ");
  fprintf(fp,"%s","|\n");
  fprintf(fp,"%s%-20s%-20s","|
","ETABLISSEMENT:",h_sigle_etb.arr);
  fprintf(fp,"%-20s%-20s"," DEPARTEMENT
:",h_sigle_departement.arr);
  fprintf(fp,"%-10s%-4s", "NIVEAU : ",h_niveau.arr);
  fprintf(fp,"%-10s%-20s","OPTION:  ",h_lib_option.arr);
  fprintf(fp,"%s","|\n");
  fprintf(fp,"%s%-40s","|      ","-------------");
  fprintf(fp,"%-40s"," -----------");
  fprintf(fp,"%-14s", "------- ");
  fprintf(fp,"%-40s","------");
  fprintf(fp,"%s","|");
  fprintf(fp,"%s","\n|");
  for(i=1;i==133;i=i+1)
    fprintf(fp,"%s"," ");
  fprintf(fp,"%s","|");
  fprintf(fp,"%s","\n|");

fprintf(fp,"%s","-----------------------------------------------------"
);

fprintf(fp,"%s","-----------------------------------------------------"
);

fprintf(fp,"%s","-----------------------------------------------------"
);
  fprintf(fp,"%s","------------");
  fprintf(fp,"%s","|\n");
  fprintf(fp,"| No ");
  fprintf(fp,"|      NOM & PRENOM(S)      ");
  fprintf(fp,"|MATRICULE");
  fprintf(fp,"|STAGE ");
  fprintf(fp,"|               NOTES                ");
  fprintf(fp,"|memo  ");
  fprintf(fp,"|total");
  fprintf(fp,"|MOY.GE");
  fprintf(fp,"|RESULT");
  fprintf(fp,"|CLASS");
  fprintf(fp,"|OBSERV |");
  fprintf(fp,"%s","\n|");

fprintf(fp,"%s","-----------------------------------------------------"
);

fprintf(fp,"%s","-----------------------------------------------------"
);

fprintf(fp,"%s","-----------------------------------------------------"
);
  fprintf(fp,"%s","------------");
  fprintf(fp,"%s","|\n");
}
```

```
int errreur()
{
  printf("erreur sql:%s",sqlca.sqlerrm.sqlerrmc);
  exit(0);
}


/*********************************************************/
/*--------------PROCEDURE NOMORE-----------------------*/
/*********************************************************/


void NOMORE()

{
  fprintf(fp," \n");
}

/*********************************************************/
/*------------------PROCEDURE AFFICHER_CORPS------------*/
/*********************************************************/

void AFFICHER_CORPS()
{
char nom_etudiant[26],prenom_etudiant[41];
char mat[8];
cpt=1;
EXEC SQL DECLARE curs_mat CURSOR FOR
        SELECT candidat.nom,
            candidat.prenom,
            etudiant.matricule
         FROM candidat,etudiant
      WHERE (candidat.code_candidat = etudiant.code_candidat)
      AND (etudiant.matricule IN (SELECT matricule
                        FROM ins_prog
                       WHERE num_pv =:h_num_pv))
  ORDER BY candidat.nom,candidat.prenom;

EXEC SQL SELECT COUNT(DISTINCT code_module)
       INTO :h_nb_mat
       FROM mod_pgm
       WHERE code_programme=:h_code_prog;

/*parametrage du nbre de col pour les notes*/

EXEC SQL WHENEVER NOT FOUND GOTO fin_curs;
EXEC SQL OPEN curs_mat;
do
{
   EXEC SQL FETCH curs_mat INTO
                              :h_nom_etudiant,
                              :h_prenom_etudiant,
                                 :h_mat;
```

```
   decharger_varhote(nom_etudiant,h_nom_etudiant.arr,
   h_nom_etudiant.len);
   decharger_varhote(prenom_etudiant,h_prenom_etudiant.arr,
   h_prenom_etudiant.len);
   decharger_varhote(mat,h_mat.arr,h_mat.len);
   fprintf(fp,"%s%-3d","| ",cpt);
   cpt=cpt+1;
   fprintf(fp,"%s%-13s%s","|",nom_etudiant," ");
   fprintf(fp,"%+13s",prenom_etudiant);
   fprintf(fp,"%s%+9s","|",mat);
   fprintf(fp,"%s","|       ");
   fprintf(fp,"|     |     |     |     |     |     ");
   fprintf(fp,"|       ");
   fprintf(fp,"|      ");
   fprintf(fp,"|       ");
   fprintf(fp,"|       ");
   fprintf(fp,"|      ");
   fprintf(fp,"|        |");
   fprintf(fp,"\n");
}
while(1);
fin_curs: EXEC SQL CLOSE curs_mat;
}

/*********************************************************/
/*---------------PROCEDURE BAS_DE_PAGE-----------------*/
/*********************************************************/

void BAS_DE_PAGE()

{
 fprintf(fp,"%s","|");

fprintf(fp,"-----------------------------------------------------------
");

fprintf(fp,"-----------------------------------------------------------
");
 fprintf(fp,"-------------------------------------------");
 fprintf(fp,"-------");
 fprintf(fp,"%s","|");
 fprintf(fp,"\n");
 fprintf(fp,"%s%d","| Total inscrits: ",cpt-1);
 fprintf(fp,"%+40s","Admis:");
 fprintf(fp,"%+40s","Membres du jury");
 fprintf(fp,"\n");
 fprintf(fp,"%s","| Presents : ");
 fprintf(fp,"%+40s","Ajournes:");
 fprintf(fp,"\n");
 fprintf(fp,"%s","| Absents:");
 fprintf(fp,"%+124s","\n|");

fprintf(fp,"-----------------------------------------------------------
");
```

```
fprintf(fp,"-----------------------------------------------------
");
 fprintf(fp,"----------------------------------------");
 fprintf(fp,"-------");
 fprintf(fp,"%s","|");
 fprintf(fp,"\n");
}
```

PV RENSEIGNE

```
#include "/home/daas/prodaas/daas_fonctions/fonc.c"
#include<stdio.h>

/*----------------declaration des variables hotes-------*/
EXEC SQL BEGIN DECLARE SECTION;
  VARCHAR username[20];
  VARCHAR password[20];
  char c;
 VARCHAR h_num_pv[7];
 VARCHAR h_nom_etudiant[26];
 VARCHAR h_prenom_etudiant[41];
 float  h_stage,  h_moy_gene;
 VARCHAR h_resultat_obt [3];
 VARCHAR h_mat[8];
VARCHAR h_sigle_etb[11];
 int nb_admis,nb_ajournes, h_rang;
  VARCHAR date_signe_pv[9];
 VARCHAR h_code_prog[6];
 VARCHAR h_sigle_departement[11];
 VARCHAR h_code_option[4];
 VARCHAR h_niveau[3];
 VARCHAR h_code_candidat[6];
    float h_moy_obt,h_coef;
  VARCHAR sigle_filiere[11];
  VARCHAR lib_filiere[41];
  VARCHAR h_code_module[8];
  VARCHAR h_code_module2[8];
  VARCHAR h_lib_option[31];
  VARCHAR h_session[2];
 int nb,nb_present,nb_absent;
EXEC SQL END DECLARE SECTION;
  FILE *fp;
  int i,cpt;
  char code_prog[6];
EXEC SQL INCLUDE SQLCA;

/*************************************************************/
/************DECLARATION DES  PROCEDURES******************/
/*************************************************************/
```

```
void BAS_DE_PAGE();

int CONNEXION();


void AFFICHER_CORPS();

void ENTETE();

void BAS_DE_PAGE();
void fin_curs_cours();
void NOMORE();
int errreur();


/*********************************************************/
/*--------------PROCEDURE PRINCIPALE--------------------*/
/*********************************************************/
main()

{
system("clear");
printf("\n\n\n\n\n\n\n\n\n\n\n");
printf("                CODE DU PROGRAMME: ");
scanf("%s",&code_prog);
strcpy(h_code_prog.arr,code_prog);
h_code_prog.len=strlen(h_code_prog.arr);
if (CONNEXION() < 0)
   exit(-1);
else
{
  system("clear");
   fp=fopen("/dev/lp0","w");
   EXEC SQL DECLARE curs_pv CURSOR FOR SELECT
                            ses_sion,
                            num_pv
                     FROM  pv
                     WHERE code_programme=:h_code_prog;
   EXEC SQL OPEN curs_pv;
 do
   {
   EXEC SQL WHENEVER NOT FOUND GOTO fin_curs_pv;
   EXEC SQL FETCH curs_pv INTO:h_session,
                     :h_num_pv;
   ENTETE();
   AFFICHER_CORPS();
   BAS_DE_PAGE();
   fprintf(fp,"\n");
   nb_admis=0;
   nb_ajournes=0;
   EXEC SQL WHENEVER NOT FOUND CONTINUE;
   }while(1);
   fin_curs_pv: EXEC SQL CLOSE curs_pv;
   fclose(fp);

}
}
```

```
/*************************************************************/
/*******************CORPS DES PROCEDURES**********************/
/*************************************************************/


/*************************************************************/
/*------------PROCEDURE DE CONNECTION ORACLE-------------*/
/*************************************************************/

int CONNEXION()

{
  strcpy(username.arr,"system");
  username.len=strlen(username.arr);
  strcpy(password.arr,"manager");
  password.len=strlen(password.arr);
  EXEC SQL WHENEVER SQLERROR DO errreur();
  EXEC SQL CONNECT :username IDENTIFIED BY :password;

  return 0;
}



/*************************************************************/
/*-----------------PROCEDURE ENTETE--------------------*/
/*************************************************************/

void ENTETE()

{
  int t;
  EXEC SQL WHENEVER SQLERROR DO errreur();
  EXEC SQL SELECT code_option,
            niveau,
            sigle_etablessement,
            sigle_departement
            INTO  :h_code_option,
            :h_niveau,
            :h_sigle_etb,
            :h_sigle_departement
            FROM  programme
         WHERE code_programme IN (SELECT code_programme
                        FROM pv
                        WHERE num_pv=:h_num_pv);
  EXEC SQL SELECT lib_option
          INTO :h_lib_option
          FROM options
          WHERE code_option =:h_code_option;
  fprintf(fp,"\n\n\n\n");
  fprintf(fp,"%s","\n|");

fprintf(fp,"%s","-----------------------------------------------"
);
```

```
fprintf(fp,"%s","-------------------------------------------------"
);

fprintf(fp,"%s","-------------------------------------------------"
);
  fprintf(fp,"%s","------------");
  for(i=1;i==133;i=i+1)
    fprintf(fp,"%s"," ");
  fprintf(fp,"%s","|\n");
  fprintf(fp,"%s%-125s","|      "," UNIVERSITE DE OUAGADOUGOU");
  fprintf(fp,"%s","|\n");
  fprintf(fp,"%s%s","|      "," DIRECTION DES AFFAIRES ");
  fprintf(fp,"%-101s","ACADEMIQUES ET SCOLAIRES");
  fprintf(fp,"%s","|\n");

fprintf(fp,"%s","|-------------------------------------------------"
);

fprintf(fp,"%s","-------------------------------------------------"
);

fprintf(fp,"%s","-------------------------------------------------"
);
  fprintf(fp,"%s","----------|");
  fprintf(fp,"\n");
  fprintf(fp,"%s%-20s%-30s","|      ","SCOLARITE DE
:",h_sigle_etb.arr);
  fprintf(fp,"%-20s%-10s","PROCES_VERBALE No : ",h_num_pv.arr);
  fprintf(fp,"%-14s%-10s","SESSION DE: ",h_session.arr);
  fprintf(fp,"%+17s","|\n");
  fprintf(fp,"%s%-50s","|      ","------------");
  fprintf(fp,"%-29s","----------------- ");
  fprintf(fp,"%-44s","---------- ");
  fprintf(fp,"%s","                |");
  fprintf(fp,"%s","\n|");
  fprintf(fp,"%s","
");
  fprintf(fp,"%s","
");
  fprintf(fp,"%s","
");
  fprintf(fp,"%s","          ");
  fprintf(fp,"%s","|\n");
  fprintf(fp,"%s%-20s%-20s","|
","ETABLISSEMENT:",h_sigle_etb.arr);
  fprintf(fp,"%-20s%-20s"," DEPARTEMENT
:",h_sigle_departement.arr);
  fprintf(fp,"%-10s%-4s", "NIVEAU : ",h_niveau.arr);
  fprintf(fp,"%-10s%-17s","OPTION:  ",h_lib_option.arr);
  fprintf(fp,"%s","|\n");
  fprintf(fp,"%s%-40s","|      ","-------------");
  fprintf(fp,"%-40s"," ------------");
  fprintf(fp,"%-14s", "------- ");
```

```
 fprintf(fp,"%-40s","------");
  fprintf(fp,"%s","|");
  fprintf(fp,"%s","\n|");
  for(i=1;i==133;i=i+1)
    fprintf(fp,"%s"," ");
  fprintf(fp,"%s","|");
  fprintf(fp,"%s","\n|");

fprintf(fp,"%s","-----------------------------------------------------"
);

fprintf(fp,"%s","-----------------------------------------------------"
);

fprintf(fp,"%s","-----------------------------------------------------"
);
  fprintf(fp,"%s","----------");
  fprintf(fp,"%s","|\n");
  fprintf(fp,"| No ");
  fprintf(fp,"|      NOM & PRENOM(S)     ");
  fprintf(fp,"|MATRI");
  fprintf(fp,"|STAGE ");
  fprintf(fp,"|                    NOTES                    ");
  fprintf(fp,"| memo  ");
  fprintf(fp,"|total  ");
  fprintf(fp,"|MOY.GE");
  fprintf(fp,"|RES");
  fprintf(fp,"|CLASS");
  fprintf(fp,"|OBSERV|");
  fprintf(fp,"%s","\n|");
  fprintf(fp,"    ");
  fprintf(fp,"|                              ");
  fprintf(fp,"|     ");
  fprintf(fp,"|      ");
  fprintf(fp,"|-----------------------------------------");
  fprintf(fp,"|       ");
  fprintf(fp,"|       ");
  fprintf(fp,"|      ");
  fprintf(fp,"|   ");
  fprintf(fp,"|     ");
  fprintf(fp,"|      |");
  fprintf(fp,"%s","\n|");
   EXEC SQL DECLARE curs_cours CURSOR FOR
                              SELECT DISTINCT code_module
                          FROM mod_pgm
                             WHERE code_programme IN
                        (SELECT code_programme
                       FROM pv
                       WHERE num_pv=:h_num_pv)
          ORDER BY mod_pgm.code_module;
EXEC SQL WHENEVER NOT FOUND GOTO fin_curs_cours;
```

```
EXEC SQL OPEN curs_cours;
   i=0;
   fprintf(fp,"%s","     ");
   fprintf(fp,"%s","|                              ");
   fprintf(fp,"%s","|      ");
   fprintf(fp,"%s","|       ");
do
{
   EXEC SQL FETCH curs_cours INTO
                              :h_code_module;
   fprintf(fp,"%s%-5s","| ",h_code_module.arr);
   i=i+1;
}
while(1);
fin_curs_cours: EXEC SQL CLOSE curs_cours;
t=i;
while(t<6)
{
   fprintf(fp,"%s","|       ");
   t=t+1;
}
fprintf(fp,"|        ");
fprintf(fp,"|        ");
fprintf(fp,"|       ");
fprintf(fp,"|    ");
fprintf(fp,"|      ");
fprintf(fp,"|       |");
fprintf(fp,"\n");

fprintf(fp,"%s","-------------------------------------------------"
);

fprintf(fp,"%s","-------------------------------------------------"
);

fprintf(fp,"%s","-------------------------------------------------"
);
  fprintf(fp,"%s","----------");
  fprintf(fp,"%s","|\n");

}

/*******************************************************/
int errreur()
{
  printf("erreur sql:%s",sqlca.sqlerrm.sqlerrmc);
  exit(0);
}
```

```
/*********************************************************/
/*--------------PROCEDURE NOMORE-------------------------*/
/*********************************************************/


void NOMORE()

{
  fprintf(fp," \n");
}



/*********************************************************/
/*-----------------PROCEDURE AFFICHER_CORPS--------------*/
/*********************************************************/

void AFFICHER_CORPS()
{
int j,k;
float total;
cpt=1;
   EXEC SQL DECLARE curs_mat CURSOR FOR
        SELECT candidat.nom,
            candidat.prenom,
            etudiant.matricule
         FROM candidat,etudiant
      WHERE (candidat.code_candidat = etudiant.code_candidat)
      AND (etudiant.matricule IN (SELECT matricule
                       FROM ins_prog
                      WHERE num_pv =:h_num_pv))
   ORDER BY candidat.nom,candidat.prenom;

EXEC SQL WHENEVER NOT FOUND GOTO fin_curs;
EXEC SQL OPEN curs_mat;
do
{
   j=0;
   total=0.0;
   EXEC SQL FETCH curs_mat INTO
                              :h_nom_etudiant,
                              :h_prenom_etudiant,
                                 :h_mat;

   h_nom_etudiant.arr[h_nom_etudiant.len]='\0';
   h_prenom_etudiant.arr[h_prenom_etudiant.len]='\0';

   EXEC SQL WHENEVER NOT FOUND CONTINUE;
   EXEC SQL SELECT note_stage
           INTO :h_stage
           FROM ins_stage
          WHERE matricule =:h_mat;
```

```
 fprintf(fp,"%s%-3d","| ",cpt);
   cpt=cpt+1;
   fprintf(fp,"%s%-13s%s","|",h_nom_etudiant.arr," ");
   fprintf(fp,"%+12s",h_prenom_etudiant.arr);
   fprintf(fp,"%s%5s","|",h_mat.arr);
   fprintf(fp,"%s%6.2f","|",h_stage);
   EXEC SQL WHENEVER NOT FOUND GOTO fin_curs_cours;
   EXEC SQL OPEN curs_cours;
do
{
   EXEC SQL FETCH curs_cours INTO :h_code_module;
   EXEC SQL SELECT coefficient
         INTO  :h_coef
         FROM  mod_pgm
         WHERE (code_module=:h_code_module)
         AND (code_programme IN (SELECT code_programme
                        FROM pv
                        WHERE num_pv=:h_num_pv));
   EXEC SQL  SELECT DISTINCT moy_obt
          INTO :h_moy_obt
             FROM ins_cours
          WHERE (matricule =:h_mat)
          AND (code_module =:h_code_module);
   total=total + h_coef*h_moy_obt;
   fprintf(fp,"%s%6.2f","|",h_moy_obt);
   h_moy_obt=0.0;
   j=j+1;
} while(1);
fin_curs_cours:EXEC SQL CLOSE curs_cours;
EXEC SQL WHENEVER SQLERROR  CONTINUE;
k=j;
while(k<6)
{
   fprintf(fp,"%s","|      ");
   k=k+1;
}
EXEC SQL SELECT moy_obt,resultat_obt,rang
      INTO :h_moy_gene,
           :h_resultat_obt,
           :h_rang
         FROM ins_prog
      WHERE matricule =:h_mat
      AND num_pv=:h_num_pv;
   fprintf(fp,"|       ");
   fprintf(fp,"%s%6.2f","|",total);
   fprintf(fp,"%s%6.2f","|",h_moy_gene);
   fprintf(fp,"%s%4s","|",h_resultat_obt.arr);
   fprintf(fp,"%s%6d","|",h_rang);
   fprintf(fp,"|       | ");
   fprintf(fp,"\n");
   h_moy_gene=0.0;
   h_rang=0;
   h_resultat_obt.arr[0]='\0';
}while(1);
```

```
fin_curs: EXEC SQL CLOSE curs_mat;
       fprintf(fp,"\n ");
       return;
}



/*********************************************************/
/*----------------PROCEDURE BAS_DE_PAGE------------------*/
/*********************************************************/

void BAS_DE_PAGE()

{
 EXEC SQL WHENEVER NOT FOUND CONTINUE;
 EXEC SQL SELECT COUNT(DISTINCT matricule)
       INTO :nb_admis
       FROM ins_prog
       WHERE resultat_obt ='AD'
       AND  num_pv=:h_num_pv;
 EXEC SQL SELECT COUNT(DISTINCT matricule)
        INTO :nb_ajournes
        FROM ins_prog
        WHERE resultat_obt='AJ'
        AND   num_pv=:h_num_pv;
 EXEC SQL SELECT COUNT(DISTINCT matricule)
        INTO :nb_absent
        FROM ins_prog
        WHERE moy_obt is null
        AND  num_pv=:h_num_pv;
 cpt=cpt-1;
 nb_ajournes=nb_ajournes + nb_absent;
 nb_present=cpt-nb_absent;
 fprintf(fp,"%s","|");

fprintf(fp,"-----------------------------------------------------
");

fprintf(fp,"-----------------------------------------------------
");
 fprintf(fp,"-------------------------------------------");
 fprintf(fp,"-------");
 fprintf(fp,"%s","|");
 fprintf(fp,"\n");
 fprintf(fp,"%s%d","| Total inscrits: ",cpt);
 fprintf(fp,"%+40s%d","Admis:",nb_admis);
 fprintf(fp,"%+40s","Membres du jury");
 fprintf(fp,"\n");
 fprintf(fp,"%s%d","| Presents : ",nb_present);
 fprintf(fp,"%+40s%d","Ajournes:",nb_ajournes);
 fprintf(fp,"\n");
 fprintf(fp,"%s%d","| Absents:",nb_absent);
 fprintf(fp,"%+124s","\n|");
```

```
fprintf(fp,"-----------------------------------------------------
");

fprintf(fp,"-----------------------------------------------------
");
 fprintf(fp,"----------------------------------------");
 fprintf(fp,"-------");
 fprintf(fp,"%s","|");
}
```

STATISTIQUE

```
#include<stdio.h>

/*----------------declaration des variables hotes-------*/
EXEC SQL BEGIN DECLARE SECTION;
  VARCHAR username[20];
  VARCHAR password[20];
  VARCHAR h_niveau[3];
  VARCHAR h_code_pays[5];
  VARCHAR h_lib_etb[41];
  VARCHAR h_nom_pays[21];
  VARCHAR h_code_etb[3];
  VARCHAR h_annee[5];
  char code_etb[3];
  int verif;
  int test,h_nb_niveau;
  int nb_g_inscrits;
  int nb_f_inscrites;
  int t_inscrits_pays,T_inscrits_pays;
  int nb_g_admis;
  int nb_f_admises;
  int t_admis_pays,T_admis_pays;
  float p_admis_pays,p_T_admis_pays;
  int t_g_admis;
  int t_g_inscrits;
  int t_f_inscrites;
  int t_f_admises;
  int nb_f_inscrits;
  char annee[5];
EXEC SQL END DECLARE SECTION;
  FILE *fp;
  float T_AD,T_INS;
EXEC SQL INCLUDE SQLCA;
```

```
/*************************************************************/
/*************DECLARATION DES  PROCEDURES*******************/
/*************************************************************/

int CONNEXION();

void AFFICHER_CORPS();

void ENTETE();

void BAS_DE_PAGE();

int errreur();

/*************************************************************/
/*--------------PROCEDURE PRINCIPALE--------------------*/
/*************************************************************/
main()

{
char c;
if (CONNEXION() < 0)
   exit(-1);
else
{
   system("clear");
   fp=fopen("/dev/lp0","w");
   EXEC SQL DECLARE curs_etb CURSOR FOR
                    SELECT code_etablissement,libelle_etab
                    FROM etablissement;
   EXEC SQL OPEN curs_etb;
   do
   {
     printf("%s","ANNEE ACADEMIQUE:");
     scanf("%s",&annee);
   }while(annee[0]=='\0');
   strcpy(h_annee.arr,annee);
   h_annee.len=strlen(h_annee.arr);
   EXEC SQL WHENEVER NOT FOUND GOTO fin;
   do
   {
     EXEC SQL FETCH curs_etb INTO:h_code_etb,
                                  :h_lib_etb;
     h_lib_etb.arr[h_lib_etb.len]='\0';
     ENTETE();
     AFFICHER_CORPS();
     BAS_DE_PAGE();
   }while(1);
   fin:EXEC SQL CLOSE curs_etb;
   fclose(fp);
}
}
```

```
/*************************************************************/
/*******************CORPS DES PROCEDURES**********************/
/*************************************************************/


/*************************************************************/
/*------------PROCEDURE DE CONNECTION ORACLE-------------*/
/*************************************************************/

int CONNEXION()

{
  strcpy(username.arr,"system");
  username.len=strlen(username.arr);
  strcpy(password.arr,"manager");
  password.len=strlen(password.arr);
  EXEC SQL WHENEVER SQLERROR DO errreur();
  EXEC SQL CONNECT :username IDENTIFIED BY :password;

  return 0;
}



/*************************************************************/
/*-----------------PROCEDURE ENTETE--------------------*/
/*************************************************************/
void ENTETE()

{
  fprintf(fp,"\n\n\n\n");
  fprintf(fp,"%s%s"," ETABLISSEMENT:",h_lib_etb.arr);
  fprintf(fp,"%s%s","\n ANNEE ACADEMIQUE:",h_annee.arr);
  fprintf(fp,"%s","\n\n\n|");

fprintf(fp,"%s","-----------------------------------------------")
;

fprintf(fp,"%s","-----------------------------------------------"
);
  fprintf(fp,"%s","------------");
  fprintf(fp,"%s","|\n");
  fprintf(fp,"%s","|ANNEE D'ET|   1ere ANNEE
 ");
  fprintf(fp,"%s","|   2ere ANNEE                                ");
  fprintf(fp,"%s","|   3ere ANNEE                                ");
  fprintf(fp,"%s","|   4ere ANNEE                                |\n");


fprintf(fp,"%s","|-----------------------------------------------"
);

fprintf(fp,"%s","-----------------------------------------------"
);
```

```
 fprintf(fp,"%s","------------|");
  fprintf(fp,"\n");
  fprintf(fp,"%s","| PAYS     |    INSCRITS     |          ADMIS
 ");
  fprintf(fp,"%s","|    INSCRITS     |       ADMIS          ");
  fprintf(fp,"%s","|    INSCRITS     |       ADMIS          ");
  fprintf(fp,"%s","|    INSCRITS     |       ADMIS          |\n");

fprintf(fp,"%s","|-----------------------------------------------"
);

fprintf(fp,"%s","-----------------------------------------------"
);
  fprintf(fp,"%s","------------\n");
  fprintf(fp,"%s","|   SEXE   | G   | F   | T   | G  | F  |  T  | %
 ");
  fprintf(fp,"%s","| G   | F   | T   | G  | F  |  T  | %   ");
  fprintf(fp,"%s","| G   | F   | T   | G  | F  |  T  | %   ");
  fprintf(fp,"%s","| G   | F   | T   | G  | F  |  T  | %   |\n");

fprintf(fp,"%s","|-----------------------------------------------"
);

fprintf(fp,"%s","-----------------------------------------------"
);
  fprintf(fp,"%s","------------");
  fprintf(fp,"%s","|\n");
}

int errreur()
{
  printf("erreur sql:%s",sqlca.sqlerrm.sqlerrmc);
  exit(0);
}



/*********************************************************/
/*------------------PROCEDURE AFFICHER_CORPS------------*/
/*********************************************************/

void AFFICHER_CORPS()
{

EXEC SQL DECLARE curs_pays CURSOR FOR
  SELECT distinct code_pays
   FROM candidat
   WHERE code_candidat IN (SELECT code_candidat
                    FROM  etudiant
                    WHERE matricule  IN
                    (SELECT matricule
                     FROM  ins_prog
                     WHERE num_pv    IN
```

```
                        (SELECT num_pv
                         FROM  pv
                         WHERE code_programme IN
                         (SELECT code_programme
                          FROM programme
                          WHERE code_etablissement=:h_code_etb))))

   ORDER BY code_pays;

EXEC SQL OPEN curs_pays;



do
{
EXEC SQL WHENEVER NOT FOUND GOTO fin_curs_pays;

EXEC SQL DECLARE curs_niveau CURSOR FOR
      SELECT DISTINCT niveau
       FROM programme
       WHERE code_etablissement=:h_code_etb;

EXEC SQL OPEN curs_niveau;

EXEC SQL FETCH curs_pays INTO : h_code_pays;

EXEC SQL SELECT nom_pays
       INTO :h_nom_pays
       FROM pays
       WHERE code_pays=:h_code_pays;

fprintf(fp,"%s%10s","|",h_nom_pays.arr);
do
{
 t_inscrits_pays=0;
 t_admis_pays=0;
 nb_g_inscrits=0;
 nb_f_inscrites=0;
 nb_g_admis=0;
 nb_f_admises=0;
 p_admis_pays=0;
EXEC SQL WHENEVER NOT FOUND GOTO fin_curs_niveau;
EXEC SQL FETCH curs_niveau INTO :h_niveau;

EXEC SQL SELECT count(DISTINCT code_candidat)
       INTO :nb_g_inscrits
       FROM candidat
       WHERE code_pays=:h_code_pays
       AND  sexe='M'
       AND code_candidat IN
       (SELECT code_candidat
        FROM etudiant
```

```
          WHERE matricule IN
          (SELECT matricule
           FROM ins_prog
           WHERE num_pv IN
           (SELECT num_pv
            FROM pv
            WHERE annee=:h_annee
            AND  code_programme IN
            (SELECT code_programme

             FROM programme
             WHERE code_etablissement=:h_code_etb
             AND niveau=:h_niveau))));

t_g_inscrits = t_g_inscrits + nb_g_inscrits;

EXEC SQL SELECT count(DISTINCT code_candidat)
        INTO :nb_g_admis
        FROM candidat
        WHERE code_pays=:h_code_pays
        AND  sexe='M'
        AND code_candidat IN
        (SELECT code_candidat
         FROM etudiant
         WHERE matricule IN
         (SELECT matricule
          FROM ins_prog
          WHERE resultat_obt='AD'
          AND  num_pv IN
          (SELECT num_pv
           FROM pv
           WHERE annee=:h_annee
           AND   code_programme IN
           (SELECT code_programme
            FROM programme
            WHERE code_etablissement=:h_code_etb
            AND niveau=:h_niveau))));

t_g_admis=t_g_admis + nb_g_admis;

EXEC SQL SELECT count(DISTINCT code_candidat)
        INTO :nb_f_inscrites
        FROM candidat
        WHERE code_pays=:h_code_pays
        AND  sexe='F'
        AND code_candidat IN
        (SELECT code_candidat
         FROM etudiant
         WHERE matricule IN
         (SELECT matricule
          FROM ins_prog
          WHERE num_pv IN
          (SELECT num_pv
           FROM pv
```

```
          WHERE  annee=:h_annee
            AND   code_programme IN
            (SELECT code_programme
             FROM programme
             WHERE code_etablissement=:h_code_etb
             AND niveau=:h_niveau))));

t_f_inscrites=t_f_inscrites + nb_f_inscrites;

EXEC SQL SELECT count(DISTINCT code_candidat)
        INTO :nb_f_admises
        FROM candidat
        WHERE code_pays=:h_code_pays
        AND  sexe='F'
        AND code_candidat IN
        (SELECT code_candidat
         FROM etudiant
         WHERE matricule IN
         (SELECT matricule
          FROM ins_prog
          WHERE resultat_obt='AD'
          AND  num_pv IN
          (SELECT num_pv
           FROM pv
           WHERE  annee=:h_annee
           AND  code_programme IN
           (SELECT code_programme
            FROM programme
            WHERE code_etablissement=:h_code_etb
            AND niveau=:h_niveau))));

 t_f_admises = t_f_admises + nb_f_admises;

 t_inscrits_pays =  nb_g_inscrits + nb_f_inscrites;

 T_inscrits_pays=T_inscrits_pays + t_inscrits_pays;

 t_admis_pays = t_g_admis + t_f_admises;

 T_admis_pays=T_admis_pays + t_g_admis +t_f_admises;

 if(t_inscrits_pays!=0)
 {
   T_AD=t_admis_pays;
   T_INS=t_inscrits_pays;
   p_admis_pays = T_AD /T_INS ;
 }
 fprintf(fp,"%s%5d","|",nb_g_inscrits);
 fprintf(fp,"%s%5d","|",nb_f_inscrites);
 fprintf(fp,"%s%5d","|",t_inscrits_pays);
 fprintf(fp,"%s%4d","|",nb_g_admis);
 fprintf(fp,"%s%4d","|",nb_f_admises);
 fprintf(fp,"%s%5d","|",t_admis_pays);
 fprintf(fp,"%s%5.2f","|",p_admis_pays);
}while(1);
```

```
fin_curs_niveau:EXEC SQL CLOSE curs_niveau;
          fprintf(fp,"%s","|");
fprintf(fp,"\n");
}while(2);
fin_curs_pays:EXEC SQL CLOSE curs_pays;
EXEC SQL WHENEVER NOT FOUND CONTINUE;
}


/*********************************************************/
/*---------------PROCEDURE BAS_DE_PAGE-----------------*/
/*********************************************************/

void BAS_DE_PAGE()

{
   printf(fp,"|-----------------------------------------");
   fprintf(fp,"-----------------------------------------");
   fprintf(fp,"-------|\n");
   if(t_admis_pays!=0)
     {
     T_AD=t_inscrits_pays;
     T_INS=t_admis_pays;
     p_T_admis_pays=T_AD /T_INS ;
     }
   fprintf(fp,"%s","|");
   fprintf(fp,"%10s","  TOTAL");
   fprintf(fp,"%s%5d","|",t_g_inscrits);
   fprintf(fp,"%s%5d","|",t_f_inscrites);
   fprintf(fp,"%s%5d","|",T_inscrits_pays);
   fprintf(fp,"%s%4d","|",t_g_admis);
   fprintf(fp,"%s%4d","|",t_f_admises);
   fprintf(fp,"%s%5d","|",T_admis_pays);
   fprintf(fp,"%s%5.2f","|",p_T_admis_pays);
   fprintf(fp,"%s","|");
   fprintf(fp,"\n\n\n\n\n");
   t_g_inscrits=0;
   t_f_inscrites=0;
   T_inscrits_pays=0;
   t_g_admis=0;
   t_f_admises=0;
   T_admis_pays=0;
   p_T_admis_pays=0;
}
```

ANNEXES

annexe1.a.

MINISTERE
DES ENSEIGNEMENTS SECONDAIRE,
SUPERIEUR ET DE LA RECHERCHE
SCIENTIFIQUE

UNIVERSITE DE OUAGADOUGOU
No----------/UO/SG/SGS

BURKINA FASO

La Patrie ou la Mort,
Nous Vaincrons

ATTESTATION

Le secrétaire Général de l'UNIVERSITE DE OUAGADOUGOU,
atteste que .....
né(e) le .....à.....
matricule.....
a satisfait le .....
aux épreuves sanctionnant la .....année du .....cycle
de .....
et a ainsi obtenu le .....
Mention:.....
Option:.....
Appréciation:.....
En fois de quoi , il lui délivre la présente attestation pour servir et valoir ce que de droit.

OUAGADOUGOU, le .....

Le Secrétaire général

annexe1.b.

BORDEREAU DES ATTESTATIONS

Procès-Verbal no: ..... du ../../..

... année ... cycle de .....

no ordre	Matricule	Nom	Prénom(s)	Date retrait	signature

annexe2.

UNIVERSITE DE OUAGADOUGOU
DIRECTION DES AFFAIRES
ACADÉMIQUES & SCOLAIRES

SCOLARITE DE:********** SESSION DE:*

PROCES-VERBAL No:****

ETABLISSEMENT:***** DEPARTEMENT:****** NIVEAU:*** OPTION:***

No	Mat	NOM	PRENOM(S)	STAGE	NOTE s/	.	NOTE s/	MEMO	TL	MOY	RES	CLAS	ob

TOTAL inscrits:****** admis:******* Membres du jury

présents:******* ajournés:******

absents:********

annexe3.

UNIVERSITE DE OUAGADOUGOU
DIRECTION DES AFFAIRES
ACADÉMIQUES & SCOLAIRES

LISTE DES RESULTATS A ********

ord	Nom	Prénom	Date et lieu de naissance	Etat civil

Arrêter la présente liste à ....noms

OUAGADOUGOU le .....

Le président du jury.

annexe4.

UNIVERSITE DE OUAGADOUGOU
DIRECTION DES AFFAIRES ACADÉMIQUES
ET SCOLAIRES

LA PATRIE OU MORT,
NOUS VAINCRONS.

INFORMATIONS NECESSAIRES A LA CALLIGRAPHIE DE DIPLOMES

NOM : .....
PRENOM : .....
ETABLISSEMENT : .....
DEPARTEMENT : .....
FILIERE : .....
PROGRAMME DE FORMATION : .....
OPTION : .....
DATE DE NAISSANCE DE L'ETUDIANT : .....
LIEU DE NAISSANCE DE L'ETUDIANT : .....
SEXE DE L'ETUDIANT : .....
NUMERO P.V : ..... SESSION : .....
ANNEE DE CREATION P.V : .....

OUAGADOUGOU LE .....

annexe5

UNIVERSITE DE OUAGADOUGOU
DIRECTION DES AFFAIRES ACADEMIQUE ET SCOLAIRE

CURSUS UNIVERSITAIRE

NOM :.....
PRENOM :....
MATRICULE :....

ANNEE	SES-SION	ETAB	DEPT	NIV	LIB-PRG	RESUL-TAT	MEN-TION

OUAGADOUGOU LE ../../..
LE RECTEUR

annexe6

UNIVERSITE DE OUAGADOUGOU
DIRECTION DES AFFAIRES ACADÉMIQUES ET SCOLAIRES

SERVICE: INSCRIPTION ET REINSCRIPTION

LISTE DES ETUDIANTS NON A JOUR DE LEURS FRAIS D'INSCRIPTION

ETABLISSEMENT	DEPARTEMENT
**********	**********

NIV	MATRICULE	NOM	PRENOM	MT DU	MT PAYE	RESTE A VERSE
***	**********	***	******	******	*******	******
***	**********	***	******	******	*******	******
TOTAL DEPARTEMENT				******	******	******

.
.
.
.

ETABLISSEMENT	DEPARTEMENT
**********	**********

NIV	MATRICULE	NOM	PRENOM	MT DU	MT PAYE	RESTE A VERSE
***	**********	***	******	******	*******	******
***	**********	***	******	******	*******	******
TOTAL DEPARTEMENT				******	*******	******

.
.
.
.

TOTAL ETABLISSEMENT	******	*******	******
TOTAL GENERAL	******	******	******

annexe7

ETABLISSEMENT: ANNEE ACADEMIQUE:

ANNEE D'ETUD	1ère année							2ème année						
PAYS	INSCRITS			ADMIS				INSCRITS			ADMIS			
SEXE	G	F	T	G	F	T	%	G	F	T	G	F	T	%
TOGO	6	13	19	1	4	5		13	15	28	8	11	19	
BENIN		2	2						1	1		1	1	
.....														
.....														
TOTAL	6	15	21	1	4	5		13	16	29	8	12	20	

annexe8

ANNEE

Anu	4 caractères

ANU_COURANTE

Annee	4 caractères

ATTESTATION

Num_attestation	numérique
Num_diplome	numérique
Date_retrait	date

A_POUR_STATUT

Matricule	numérique
statut	caractère
Annee	4 caractères

BAC

Code_diplome_entree	2 caractères
Serie	caractère

CANDIDAT

Code_candidat	5 numériques
Nom	25 caractères
Prenom	40 caractères
Date_naiss	date
Lieu_naiss	20 caractères
Sexe	caractère
Adr_p	40 caractères
Code_origine_orig_socia	4 caractères
Code_ethnie	4 caractères
Code_etab_orig	4 caractères
Code_diplome_entree	2 caractères
Code_pays	4 numériques
Comments	40 caractères
Date_dip	date
Mention_dip	2 caractères
Etat_civil	80 caractères

CATEGORIE_DIP

Code_categorie_dip	2 caractères
Lib_cat_dip	30 caractères
Code_cycle	1 caractère

CODE_CANDIDAT_DISPONIBLES

Code_candidat	5 numériques

CTRL_CONNAIS

Code_ctrl_connais	4 caractères
Type_ctrl	1 caractère
Lib_ctrl	15 caractères
Poids	3 numériques
Code_module	7 caractères

CYCLE

Code_cycle	1 caractère
Lib_cycle	15 caractères

DDE_INSC

Code_programme	5 caractères
Code_candidat	5 numériques
Type_acces	1 caractère
Annee	4 caractères

DEM_ACC

Code_candidat	5 numériques
Code_programme	5 numériques
Annee	4 caractères
Type_acces	2 caractères
Validite	1 caractère
Nom	25 caractères
Prenom	40 caractères
Moyenne	5 numériques
Rang	4 numériques

DEPARTEMENT

Code_departement	3 numériques
Sigle_depart	10 caractères
Libelle_depart	40 caractères
Code_etablissement	2 numériques

DIPLOME

Num_diplome	7 numériques
Code_option	3 caractères
Code_filiere	6 numériques
Matricule	7 numériques
Anu	4 caractères

DIPLOME_ENTREE

Code_diplome	2 caractères
Nom_dip	20 caractères

DIPLOME_SUPPRIME

Num_diplome	7 numériques

DISPENSE_COURS

Num_intervenant	5 numériques
Code_module	7 caractères
Code_programme	5 caractères
Annee	4 caractères
Niveau	2 numériques
Nbre_heures	3 numériques
Cumul_realisation	3 numériques
Nature_dispense	2 caractères

DISPENSE_ENCADREMENT

Num_intervenant	5 numériques
Code_module	7 caractères
Code_programme	5 caractères
Annee	4 caractères
Niveau	2 numériques
Nbre_heures	3 numériques
Cumul_realisation	3 numériques
Nature_dispense	2 caractères

DISPENSE_TP_TD

Num_intervenant	5 numériques
Code_module	7 caractères
Code_programme	5 caractères
Annee	4 caractères
Niveau	2 numériques
Nbre_heures	3 numériques
Cumul_realisation	3 numériques
Nature_dispense	2 caractères

ENSEIGNANT

Num_intervenant	5 numériques
Code_departement	3 numériques

EPOUSE

Code_candidat	5 numériques
Nom_jf	20 caractères

EQUIVAL_MOD

Code_module	7 caractères
Matricule	7 numériques
Date_obt	date

EQUIVAL_PGME

Matricule	7 numériques
Date_equiv	date
Code_programme	5 caractères

EST_SIGNE

Num_intervenant	5 numériques
Num_pv	6 caractères

ETABLISSEMENT

Code_etablissement	2 numériques
sigle_etab	10 caractères
Libelle_etab	40 caractères

ETAB_ORIG

Code_etab_orig	4 caractères
Nom_sup	20 caractères

ETAB_SECOND

Code_etab_second	4 caractères
Nom_etab_second	20 caractères

ETHNIE

Code_ethnie	4 caractères
Nom_ethnie	20 caractères

ETUDIANT

Matricule	7 numériques
Prof_pere	20 caractères
Prof_mere	20 caractères
Etab_second	25 caractères
Adr_etudiant	40 caractères
Situ_mat	1 caractère
Code_candidat	5 numériques

FILIERE

Code_filiere	6 numériques
Sigle_filiere	10 caractères
Lib_filiere	40 caractères
Dip_autre_fac	1 caractère
code_departement	3 numériques
Code_categorie_dip	2 caractères
Intitule_diplome	40 caractères

FRAIS

No_quit	8 numériques
Mt_quit	8 numériques
Date_quit	date
Nat_frais	2 caractères
Matricule	7 numériques
Anu	4 caractères
Code_departement	3 numériques

INS_CONTROLE

Num_pv	6 caractères
Matricule	7 numériques
Code_ctrl_connais	4 caractères
date_ins	date
Note_ctrl	5 numériques

INS_COURS

Num_pv	6 caractères
MATRICULE	7 numériques
Code_module	7 caractères
Date_insc	date
Moy_obt	5 numériques

INS_PROG

Num_pv	6 caractères
Matricule	7 numériques
Date_ins_prog	date
Etat_financier	numérique
Etat_insc	numérique
Validation	1 caractère
Mt_ins_prog	8 numériques
Mode_paie	1 caractère
Moy_obt	5 numériques
Resultat_obt	2 caractères
Rang	4 numériques
Mention	2 caractères

INS_STAGE

Num_pv	6 caractères
Matricule	7 numériques
Code_module	7 caractères
Code_ins_stage	4 caractères
Date_insc	date
Note_stage	5 numériques

INTERVENANT

Num_intervenant	5 numériques
Nom_interv	25 caractères
Prenom_interv	40 caractères
Adr_interv	40 caractères
Code_titre_fonction	3 caractères

JURY_THESE

Num_intervenant	5 numériques
Code_these	4 caractères

MEMBRE_JURY

Num_intervenant	5 numériques

MODULE

Code_module	7 caractères
Intitule_module	40 caractères
Type_module	1 caractère
Localisation	40 caractères
Heures_cours	3 numériques
Heures_td	3 numériques
Heures_tp	3 numériques
Heures_encadre	3 numériques
Examen_term_session1	1 caractère
Examen_term_session2	1 caractère
Ctrl_cont_oblig	1 caractère
Date_creation	date
Date_modif	date
Date_fin_org	date
Code_etab_gestionnaire	2 numériques

MOD_PGM

Code_programme	5 caractères
Code_module	7 caractères
Obligation	1 caractère
Coefficient	5 numériques

MOIS

Num_mois	2 numériques
Libelle_mois	15 caractères

NIVEAU

Niveau	2 numériques

NOTES_ACCES

Annee	4 caractères
Type_acces	1 caractère
Code_programme	5 caractères
Code_module	7 caractères
Code_candidat	5 numériques
Note	5 numériques

NUM_DIPLOME_DISPONIBLE

Num_diplome	7 numériques

OPTIONS

Code_option	3 caractères
Lib_option	30 caractères
Code_filiere	6 numériques

ORIG_SOCIA

Code_origine	4 caractères
Nom_orig	20 caractères

PARCHEMIN

Num_parchemin	7 numériques
Num_diplome	7 numériques
Date_retrait	date
Date_arrivee	date
Etat	1 caractère

PARCHEMIN_CREER

Num_diplome	7 numériques

PAYS

Code_pays	4 numériques
Nom_pays	20 caractères
Nationalite	20 caractères

PREREQUIS

Code_est_prerequis	5 caractères
Code_requiert	5 caractères
Commentaires	50 caractères

PROGRAMME

Code_programme	5 caractères
Annee_creation	4 caractères
Annee_modif	4 caractères
Annee_fin_org	4 caractères
Access_travailleur	1 caractère
Form_continue	1 caractère
Ens_distance	1 caractère
Donne_dip	1 caractère
Possibilite_conc	1 caractère
Possibilite_es	1 caractère
Possibilite_dossier	1 caractère
Apres_bac	1 caractère
Perequis_univ	1 caractère
Type_prog	1 caractère
Nb_max_ins	2 numériques
Code_cycle	1 caractère
Code_option	3 caractères
Niveau	2 numériques
Code_filiere	6 numériques
Libelle_programme	30 caractères
Code_departement	3 numériques
Code_etablissement	2 numériques
Sigle_departement	10 caractères
Sigle_etablissement	10 caractères
Etat	1 numérique

PV

Num_pv	6 caractères
Date_creation_pv	date
Date_signe_pv	date
Etat	1 numérique
Ses_sion	1 caractère
Code_programme	5 caractères
Code_departement	3 numériques
Code_etablissement	2 numériques
Annee	4 caractères

PV_VISE

Num_diplome	7 numériques
Num_pv	6 caractères

REALISATION

Num_intervenant	5 numériques
Code_module	7 caractères
Code_programme	5 caractères
Nature_intervention	2 caractères
Num_mois	2 numériques
Realisation	3 numériques
Annee	4 caractères

RESULTATS_ACCES

Code_candidat	5 numériques
Code_programme	5 caractères
Annee	4 caractères
Type_acces	1 caractère
Moyenne	5 numériques
Rang	4 numériques
Resultat	2 caractères

SECURITE_DAAS

Nom_log	25 caractères
Pswd_log	25 caractères
Group_num	2 numériques
Niveau_priorite	2 numériques
Num_entree	2 numériques

SEQUENCES

Matricule	7 numériques

SES_SION

Ses_sion	1 caractère

STATUT

Statut	1 caractère

STAT_MODULE

Anu	4 caractères
Code_module	7 caractères
Nb_insc_session1	4 numériques
Nb_admis_session1	4 numériques
Nb_ins_session2	4 numériques
Nb_admis_session2	4 numériques

STAT_PGME

Code_programme	5 caractères
Anu	4 caractères
Nb_dem_acces	4 numériques
Nb_acces_conc	4 numériques
Nb_acces_es	4 numériques
Nb_acces_dossier	4 numériques
Nb_acces_DOB	4 numériques
Nb_ins_session1	4 numériques
Nb_admis_session1	4 numériques
Nb_ins_session2	4 numériques
Nb_admis_session2	4 numériques

THESE

Code_these	4 caractères
Sujet	20 caractères
Num_diplome	7 numériques

TITRE_FONCTION

Code_titre_fonction	3 caractères
Lib_titre	20 caractères
Titre_ens_sup	1 caractère

CONCLUSION

Nous arrivons à fin de ce stage avec une connaissance plus élargie, surtout en technique de programmation.Ce stage nous aurait permis d'apprendre beaucoup de choses que l'on ne peut connaître que par la pratique.Après avoir braver certaines difficultés inérantes au fait que nous faisons nos premiers pas dans le monde de la programmation, le reste du stage nous a été très instructif à bien d'égards.C'est pourquoi le sentiment qui nous anime actuellement est celui de tout Homme amoureux du savoir et de la connaissance quand il a appris.Et c'est avec une grande joie que remercions ceux qui ont été dévouement et par leur dispostion à montrer à tous ceux désireux apprendre le savoir immense qu'ils possèdent.Il s'agit notamment de nos aînés dont nous nous ferons le plaisir de nommer ici:

M.WALBEOGO Nagoukoamba
M.BILA Abdoulaye
M.TAPSOBA Stéphane

Le plaisir est également pour nous de remercier notre maître de stage M.NOMBRE Lassané et notre superviseur M.ZONGO Sylvain qui ont veillé à ce que nous soyons dans de bonnes conditions de travail.

Un remerciement particulier est adressé au chef de centre de l'Ecole Supérieure d'Informatique(ESI) M.CONGO Felix pour la compréhension dont il a fait preuve et pour ce qu'il nous a été donné d'apprendre avec lui.