université PARIS DIDEROT PARIS 7

PARTIEL INTRO À L'INFORMATIQUE & À LA PROGRAMMATION (IF1) 26/10/2013 13^{00}-15^{00}

Aucun document. Aucune machine. Barème indicatif. Exercices tous indépendants.
Une question peut toujours être traitée en utilisant les précédentes (traitées ou non).
Les morceaux de code Java devront être clairement présentés, indentés et commentés.

Exercice 1. (5 points) Expliquer ce que produit l'exécution de chacun des deux programmes suivants :

```
class Obwod{
   public static void main(String[] args){
      int i=2;
      if(i>0){
         int j;
         j = 2+2 *i;
         i = i+2 *j;
         System.out.println(i+" "+j);
         if(j>0){
            int k;
            k = i+j/2;
            j = j+k/2;
            i = k+i/2;
            System.out.println(i+" "+j+" "+k);
         }
         int k;
         k = i+j/2;
         j = j+k/2;
         i = k+i/2;
         System.out.println(i+" "+j+" "+k);
      }
      double j;
      int k;
      i = 2+i/2;
      j = 2+i/2;
      k = 1+i/2;
      System.out.println(i+" "+j+" "+k);
      i = 2+i/2;
      j = i/4.0;
      k = i/4;
      System.out.println(i+" "+j+" "+k);
   }
}
```

```
class Magic{
   static int cad(int m){
      System.out.println("cad prend "+m);
      m = m *2;
      return m-1;
   }
   static int abra(int m){
      int cad = 3;
      System.out.println("abra prend "+m);
      return cad *m+cad(m/3);
   }
   public static void main(String[] args){
      int m = 6;
      m = abra(m)+m;
      System.out.println("resultat "+m);
      m = abra(cad(abra(3)));
      System.out.println("resultat "+m);
    }
}
```

Exercice 2. (3 points) On considère les *fonctions booléennes ternaires*, c'est-à-dire les fonctions qui prennent trois arguments de type booléen et renvoient un résultat de type booléen.

1. Donner le nombre total de telles fonctions. Justifier.
2. On s'intéresse à l'une d'elles, appelée *sheraff*, dont l'évaluation donne *faux* si et seulement si les trois arguments portent la même valeur de vérité. L'implémenter de deux manières différentes, sous la forme de deux fonctions `sheraff1` et `sheraff2` (indépendantes l'une de l'autre) avec ces conditions :
   (1) le corps de `sheraff1` ne doit contenir qu'une seule instruction,
   (2) le corps de `sheraff2` ne doit utiliser aucun opérateur booléen (pas même l'opérateur ==).

Exercice 3. (4 points) Voici une adaptation d'une question du dernier *cahier d'évaluation de CM2*.

On souhaite expédier `n` colis en combinant éventuellement deux modes de transport :
– par bateau, en mettant les colis dans des conteneurs chacun pouvant contenir jusqu'à cinquante colis ;
– par la route, en mettant les colis dans des camions.
Les tarifs sont de `pBat` euros par conteneur (bateau) et de `pCam` euros par colis (camion).

Aucune boucle ne peut être utilisée dans cet exercice.

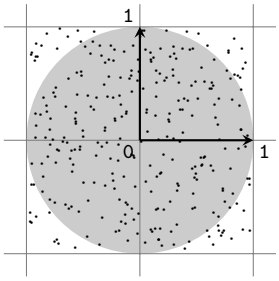
1a. Expliquer quelle stratégie permet de minimiser le coût du transport.

1b. Écrire une fonction `cout` qui prend en arguments un nombre (entier) de colis `n` et deux prix `pBat` et `pCam` (réels en euros) et renvoie le coût minimum du transport de ces `n` colis. Par exemple :

`cout(330, 349.99, 6.99)` renvoie `2306.70`
(correspondant à 330 colis par la route),
`cout(330, 349.99, 8.99)` renvoie `2369.64`
(correspondant à 6 conteneurs par bateau et 30 colis par la route),
`cout(340, 349.99, 8.99)` renvoie `2449.93`
(correspondant à 7 conteneurs par bateau, dont un rempli à 80%).

2. Proposer une nouvelle fonction `cout` qui permettrait de prendre en considération la répercussion à partir du `01/01/2014` de l'écotaxe de `0,00111` euro par colis et par kilomètre parcouru sur route.

Exercice 4. (4 points) Il s'agit d'utiliser le hasard pour obtenir une approximation de π.



1. En utilisant `Math.random()` qui renvoie un réel pseudo-aléatoire compris entre `0.0` (largement) et `1.0` (strictement), écrire une fonction `aleaPlusMoinsUn` qui renvoie un réel pseudo-aléatoire compris entre `-1.0` (largement) et `1.0` (strictement).

   La classe `Math` *ne doit plus être utilisée dans la suite.*

2. Écrire une fonction `dansDisqueUnite` qui prend en arguments deux réels `abs` et `ord` et teste si le point de coordonnées (`abs`, `ord`) appartient au disque unité.
3. Écrire une fonction `nbDansDisqueUnite` qui prend en argument un entier `n`, tire aléatoirement `n` points d'abscisse et d'ordonnée entre `-1.0` et `1.0` et renvoie le nombre de ces points appartenant au disque unité.
4. En déduire une fonction `pi` qui prend en argument un entier `n` et renvoie une approximation de π.

Exercice 5. (5 points) Dans le *jeu du 13*, deux joueurs retirent à tour de rôle 1, 2 ou 3 allumette(s) d'un tas qui contient 13 allumettes au début du jeu. Le joueur qui retire la dernière allumette gagne.

1. Écrire la classe `Jeu13` qui implémente ce jeu conformément à l'exemple d'exécution suivant :

```
$ java Jeu13
Au tour du joueur 0. Il reste 13 allumette(s). Combien en retirez-vous? 3
Au tour du joueur 1. Il reste 10 allumette(s). Combien en retirez-vous? 2
Au tour du joueur 0. Il reste 8 allumette(s). Combien en retirez-vous? 3
Au tour du joueur 1. Il reste 5 allumette(s). Combien en retirez-vous? 1
Au tour du joueur 0. Il reste 4 allumette(s). Combien en retirez-vous? 3
Au tour du joueur 1. Il reste 1 allumette(s). Combien en retirez-vous? 1
Le joueur 0 a perdu.
$
```

On admettra que les deux joueurs jouent de manière légale, c'est-à-dire que l'entier qu'ils rentrent à chaque tour est compris entre 1 et 3 et inférieur ou égal au nombre d'allumettes restantes.

2. Proposer une stratégie gagnante pour le joueur `0`.