Exercice 1 : (4 pts)

a) La fonction f est de type **Booléen** (car elle reçoit le résultat de l'évaluation de l'expression logique x=T[i]

- Le type est booléen ou logique (1pt pour le type + 0,75pt justif.)
- On accepte le type réel avec pour b) 0,1,1

b) f(2, 8 , T) = **faux** (0,75 pts)
f(5 , 8 , T) = **vrai** (0,75 pts)
f(100 , 8 , T) = **vrai** (0,75 pts)

- Toutes les formes de TRUE/ FALSE sont acceptées (féminin, masculin, français ou anglais, V/F, OUI/NON)
- La réponse 0/1 sera acceptée avec une correspondance entre, 0 et False et 1 et True même si dans a) le type booléen n'est pas mentionné.
- La réponse 0/1 sera acceptée avec un type de fonction réel

Exercice2 :

a) Analyse de la fonction Catalan (2 pts)

- Cette fonction aura un seul paramètre : epsilon

Résultat = Catalan
b) Catalan ← K
c) K = [K← 0, i ← -1, s ← -1] Répéter

i ← i + 1
terme ← $1/(2.i+1)^2$
s ← -s
K ← K + s.terme
Jusqu'à (terme < epsilon)

Fin catalan

- On accepte pour l'analyse : toute forme de stratégie de résolution correcte. (Grille d'analyse, spécification, pseudo code, analyse ascendante ou descendante)
- On accepte pour la multiplication : */ x / .
- On accepte pour le calcul du carré : 2, sqr(), carré()
- On accepte toute solution équivalente : valeur absolue et différence

Points clés de l'analyse	Barème
Affectation du résultat de la fonction	0.25
Choix de la structure répétitive avec sa condition Si solution récursive	0.5 (0.25 + 0.25) 0.25 (pour la condition de sortie)
Initialisation	0.25
Calcul du terme	0.25
Signe	0.25
Cumul	0.25
Progression (i)	0.25

d) Algorithme (2 pts -0.25/faute, pas de note négative)

0) Def FN catalan(epsilon : réel) : réel
1) [K ← 0, i ← -1, s ← -1] Répéter
I ← I + 1

terme ← $1/(2.i + 1)^2$
s ← -s
K ← K + s.terme
Jusqu'à (terme <epsilon)

2) Catalan ← K
3) Fin catalan

- L'évaluation de l'algorithme se fera indépendamment de celle de l'analyse.
- On accepte pour l'entête de l'algorithme : DEF Fonction, Function, Fonction
- Version récursive : Rassembler les points qui permettent de définir le problème sous la forme récursive (le point d'arrêt, l'initialisation, le traitement, l'appel)

Problème : (7 pts)

Points clés de l'analyse	Barème
Programme principal (Modularité et cohérence de la solution)	1 + 0.5
Tri du tableau	1
Recherche des rangs	1
Calculs de la moyenne	0.5
Calcul des 3 notes maximales	0.5
Calcul des 3 nombres d'élèves	0.5
Ouverture, assignation, parcours, écriture, lecture, fermeture des fichiers.	1.5 (6 x 0.25)
TDO	0.5

Traitement	Points clés du traitement	Barème
Programme principal (1.5)	Modularité + cohérence	1 + 0.5
Création fichier resultat.dat (3)	• Association et ouverture • Lecture des données de concours et écriture dans resultat.dat • Calcul des moyennes • Détermination du rang • Tri	• 0.5 • 0.5 • 0.5 • 0.75 • 0.75
Création du fichier details.txt (2.25)	• Association et ouverture • Calcul du nombre d'admis • Calcul de maxnote1, nb1 • Calcul de maxnote2, nb2 • Calcul de maxnote3, nb3 • Ecriture sur le fichier details.txt	• 0.25 • 0.5 • 0.25 • 0.25 • 0.25 • 0.5
TD des Objets et des types		0.5

- La saisie du fichier concours.dat, n'est pas notée (car non demandée)

1°) Analyse du programme principal (pts)

Résultat : (resultat.dat , details.txt)
Details.txt=associer(details, "c:\details.txt")
Details= Proc Detailler(result,details)
Resultat.dat=associer(result, "c:\resultat.dat")
Result=Proc Resultats(concours,result)
Concours= associer(concours, "c:\concours.dat")

2°) Analyse des modules envisagés

Analyse de la procédure Detailler (pts)

```
Resultat : f_details
f_details=[recréer (f_details)]
            Ecrire_nl(f_details, "nombre d'admis = ", n_admis)
            Ecrire_nl(f_details, maxnote1," " , nb1)
            Ecrire_nl(f_details, maxnote2," " , nb2)
            Ecrire_nl(f_details, maxnote3," " , nb3)
          Fermer(f_details)
n_admis = [ ouvrir(f_result)  ;  n_admis ← 0 ] Tant que non(fin fichier(f_result)) Faire
                                                [Lire(f_result,cand)] Si (cand.moyenne≥10) Alors
                                                        n_admis←n_admis+1
                                                Fin Si
                                            FinTantQue

(maxnote1, maxnote2, maxnote3)=[ pointer(f_result, 0) ;Lire(f_result, cand) ;maxnote1←cand.note1
                                    maxnote2←cand.note2 ; maxnote3←cand.note3]
                                    Tant que (NON (fin fichier(f_result))) Faire
                                            Lire(f_result,cand)
                                            Si cand.note1>maxnote1  Alors
                                                    maxnote1 ← cand.note1
                                            Fin Si
                                            Si cand.note2>maxnote2  Alors
                                                    maxnote2 ← cand.note2
                                            Fin Si
                                            Si cand.note3>maxnote3  Alors
                                                    maxnote3 ← cand.note3
                                            Fin Si
                                    FinTantQue
(nb1,nb2,nb3)=[ pointer(f_result,0) ; nb1←0 ; nb2← 0 ; nb3←0]
                                    Tant que NON (fin fichier(f_result)) Faire
                                            Lire(f_result,cand)
                                            Si cand.note1>=10  Alors
                                                    nb1 ←nb1+1
                                            Fin Si
                                            Si cand.note2>=10  Alors
                                                    nb2 ←nb2+1
                                            Fin Si
                                            Si cand.note3>=10  Alors
                                                    nb3 ←nb3+1
                                            Fin Si
                                    FinTantQue
                                    Fermer(f_result)
```

Analyse de la procédure resultats (pts)

```
Resultat : f_result
f_result=[recréer(f_result)]   Pour  i  de 1 à n faire
                                       Ecrire(f_result,Tf[i])
                                Fin Pour
Tf= [ Tt[1].rang  ← 1 ]        Pour i de 2 à n Faire
                               Si Tt[i].moyenne=Tt [i-1].moyenne Alors
                                       Tt [i].rang←Tt [i-1].rang
                               Sinon
                                       Tt [i].rang←i
                               FinSi
                          Fin Pour
Tt=Proc Tri (n,T)
( n , T )  =  [ouvrir (f_concours) ; n ←0 ] Tant que NON (fin fichier (f_concours)) faire
                                                n ← n+1
                                                Lire(f_concours,cand)
                                                T[n].Num←cand.Num
                                                T[n].Nom←cand.Nom
                                                T[n].Prenom←cand.Prenom
                                                T[n].matiere1←cand.matiere1
                                                T[n].note1←cand.note1
                                                T[n].matiere2←cand.matiere2
                                                T[n].note2←cand.note2
                                                T[n].matiere3←cand.matiere3
                                                T[n].note3←cand.note3
                                                T[n].moyenne ←
                                                (cand.note1+2*cand.note2+2*cand.note3)/5
                                        FinTant Que
                                        Fermer (f_concours)
```

Analyse de la procédure tri (pts)

```
Résultat : Ttrié
Ttrié= [      ] Répéter
                Permut ← faux
                [i ←0] Répéter
                        [i ← i + 1]Si (T[i] .moyenne< T[i+1].moyenne)  Alors
                                                Aux ← T[i]
                                                T[i] ← T[i+1]
                                                T[i+1] ← Aux
                                                Permut ← vrai
                                        FinSi
                     Jusqu'à  (i=n-1)
                n ← n-1
        Jusqu'à permut = faux
```

3°) **Les algorithmes** (5pts -0.25/faute sans note négative et sans pénaliser 2 fois la même faute)

Algorithme du Programme principal

	Points clés de l'analyse	Barème
	Programme principal	1
Création du fichier resultat.dat (2.25)	Lecture de concours.dat	0.5
	Calcul de la moyenne	0.5
	Tri	0.5
	Détermination du rang	0.5
	Ecriture dans le fichier resultat.dat	0.25
Création du fichier details.txt (1.75)	Lecture du fichier resultat.dat	0.25
	Calcul des 3 notes maximales et des 3 nombres	0.75
	Calcul du nombre d'admis	0.5
	Ecriture dans le fichier détails.txt	0.25

0) Début PP
1) Associer(concours, "c:\concours.dat")
2) Associer(result, "c:\resultat.dat")
3) Proc Resultats(concours,result)
4) Associer(details, "c:\details.dat")
5) Proc Detailler(result,details)
6) Fin

Tableau de déclaration des nouveaux types globaux

Type
Enreg1 = enregistrement Num: entier Nom : chaîne[30] Prenom : chaîne[30] Matière1:chaîne[20] Note1 :réel Matière2:chaîne[20] Note2 :réel Matière3:chaîne[20] Note3 :réel Fin enreg1 Enreg2 = enregistrement Num: entier Nom : chaîne[30] Prenom : chaîne[30] Matière1 : chaîne[20] Note1 : réel Matière2 : chaîne[20] Note2 : réel Matière3 : chaîne[20] Note3 : réel Moyenne : réel Rang :entier Fin enreg2 Tfile1 = fichier de enreg1

Tfile2= fichier de enreg2

Tableau de déclaration des objets globaux

Objet	Type/Nature	Rôle
Concours	Tfile1	Fichier contenant les données initiales
Result	Tfile2	Fichier des résultats
Detail	text	Fichier texte qui va contenir les détails

Algorithme de la procedure Detailler

```
0) Debut PROC Detailler (var f_result : Tfile2  ; var f_Details : text)
1) [Ouvrir (f_result)  ;  n_admis ← 0]Tant que non (fin fichier(f_result)) Faire
                                        [f_result,cand)]  Si  (cand.moyenne≥10) Alors
                                                              n_admis←n_admis+1
                                                          FinSi
                                    Fin Tant que
2) [pointer(f_result,0) ; Lire(f_result,cand) ;
   maxnote1←cand.note1  ;  maxnote2←cand.note2  ;  maxnote3←cand.note3]
   Tant que non (fin fichier(f_result)) Faire
         Lire(f_result,cand)
         Si cand.note1>maxnote1  Alors
               maxnote1 ← cand.note1
         FinSi
         Si cand.note2>maxnote2  Alors
               maxnote2 ← cand.note2
         FinSi
         Si cand.note3>maxnote3  Alors
               maxnote3 ← cand.note3
         FinSi
   Fin Tant que
3) [pointer(f_result,0)  ;  nbmax1←0  ;  nbmax2← 0  ;  nbmax3←0]
   Tant que non (fin fichier(f_result)) Faire
         Lire(f_result,cand)
         Si cand.note1 >= 10  Alors
               nbmax1 ←nbmax1+1
         Fin Si
         Si cand.note2 >= 10  Alors
               nbmax2 ←nbmax2+1
         Fin Si
         Si cand.note3 >=10  Alors
               nbmax3 ←nbmax3+1
         Fin Si
   Fin Tant que
   Fermer (f_result)
4) recréer (f_details)
   Ecrire_nl(f_details, "nombre d'admis = ",n_admis)
   Ecrire_nl(f_details, maxnote1,"  " ,nb1)
```

Ecrire_nl(f_details, maxnote2," " ,nb2)
Ecrire_nl(f_details, maxnote3," " ,nb3)
Fermer(f_details)

5) Fin Detailler

Tableau de déclaration des objets de la procédure Detailler

Objet	Type/Nature	Rôle
cand	Enreg2	Permet de saisir un enregistrement du fichier f_result
n_admis	Entier	Permet de déterminer le nombre d'admis
maxnote1	Réel	Permet de déterminer la note1 maximale
maxnote2	Réel	Permet de déterminer la note2 maximale
maxnote3	Réel	Permet de déterminer la note3 maximale
nb1	Entier	Permet de déterminer le nombre de candidats ayant une note1 >= 10
nb2	Entier	Permet de déterminer le nombre de candidats ayant une note2 >= 10
nb3	Entier	Permet de déterminer le nombre de candidats ayant une note2 >= 10

Analyse de la procédure Resultats

0) Début Proc resultats(var concours : Tfile1 ; var f_result : Tfile2)
1) [ouvrir (f_concours) ; n ←0] Tant que NON (fin fichier (f_concours)) faire
Lire(f_concours,cand)
n ← n+1
T[n].Num←cand.Num
T[n].Nom←cand.Nom
T[n].Prenom←cand.Prenom
T[n].matiere1←cand.matiere1
T[n].note1←cand.note1
T[n].matiere2←cand.matiere2
T[n].note2←cand.note2
T[n].matiere3←cand.matiere3
T[n].note3←cand.note3
T[n].moyenne ← (cand.note1+2*cand.note2+2*cand.note3)/5
FinTant Que
Fermer (F_concours)
2) Proc Tri (n,T)
3) [T[1].rang ← 1]Pour i de 2 à n Faire
Si T[i].moyenne=T[i-1].moyenne Alors
T[i].rang←t[i-1].rang
Sinon
T[i].rang←i
FinSi
Fin Pour
4) Recréer(F_result)
Pour i de 1 à n faire
Ecrire(F_result,T_f[i])
Fin Pour
5) Fin Resultats

Tableau de déclaration des nouveaux types de la procédure Resultats

Type
Vect = tableau de enreg2

Tableau de déclaration des objets de la procédure Resultats

Objet	Type/Nature	Rôle
Cand	Enreg1	Permet de saisir un enregistrement du fichier F_concours
T	Vect	Tableau pour extraire et trier les enregistrements du fichier F_result
N	Entier	Nombre total d'enregistrements dans le fichier F_result
i	Entier	compteur

Algorithme de la procédure Tri

```
0) Début Proc Tri (n : entier ; var T : Vect)
1) Répéter
        Permut ← faux
        i ←0
        Répéter
            i ← i + 1
            Si T[i] .moyenne< T[i+1].moyenne  Alors
                        Aux ← T[i]
                        T[i] ← T[i+1]
                        T[i+1] ← Aux
                        Permut ← vrai
            Fin Si
        Jusqu'à ( i=n-1)
        n ← n-1
   Jusqu'à (permut = faux)
2) Fin Tri
```

Tableau de déclaration des objets de la procédure Tri

Objet	Type/Nature	Rôle
permut	booléen	Variable pour vérifier s'il a eu une permutation
aux	enreg2	Variable auxiliaire pour permuter le contenu de deux cases
i	entier	Compteur