Méthodes Numériques et Informatiques (MP050)
Examen de TP du 23 juin 2010

Calculatrices et documents autorisés
Les deux parties sont indépendantes.
Les questions indépendantes sont signalées par le symbole ⋆

I Unix

– Pour chacune des questions unix, indiquer les commandes permettant de répondre.
– Dans toutes les réponses exprimées en UNIX, indiquer explicitement **tous** les blancs par le symbole ␣

Le fichier `im_auto.txt` (voir III.A, page 4) répertorie les voitures les plus immatriculées en France au cours des années 2008 et 2009. Chaque ligne contient les informations relatives à un modèle de voiture. Le fichier est organisé en 6 colonnes :

1. modèle
2. marque
3. gamme
4. nationalité du constructeur (`DEU` : Allemande, `FRA` : Française, `ITA` : Italienne, `JPN` : Japonaise, `USA` : Américaine)
5. nombre d'immatriculations en 2008
6. nombre d'immatriculations en 2009

⋆ **Q 1 :** Vous êtes dans votre répertoire utilisateur. Créer un sous-répertoire nommé `automobile/`, puis faites-en votre répertoire de travail. Y copier le fichier `im_auto.txt` disponible dans le répertoire `/home/lefrere/M1/2009-2010/etu/mni/exam2/`

⋆ **Q 2 :** Quel est le nombre de modèles de voitures référencés dans le fichier `im_auto.txt` ?

⋆ **Q 3 :** Quel est le nombre de modèles de voitures françaises ?

⋆ **Q 4 :** Faire la liste des 5 modèles les plus vendus en 2009, classés par nombre d'immatriculations décroissant.

⋆ **Q 5 :** Afficher les 3 berlines les plus vendues en 2009.

⋆ **Q 6 :** Afficher les 10 modèles les plus vendus en 2008, classés par ordre alphabétique de marque.

⋆ **Q 7 :** Créer le fichier `hors_europe.txt` répertoriant les modèles non européens (l'usage de plusieurs commandes successives est possible).

⋆ **Q 8 :** Afficher les 2 modèles de voitures étrangères les plus vendus en 2008.

⋆ **Q 9 :** Créer un fichier `bilan.txt` contenant les 6 colonnes du fichier `im_auto.txt` plus une 7ième colonne indiquant pour chaque modèle le nombre total d'immatriculations sur les années 2008 à 2009.

⋆ **Q 10 :** Afficher les modèles dont le nombre d'immatriculations a progressé entre 2008 et 2009.

⋆ **Q 11 :** Calculer combien de voitures françaises ont été immatriculées en 2009.

Q 12 : En utilisant le fichier `bilan.txt`, afficher le nom et seulement le nom du constructeur du modèle le moins vendu sur l'ensemble de la période 2008-2009.

⋆ **Q 13 :** Parmi les modèles de marque Renault, quel est le classement de la Modus en 2009 ?

II Un jeu de hasard

– Traiter les questions de programmation en choisissant soit le Fortran, soit le langage C et l'indiquer sur votre copie.
– **Préciser en commentaire de tous vos programmes vos nom, prénom et login.**
– Créer un répertoire de travail spécifique appelé `ex-mni-ses2` (sous votre répertoire d'accueil) et en faire votre répertoire de travail pour tous les programmes en C ou fortran.

On cherche dans cet exercice à étudier le gain d'un joueur à un jeu de hasard se déroulant en plusieurs tirages (comme par exemple le jeu de la roulette). On note g_n le gain du joueur après le tirage n (par définition, $g_0 = 0$). Le passage de g_n à g_{n+1} est simulé numériquement par le tirage (au hasard) d'un nombre δ issu une variable aléatoire réelle Δ (l'incrément de gain). Cette variable aléatoire aura différentes lois de probabilité au cours de l'exercice.
La simulation d'une partie de ce jeu se déroule donc selon l'algorithme suivant :

i on part de $g_0 = 0$

ii on tire au hasard δ

iii on calcule le gain suivant : $g_1 = g_0 + \delta$

iv on itère sur les deux points précédents pour construire l'évolution du gain au cours de la partie.

De manière plus générale, cette classe de modélisation, appelée marche aléatoire continue, permet de simuler de nombreux phénomènes comme, par exemple, la diffusion en physique ou l'évolution du cours d'une action en bourse.

Remarque : La séquence pseudo-aléatoire fournie par les générateurs appelés dans les deux langages est la même à chaque exécution ; on ne s'étonnera donc pas du caractère répétitif du jeu, manifeste tant que l'on ne traite qu'une seule partie (II.A, II.B et II.C).
Les prototypes **incomplets** des fonctions demandées sont indiquées en annexe III.B, page 5 pour le langage C, ou III.C, page 5 pour le langage fortran.
En fortran on placera toutes les procédures dans un module.

II.A Préliminaire avec tirage uniforme

⋆ **Q 14 :** Écrire, dans un fichier nommé `marche.c` ou `marche.f90`, un programme demandant à l'utilisateur le nombre total `nbtir` de tirages du jeu, et déclarer un tableau `gain` contenant `nbtir` éléments (à l'exécution, l'utilisateur choisira `nbtir = 100`).

⋆ **Q 15 :** Écrire une fonction nommée `tirage_uni` renvoyant **un** réel issu d'une variable distribuée uniformément entre -1 et $+1$. Pour cela, on s'aidera :
– en C, de la fonction `rand()` qui renvoie un **entier** distribué uniformément entre `0` et `RAND_MAX`
– en fortran, de la subroutine `random_number(x)` qui affecte au réel `x` un nombre aléatoire distribué uniformément entre `0` et `1`

Q 16 : Construire, dans le programme principal, l'évolution du gain en appelant plusieurs fois la fonction `tirage_uni` (dans les deux langages, il suffit d'écrire `tirage_uni()` dans une expression pour appeler cette fonction sans argument). On affichera à chaque nouveau tirage le gain g_n du joueur pour vérifier le comportement du programme. (On pourra dans la suite de l'exercice, si cela s'avère plus confortable, passer en commentaire les lignes de code correspondant à l'affichage.)

II.B Temps de sortie du jeu

Le joueur adopte la stratégie suivante : si son gain devient « très » négatif (il a perdu trop d'argent), ou « très » positif (il a atteint son objectif), il quitte le jeu. On cherche donc dans cette question à connaître le premier tirage pour lequel la suite g_n sort de l'intervalle $[-S; S]$, où S est le « seuil » que s'est fixé le joueur pour quitter le jeu (tant en positif qu'en négatif). On appelera par la suite ce tirage le « temps de sortie ».

⋆ **Q 17 :**
– Demander, dans le programme principal, la valeur de S à l'utilisateur. (À l'exécution, on choisira `3.0`)
– Écrire une fonction `tps_sortie` qui renvoie le premier entier `p` tel que $|g_p| > S$. Si aucune valeur ne correspond à ce critère, la fonction renverra le nombre `nbtir + 1`
– Afficher, dans le programme principal, la valeur du temps de sortie.

II.C Cas du tirage gaussien

⋆ **Q 18 :** Écrire une nouvelle fonction de tirage aléatoire appelée `tirage_nor` permettant de simuler une variable aléatoire gaussienne centrée réduite, c'est-à-dire de moyenne nulle et de variance 1. Pour cela, on utilisera l'algorithme suivant :
- si X_1 et X_2 sont deux variables aléatoires réelles indépendantes, de distribution uniforme dans $]0;1]$,
- alors la variable $G = \cos(2\pi X_2)\sqrt{-2\ln X_1}$ est une variable aléatoire gaussienne centrée réduite.

Q 19 : Dans le programme principal, passer en commentaire l'appel de `tirage_uni`, et ajouter une ligne pour appeler désormais la fonction `tirage_nor`. Exécuter le reste du programme avec ce nouveau tirage gaussien.

II.D Simulation de plusieurs parties

Copier le programme précédent dans un nouveau fichier nommé `marche2.c` ou `marche2.f90`.

Q 20 : On cherche maintenant à construire plusieurs simulations de parties au sein du programme.
- Passer en commentaire, dans ce nouveau fichier, les lignes du **programme principal** en lien avec le calcul du temps de sortie.
- Demander au début du programme à l'utilisateur le nombre de parties `nbpar` à construire.
- Modifier également l'allocation du tableau `gain` qui sera désormais un tableau 2D, dans lequel la première dimension sera celle correspondant aux différentes parties, et la seconde aux différents tirages pour chaque partie.

Q 21 : Construire, en appelant plusieurs fois la fonction `tirage_uni`, les différentes parties dans le programme principal et afficher temporairement le tableau des gains pour vérifier le fonctionnement. L'utilisateur pourra par exemple choisir `nbpar = 8` et `nbtir = 20`.

II.E Statistiques

Dans toute cette partie, on choisira un nombre de parties assez grand (`nbpar = 1000` au moins).

⋆ **Q 22 :** Écrire une fonction de type `void` en C, ou une `subroutine` en fortran, nommée `print_stat`, qui calcule et affiche, pour chaque tirage jusqu'à `nbtir`, la valeur moyenne et la variance des gains sur toutes les parties. On passera l'ensemble du tableau 2D `gain` à la procédure `print_stat`. Vérifier que le gain moyen est toujours à peu près nul, mais que la variance croît linéairement avec le nombre de tirages (à l'exécution, on pourra choisir pour cette question `nbtir = 20`).

Q 23 : Allouer dans le programme principal un tableau `sortie` contenant `nbpar` éléments, destiné à contenir les temps de sortie des différentes parties. Transformer la fonction `tps_sortie` en une fonction de type `void` en C, ou en une subroutine en fortran, nommée `cal_sortie` pour qu'elle admette maintenant le tableau 2D `gain` en argument, et remplisse le tableau 1D `sortie` qui lui sera également passé en argument. Afficher les différents temps de sortie dans le programme principal.

⋆ **Q 24 :** Écrire une fonction `moyenne` qui renvoie la moyenne d'un tableau 1D. Appeler cette fonction dans le programme principal, avec comme argument le tableau `sortie`, et afficher le temps de sortie moyen pour les `nbpar` parties.

Q 25 : On cherche dans cette question à étudier le comportement du temps de sortie **moyen** en fonction de S. Passer en commentaire dans le programme principal les lignes de code demandant à l'utilisateur de choisir S. Faire varier S de 0 à 8 par pas de 0.5, et afficher pour chaque valeur de S le temps de sortie moyen. Prendre un nombre de tirages d'au moins 1000 pour éviter toute saturation sur `tps_sortie`. Vérifier que celui-ci croît à peu près comme S^2.

En fin d'épreuve,
- **imprimer les fichiers sources créés ;**
- **se connecter sur le serveur d'applications :**
  `ssh sappli1.ccre.upmc.fr`
  et fournir le mot de passe (pas d'écho lors de la saisie)
- **faire de `ex-mni-ses2` votre répertoire de travail**
  `cd ex-mni-ses2`
- **lancer la commande unix interactive**
  `envoi1-mni.sh`
  et répondre aux questions pour envoyer les fichiers sources fortran ou C

III Annexes

III.A Annexe de la partie I

fichier im_auto.txt

107	Peugeot	citadine	FRA	26758	39670
206/207	Peugeot	citadine	FRA	167931	182362
307/308	Peugeot	berline	FRA	92266	83390
500	Fiat	citadine	ITA	21451	25453
C1	Citroen	citadine	FRA	25009	40726
C2	Citroen	citadine	FRA	19233	20268
C3	Citroen	citadine	FRA	60136	105744
C4-Xsara	Citroen	berline	FRA	137026	108811
C5	Citroen	berline	FRA	25697	32209
Clio	Renault	citadine	FRA	152578	146266
Corsa	Opel	citadine	USA	35373	35920
Fiesta	Ford	citadine	USA	38070	50603
Focus	Ford	berline	USA	42106	36934
Golf	Volkswagen	berline	DEU	41228	46746
Ibiza	Seat	citadine	DEU	18385	24067
Kangoo	Renault	espace	FRA	21501	20642
Laguna	Renault	berline	FRA	33011	24377
Logan	Dacia	citadine	FRA	34095	20939
Megane	Renault	berline	FRA	121062	153399
Mini	Mini	citadine	DEU	18999	17765
Modus	Renault	citadine	FRA	35034	37252
Polo	Volkswagen	citadine	DEU	41491	44608
Punto	Fiat	citadine	ITA	22759	23999
Qashqai	Nissan	4x4	JPN	17861	20881
Twingo	Renault	citadine	FRA	65333	107367
Yaris	Toyota	citadine	JPN	30430	29931

III.B Annexe de la partie II : langage C

Dans cette annexe, les prototypes indiqués sont incomplets. En particulier, à part dans le cas du type `void`, les types des arguments ou des fonctions restent à déterminer.

Fonction `tirage_uni` (question 15) :
`tirage_uni(void)`

Fonction `tps_sortie` dans la question 17 :
`tps_sortie(m, tab[m], seuil)`

Fonction `tirage_nor` (question 18) :
`tirage_nor(void)`

Fonction `print_stat` (question 22) :
`void print_stat(n, m, tab[n][m])`

Fonction `cal_sortie` dans la question 23 :
`void cal_sortie(n, m, tab[n][m], seuil, sortie[n])`

Fonction `moyenne` de la question 24
`moyenne(n, tab[n])`

III.C Annexe de la partie II : langage Fortran

Dans cette annexe, les interfaces indiquées sont incomplètes. En particulier, les types des arguments ou des fonctions restent à déterminer Il en est de même pour les vocations (`INTENT`) des arguments.

Fonction `tirage_uni` (question 15) :
`function tirage_uni()`

Fonction `tps_sortie` dans la question 17 :
`function tps_sortie(tab, seuil)`

Fonction `tirage_nor` (question 18) :
`function tirage_nor()`

Sous-programme `print_stat` (question 22) :
`subroutine print_stat(tab)`

Sous-programme `cal_sortie` dans la question 23 :
`subroutine cal_sortie(tab, seuil, sortie)`

Fonction `moyenne` de la question 24 :
`function moyenne(tab)`