TP5 : Probabilités.

Préliminaires.
Téléchargez depuis le site l'archive `TP5.zip` et extrayez les fichiers dans votre dossier personnel puis vérifiez la compilation du fichier `TP5.cpp`.

Le début du fichier contient des fonctions qui seront principalement utilisées dans les tests : `evalProba`, `closeEnought`, et `testProba`. En particulier, ces fonctions utilisent *des pointeurs sur d'autres fonctions* : vous n'avez pas besoin de comprendre leur fonctionnement précis pour faire le TP.

Les fonctions `randint` et `rand01` sont là pour vous aider à manipuler le tirage aléatoire du c++. La fonction `afficheDistribution` est utilisée dans le premier exercice pour visualiser les résultats.

Exercice 1 (Lancé de dés)**.** (1) Observez les deux fonctions `deUniforme` et `de4Pipe` qui tirent des résultats de dés (à 6 faces puis à 4 faces) en utilisant la fonction `randint` qui vous est donnée.

(a) Lancez la fonction `exempleDeUniforme` pour en comprendre l'utilisation.

(b) Observez le code de `de4Pipe`, quelle est la probabilité d'obtenir les différents nombres 1,2,3 et 4 ?

(c) Vérifiez en lançant les exemples que la probabilité évaluée est proche de celle que vous calculez de façon théorique. (Pour évaluer la probabilité, la fonction `evalProba` effectue un grand nombre de tirages et calcul la proportion d'une valeur donnée)

(2) En vous inspirant de l'exemple précédent, implantez la fonction `dePipe` qui tire un dé avec une probabilité $\frac{1}{8}$ de tomber sur chacun des nombres $1, \ldots, 5$ et une probabilité $\frac{3}{8}$ de tomber sur 6. **Attention**, la fonction doit bien retourner un **entier entre 1 et 6** et non pas une probabilité !

(3) On souhaite à présent **observer** et vérifier les propriétés de nos dés. Pour cela, on effectue un grand nombre de lancés et on calcule la **distribution** des résultats : c'est-à-dire le nombre de fois que l'on obtient chacune des valeurs. Compléter la fonction `distriDeUniforme` : la création du tableau vous a été donnée, il reste à effectuer les tirages et à mettre dans chaque case i le nombre de fois que le nombre $i+1$ est tiré. Par exemple, la case 0, contiendra le nombre de 1, la case 1 le nombre de 2, etc.

(4) Selon le même modèle, implantez la fonction `distriDePipe`.

(5) Implantez la fonction `distriSommesDes` qui calcule la distribution de la somme de deux dés uniformes.

Exercice 2 (Lancé de pièces)**.**
Dans cet exercice, on va chercher à simuler l'expérience suivante : quelle est la probabilité d'obtenir exactement k piles après n lancés d'une pièce pipée ?

(1) Implantez la fonction `piece` qui simule un lancé de pièce avec une probabilité donnée d'obtenir pile (vous devez utiliser la fonction `rand01` qui vous est donnée).

(2) Implantez la fonction `nbPiles` qui utilise `piece` pour effectuer n lancés de pièces selon la probabilité p et compte le nombre de fois que l'on obtient pile.

(3) On cherche à présent à évaluer la probabilité d'obtenir exactement k piles sur n lancés. Pour cela on va reproduire l'expérience de la question précédente un grand nombre de fois et calculer la proportion de fois où le nombre de piles est exactement la valeur k. C'est le rôle de la fonction `pnkEval` : les paramètres n, k, et p correspondent à l'expérience et N au nombre de fois que l'on souhaite la répéter. Par exemple, pour évaluer la probabilité d'obtenir 5 fois piles sur 10 lancés ($k = 5$ et $n = 10$) : on peut reproduire 10000 fois les 10 lancés et compter le nombre de fois où l'on a obtenu exactement 5 piles. On divisera ensuite ce nombre par 10000 et on obtient ainsi une évaluation expérimentale de la probabilité.

(4) On cherche à présent à calculer la probabilité exacte pour la comparer à l'évaluation.

(a) Supposez que $A = P_{n-1,k-1}$ la probabilité d'obtenir $k-1$ piles sur n lancés et $B = P_{n-1,k}$ la probabilité d'obtenir k piles avec $n-1$ lancés, quelle est la probabilité $P_{n,k}$ d'obtenir exactement k piles avec n lancés ?

(b) En déduire une formule récursive pour $P_{n,k}$.

(c) Implantez la fonction `pnkRec` et lancez les tests correspondants : on teste d'abord que la fonction renvoie un résultat correct puis on teste la correspondance avec l'évaluation précédente.

(5) Le problème de la fonction `pnkRec` est que sa complexité est 2^n. Il est possible de faire mieux en calculant les probabilités dans un double tableau (méthode dynamique). Implantez la fonction `pnkDyn` dont on vous donne quelques instructions (en particulier, la création du tableau double dimension).

Exercice 3 (Le choix gagnant)**.**
On imagine le jeu suivant : une somme d'argent importante est cachée dans une boite alors que deux autres boites sont laissées vide. La boite gagnante est choisie uniformément parmi les 3. Le joueur choisi une des trois boites. A ce moment-là, le maître du jeu qui a l'ensemble des informations (il sait quelle boite a choisi le joueur et quelle boite est gagnante) ouvre une des boites perdantes restantes puis offre au joueur la possibilité de changer son choix. Que doit faire le joueur ?

Exemple :

La boite gagnante est la boite 2. Le joueur choisit la 1. Le maître du jeu ouvre la boite 3 (il ne peut ouvrir ni la 1 car c'est le choix du joueur, ni la 2 car c'est la boite gagnante). Le joueur conserve son choix et perd.

La boite gagnante est la boite 3. Le joueur choisit la boite 3. Le maître du jeu ouvre la boite 1 (il avait le choix entre la 1 et la 2). Le joueur change d'avis et choisit donc la boite 2. Le joueur perd à nouveau.

(1) Imaginez 2 exemples ou le joueur gagne (l'un où il conserve son choix et un où il change d'avis)

(2) Implantez la fonction `boiteDevoilee` qui correspond à l'action du maître de jeu connaissant la boite gagnante et le choix du joueur. Les boites sont numérotées de 1 à 3. La règle est la suivante :
— Si le joueur n'a pas choisi la boite gagnante, le maître du jeu n'a qu'une seule possibilité (la seule boite qui n'est ni gagnante, ni choisie par le joueur)
— Si le joueur a choisi la boite gagnante alors le maître du jeu dévoile une des deux boites restantes au hasard avec probabilité 0.5.

(3) On a implanté la fonction `joueurMemeAvis` qui simule une partie ou le joueur ne change jamais d'avis (en particulier, on n'a pas besoin de connaître la boite dévoilée par le

maître du jeu). La probabilité est clairement 1/3 ce que confirme l'évaluation. Implantez la fonction `joueurChangeAvis` qui simule le cas ou le joueur change systématiquement d'avis après que le maître du jeu ait dévoilé une boite. Remarquez que le joueur n'a alors qu'un seul choix : la boîte non dévoilée qu'il n'a pas choisie. Evaluez la probabilité obtenue (grâce à la fonction `evalProba`, regardez dans la fonction `main`) et expliquez le résultat.

Exercice 4 (Aller plus loin ♣)**.**

(1) Quel est l'évènement le plus probable ?
— Tirer au moins une face 1 en lançant 6 dés non pipés simultanément ?
— Tirer au moins 2 faces 1 en lançant 12 dés non pipés simultanément ?
— Tirer au moins 3 face 1 en lançant 18 dés non pipés simultanément ?
Evaluez expérimentalement ces probabilités pour trouver la réponse (vous pouvez aussi calculer la formule exacte et vérifier).

(2) On souhaite tirer un nombre uniformément entre 0 et 35 et on possède deux dés à 6 faces non pipés. Implantez une fonction qui simule l'expérience en utilisant uniquement la fonction `deUniforme` et vérifiez par des évaluations que votre distribution est uniforme.

(3) Même question, mais cette fois on cherche un nombre entre 0 et 12.