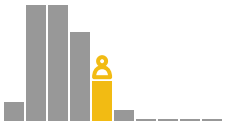
Afin d'éviter la fuite de questions, ce rapport ne doit pas être partagé avec les candidats.

YORO, Ange Carmel (ange-carmel.yoro@atos.net)

Campagne : Java boosté Algorithmes - Senior Langage(s) de programmation : Java Langage : Français Date : 08/02/2020

SCORE	RANG	DURÉE	MEILLEUR QUE
48%	1	5H04	96%
6 588 / 13 700 pts	/ 5	/ 13H14	des développeurs



Java

48% (6 588 / 13 700)

Connaissance du langage	57% (857 / 1 509)
Fiabilité	50% (953 / 1 900)
Modélisation	29% (160 / 557)
Résolution de problèmes	47% (4 618 / 9 734)

Attention

- Problème de lisibilité de code détecté

Question 1: Magic Stones

Java 30:00 / 30:00 0 / 600 pts

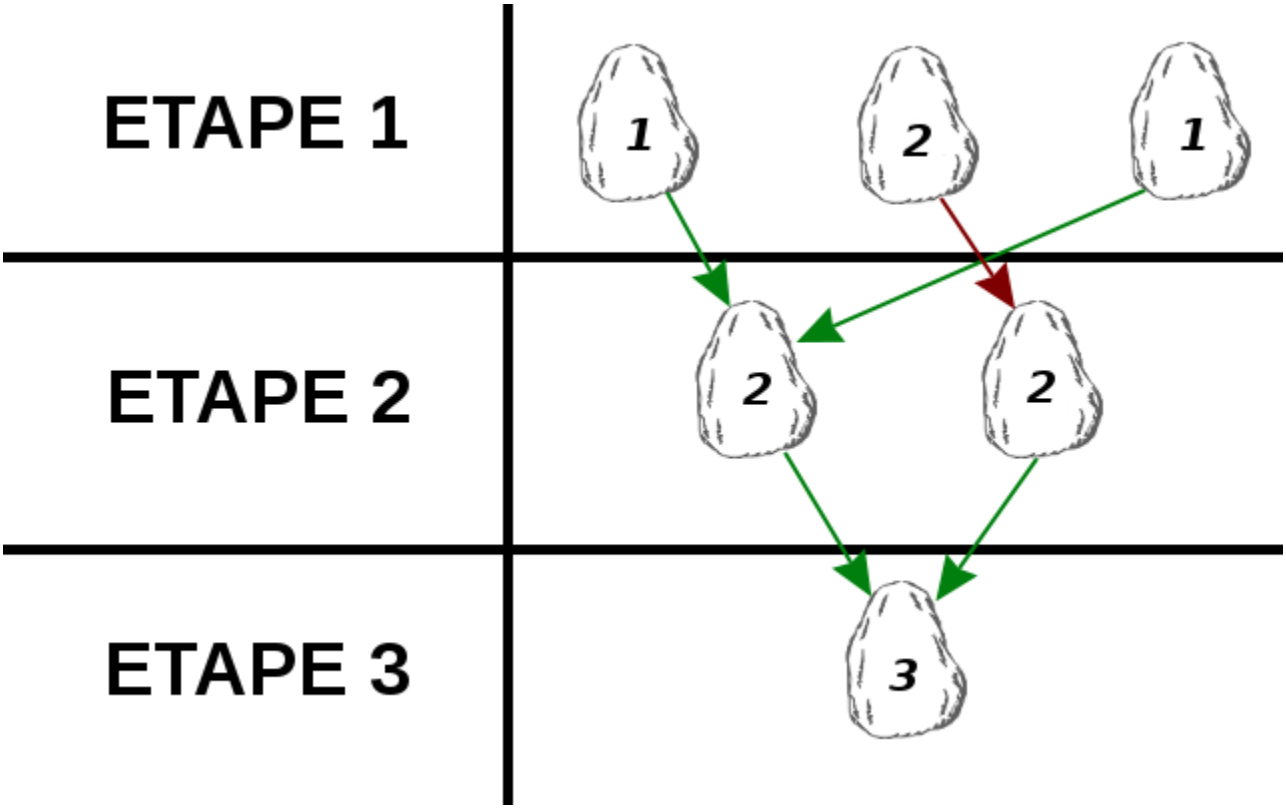
Le temps alloué à cette question s'est écoulé. La réponse du candidat a été automatiquement récupérée à la fin du décompte.

Question

Vous transportez un certain nombre de pierres ayant chacune un niveau ***n*** (entier, ***0 < n < 10000***).

Deux pierres d'un même niveau n peuvent être fusionnées pour créer une pierre de niveau n + 1.

La méthode ***magic*** retourne le nombre minimum de pierres pouvant être atteint par fusions successives.



Dans cette exemple, le nombre minimum de pierres est 1.

Écrivez le corps de la méthode ***magic(stones)***. ***stones*** est une liste d'entiers qui contient au moins un élément.

Réponse

```java
// Java code below
import java.util.*;
import java.io.*;
import java.nio.*;
import java.math.*;

class Solution {
    public static int magic(List<Integer> stones) {
        List<Integer> table = new ArrayList<Integer>();
        Collections.sort(stones);
        int i = 0;
        boolean arret = true;
        while (arret) {
            arret = false;
            while (i < stones.size()-1) {
                if(stones.get(i) == stones.get(i+1)){
                    table.add(stones.get(i)+1);
                    i +=2;
                    arret = true;
                }else{
                    table.add(stones.get(i));
                    i +=1;
                }
            }
            if(table.size()>0){
                stones = table;
                table.clear();
            }

        }
        return stones.size();

    }
}
```

Résultat

Tests basiques
Résolution de problèmes ~~+450pts~~

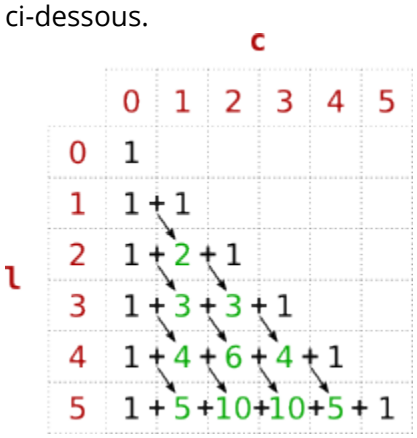
Tests complexes
Fiabilité ~~+150pts~~

Question 2: Pyramide

Java 26:16 / 30:00 0 / 600 pts

Question

Implémentez ***get(int l, int c)*** en découvrant un modèle de construction générique à partir de l'illustration ci-dessous.



Données : ***0 ≤ c ≤ l ≤ 10000***
Important :
Essayez de privilégier un faible usage de la mémoire RAM, 64 bits peuvent ne pas suffire pour stocker certains résultats ! Privilégiez ensuite le temps d'exécution si c'est possible.

Exemples :
get(4, 2) devrait retourner la chaîne ***6***
get(5, 0) devrait retourner la chaîne ***1***
get(67, 34) devrait retourner la chaîne ***14226520737620288370***

Réponse

```java
import java.math.*;

class A {

    /**
     * @return the number in the (l, c) cell
     */
  static String get(int l, int c) {
    return String.valueOf(getFactoriel(l)/(getFactoriel(c)*getFactoriel(l-c)));
  }

    static Integer getFactoriel(int nobre){
        if(nobre == 0) return 1;
        else{
            return nobre *getFactoriel(nobre-1);
        }
    }
}
```

Résultat

Le résultat est correct avec un jeu de données simple
Résolution de problèmes ~~+395pts~~

NA L'algorithme retourne des valeurs justes (et dans les temps) quand 64 bits ne suffisent plus
Résolution de problèmes ~~+58pts~~

NA get(0, 0) retourne 1
Fiabilité ~~+29pts~~

NA L'utilisation de la mémoire n'augmente pas (ou peu) quand c augmente
Résolution de problèmes ~~+59pts~~

NA L'algorithme retourne toujours un résultat correct dans les temps ($0 \leq c \leq l \leq 10000$)
Résolution de problèmes ~~+59pts~~

Question 3: Du désordre le plus grand gagne

Java 01:48 / 05:00 100 / 100 pts

Question

Implémentez la méthode ***Algorithm.findLargest(int[] numbers)*** afin qu'elle retourne le plus grand nombre dans ***numbers***.

Note : Le tableau contient toujours au moins un nombre.

Réponse

```java
import java.util.Arrays;

class Algorithm {

    /** @return the largest number of the given array */
  static int findLargest(int[] numbers) {
    // Your code goes here
    return Arrays.stream(numbers).max().getAsInt();
  }

}
```

Résultat

- Fonctionne dans des cas simples
  Résolution de problèmes +32pts
  - Fonctionne quand le tableau contient seulement Integer.MIN_VALUE
    Fiabilité +58pts
  - Fonctionne quand le plus grand élément est à la position 0
    Fiabilité +5pts
  - Fonctionne quand le plus grand élément est à la fin du tableau
    Fiabilité +5pts

Problème de lisibilité de code détecté

The utility class name 'Algorithm' doesn't match '[A-Z][a-zA-Z0-9]+(Utils?|Helper)'

Question 4: Utilisation d'un buffer/join à la place de '+'

Java 02:30 / 05:00 83 / 100 pts

Question

StringUtils.concat(String[] strings) sert à joindre des chaînes de caractères bout à bout.

Par exemple, à partir d'un tableau contenant ***"f", "o", "o", "bar"***, ***concat*** devrait retourner ***"foobar"***.

Données : ***strings*** contient toujours au moins un élément.

Implémentez ***StringUtils.concat(String[] strings)***.

Réponse

```java
class StringUtils {

    /**
     * Concatenates the given array of strings.
     */
  static String concat(String[] strings) {
    StringBuffer bf = new StringBuffer();
        for(String b : strings){
            bf.append(b);
        }

        return bf.toString();
  }

}
```

Résultat

Les chaînes sont concaténées
Connaissance du langage +25pts

Utilisation d'un buffer ou join() plutôt que '+'
Connaissance du langage +58pts

Utilisation de StringBuilder ou String.join() plutôt que StringBuffer
Connaissance du langage ~~+17pts~~

Question 5: Somme de facteurs

Cross 04:55 / 10:00 200 / 200 pts

Question

La méthode ***computeMultiplesSum(n)*** doit renvoyer la somme de tous les multiples positifs de 3 ou 5 ou 7 strictement inférieurs à ***n***.

Par exemple, pour ***n=11***, on obtient 3,5,6,7,9,10 en tant que multiples et la somme de ces multiples vaut 40.

Implémentez ***computeMultiplesSum(n)***.

Contraintes:
0 ≤ n < 1000

Réponse

```java
import java.util.*;
import java.io.*;
import java.math.*;

class Solution {

    public static int computeMultiplesSum(int n) {
        // Write your code here
        // To debug: System.err.println("Debug messages...");
        int somme = 0;
        while(n >= 3){
            n--;
            boolean a = n%3 == 0;
            boolean b = n%5 == 0;
            boolean c = n%7 == 0;
            if(a || b || c) {
                somme +=n;
            }
        }
        return somme;
    }

    /* Ignore and do not change the code below */
    // #region main
    public static void main(String args[]) {
        Scanner in = new Scanner(System.in);
        int n = in.nextInt();
        PrintStream outStream = System.out;
        System.setOut(System.err);
        int res = computeMultiplesSum(n);
        System.setOut(outStream);
        System.out.println(res);
    }
    // #endregion
}
```

Résultat

n=20
Résolution de problèmes +40pts

n=125
Résolution de problèmes +40pts

n=202
Résolution de problèmes +40pts

n=996
Résolution de problèmes +40pts

n=0
Fiabilité +40pts

Question 6: Suppression de doublons

Cross 03:05 / 12:00 100 / 100 pts

Question

Implémenter la méthode ***filterDuplicates(data)*** qui prend un tableau ***data*** en entrée et retourne un tableau contenant les valeurs de ***data*** sans les doublons de valeurs.

L'ordre initial des valeurs doit être conservé.

Exemple: ***[7 3 6 4 3 3 4 9]*** => ***[7 3 6 4 9]***

Constraintes:
data n'est jamais ***null***

Réponse

```java
import java.util.*;
import java.io.*;
import java.math.*;

class Solution {

    public static int[] filterDuplicates(int[] data) {
        // Write your code here
        // To debug: System.err.println("Debug messages...");

        return Arrays.stream(data).distinct().toArray();
    }

    /* Ignore and do not change the code below */
    // #region main
    public static void main(String args[]) {
        Scanner in = new Scanner(System.in);
        int n = in.nextInt();
        int[] data = new int[n];
        for (int i = 0; i < n; i++) {
            data[i] = in.nextInt();
        }
        PrintStream outStream = System.out;
        System.setOut(System.err);
        int[] filtered = filterDuplicates(data);
        System.setOut(outStream);
        for (int i = 0; i < filtered.length; i++) {
            System.out.println(filtered[i]);
        }
    }
    // #endregion
}
```

Résultat

Tableau simple
Résolution de problèmes +25pts

Grand tableau
Résolution de problèmes +25pts

Valeur unique
Fiabilité +15pts

Valeurs négatives
Résolution de problèmes +20pts

Tableau vide
Fiabilité +15pts

Question 7: Intervalles

Cross 10:54 / 12:00 200 / 200 pts

Question

Implémentez la méthode ***findSmallestInterval(numbers)*** qui retourne le plus petit intervalle positif entre deux éléments du tableau ***numbers***.

Par exemple, si on considère le tableau ***[1 6 4 8 2]*** , le plus petit intervalle est ***1*** (différence entre ***2*** et ***1***)

Contraintes:
numbers contient au moins deux éléments et au maximum 100 000 éléments. La solution qui privilégie la vitesse d'exécution pour les tableaux de grande taille obtiendra le plus de points. La différence entre deux éléments ne dépassera jamais la capacité d'un entier pour votre langage.

Réponse

```java
import java.util.*;
import java.io.*;
import java.math.*;

class Solution {

    public static int findSmallestInterval(int[] numbers) {
        // Write your code here
        // To debug: System.err.println("Debug messages...");
        Arrays.sort(numbers);
        int max = numbers[1] - numbers[0];
        for(int i = 0; i < numbers.length-1; i++){
            int diff = numbers[i+1] - numbers[i];
            if(diff < max) max = diff;
        }
        return max;
    }

    /* Ignore and do not change the code below */
    // #region main
    public static void main(String args[]) {
        Scanner in = new Scanner(System.in);
        int n = in.nextInt();
        int[] numbers = new int[n];
        for (int i = 0; i < n; i++) {
            numbers[i] = in.nextInt();
        }
        PrintStream outStream = System.out;
        System.setOut(System.err);
        int res = findSmallestInterval(numbers);
        System.setOut(outStream);
        System.out.println(res);
    }
    // #endregion
}
```

Résultat

Validateur simple
Résolution de problèmes +40pts

Couple inversé
Résolution de problèmes +40pts

Nombres négatifs
Résolution de problèmes +40pts

Très grand tableau
Résolution de problèmes +40pts

Grande différence
Fiabilité +40pts

Question 8: Soldes d'été

Cross 14:04 / 15:00 235 / 300 pts

Question

C'est bientôt les soldes d'été !

Vous travaillez pour un magasin qui souhaite offrir une réduction de ***discount%*** sur le produit le plus cher acheté par un client donné pendant la période des soldes.

Le responsable du magasin vous demande de développer la méthode ***calculateTotalPrice()***.

Cette méthode :
prend en paramètres la liste de prix des produits achetés par le client et le pourcentage de réduction ***discount***. retourne le prix de vente total (arrondi à l'entier inférieur si le total ne tombe pas rond).

Contraintes:
$0 \leq$ ***discount*** ≤ 100 $0 <$ prix d'un produit < 100000 $0 <$ nombre de produits < 100

Réponse

```java
import java.util.*;
import java.io.*;
import java.math.*;

class Solution {

    public static int calculateTotalPrice(int[] prices, int discount) {
        // Write your code here
        // To debug: System.err.println("Debug messages...");
        Arrays.sort(prices);
        prices[prices.length-1] -= (prices[prices.length-1]*discount)/100;
        double somme = 0;
        for(int price : prices){
            somme+=price;
        }

        return (int)somme;
    }

    /* Ignore and do not change the code below */
    // #region main
    public static void main(String args[]) {
        Scanner in = new Scanner(System.in);
        int discount = in.nextInt();
        int n = in.nextInt();
        int[] prices = new int[n];
        for (int i = 0; i < n; i++) {
            prices[i] = in.nextInt();
        }
        PrintStream outStream = System.out;
        System.setOut(System.err);
        int price = calculateTotalPrice(prices, discount);
        System.setOut(outStream);
        System.out.println(price);
    }
    // #endregion
}
```

Résultat

Simple somme
Résolution de problèmes +35pts

Bonne vente
Résolution de problèmes +35pts

Gros discount
Résolution de problèmes +35pts

Arrondi correct
Fiabilité ~~+35pts~~

Un produit gratuit
Résolution de problèmes +35pts

Fin de solde
Résolution de problèmes +35pts

Grosse vente
Résolution de problèmes ~~+30pts~~

Même prix
Fiabilité +30pts

Un seul produit
Fiabilité +30pts

Question 9: Correction simple

Java 01:27 / 02:00 100 / 100 pts

Question

La méthode ***sumRange*** devrait retourner la somme des entiers compris entre 10 et 100 inclusifs contenus dans le tableau passé en paramètre.

Corrigez la méthode ***sumRange***.

Note : le paramètre ints n'est jamais null.

Réponse

```java
// Java code below
import java.util.*;
import java.io.*;
import java.nio.*;
import java.math.*;

class Solution {

  static int sumRange(int[] ints) {
      int sum = 0;
      for (int i = 0; i < ints.length; i++) {
          int n = ints[i];
          if (n >= 10 && n <= 100) sum += n;
      }
      return sum;
  }
}
```

Résultat

La solution fonctionne avec { 1, 4, 9, 12, 98, -10, 10 }
Résolution de problèmes +50pts

La solution fonctionne avec { 30, 4, 9, 12, 98, -10, 10 }
Fiabilité +50pts

Problème de lisibilité de code détecté

The utility class name 'Solution' doesn't match '[A-Z][a-zA-Z0-9]+(Utils?|Helper)'

Question 10: Expression booléenne simple

Java 01:47 / 02:00 100 / 100 pts

Question

A.a(int i, int j) devrait retourner ***true*** si un des arguments est égal à 1 ou si leur somme est égale à 1.

Par exemple :
A.a(1, 5) retourne ***true***
A.a(2, 3) retourne ***false***
A.a(-3, 4) retourne ***true***

Réponse

```java
// Java code below
class A {

  static boolean a(int i, int j) {
    return (i == 1 || j == 1 || i+j == 1)?true:false;
  }

}
```

Résultat

Retourne true si i ou j est égal à 1, sinon false
Connaissance du langage +67pts

Retourne true si i+j est égal à 1
Fiabilité +33pts

Problème de lisibilité de code détecté

The utility class name 'A' doesn't match '[A-Z][a-zA-Z0-9]+(Utils?|Helper)'

Question 11: Synchronisation d'un compteur

Java 02:00 / 02:00 0 / 100 pts

Le temps alloué à cette question s'est écoulé. La réponse du candidat a été automatiquement récupérée à la fin du décompte.

Question

Transformez la méthode ***Counter.increment()*** pour qu'elle supporte l'accès concurrentiel de plusieurs threads.

Réponse

```
1  class Counter {
2
3    private static int count = 0;
4
5      /**
6       * Increments count in a thread-safe manner.
7       */
8    public static int increment() {
9      this.count = this.count + 1;
10     return count;
11   }
12
13 }
```

Résultat

Accès concurrentiel synchronisé
Connaissance du langage ~~+100pts~~

Question 12: Exception

Java 03:45 / 05:00 147 / 200 pts

Question

Mettez à jour le code en appliquant les règles suivantes : Si une exception est levée par ***s.execute()*** alors appeler ***c.rollback()*** et propager l'exception, sinon appeler ***c.commit()*** Dans tous les cas, ***c.close()*** doit être appelée avant de quitter la méthode ***a(Service s, Connection c)***

Réponse

```
class A {

    /**
     * Executes the service with the given connection.
     */
    void a(Service s, Connection c) {
        // update this code
    try{
      s.setConnection(c);
      s.execute();
      c.commit();
    }catch(Exception e){
      c.rollback();
    }finally{
      c.close();
    }
  }

}

interface Service {
  void execute() throws Exception;
  void setConnection(Connection c);
}

interface Connection {
  void commit();
  void rollback();
  void close();
}
```

Résultat

Fonctionne correctement quand il n'y a pas d'exception
Connaissance du langage +56pts

Utilisation de finally
Connaissance du langage +65pts

Rollback et close quand il y a une exception
Connaissance du langage +26pts

L'exception est propagée
Connaissance du langage ~~+53pts~~

Question 13: Hello World

Java | 05:00 / 05:00 | 0 / 200 pts

Le temps alloué à cette question s'est écoulé. La réponse du candidat a été automatiquement récupérée à la fin du décompte.

Question

console> ***java Echo Hello you !***
Hello
you
!
console>

Écrivez le programme ***Echo***.

Réponse

```java
// Java code below
class Echo {
  private static void execute(String... args){
        System.out.println(String.join("\n",args));
    }
}
```

Résultat

Le résultat affiché dans la console correspond
Connaissance du langage ~~+182pts~~

Le résultat affiché prend en compte les arguments de la ligne de commande
Fiabilité ~~+18pts~~

Question 14: Range Sum

Java 03:19 / 05:00 200 / 200 pts

Question

Écrivez le corps de la méthode ***calc(array, n1, n2)***.

array est un tableau d'entiers.

Les paramètres ***n1*** et ***n2*** sont des entiers définis par la relation ***0 <= n1 <= n2 < array.length***.

La méthode ***calc*** doit retourner la somme des entiers de ***array*** dont l'index appartient à l'intervalle ***[n1; n2]***.

Réponse

```
1  class Solution {
2      /**
3       * @return the sum of integers whose index is between n1 and n2
4       */
5      public static int calc(int[] array, int n1, int n2) {
6          int somme = 0;
7          for(int i = n1; i <= n2; i++){
8              somme+=array[i];
9          }
10         return somme;
11     }
12 }
```

Résultat

Tests basiques
Résolution de problèmes +100pts

Tests avancés
Résolution de problèmes +100pts

Problème de lisibilité de code détecté

The utility class name 'Solution' doesn't match '[A-Z][a-zA-Z0-9]+(Utils?|Helper)'

Question 15: Abstraction

Java 05:29 / 08:00 200 / 200 pts

Question

Essayez d'améliorer le code affiché dans l'éditeur de réponses tout en conservant le comportement actuel du programme.

Réponse

```
abstract class Animal {
    String name;
     String getName() {
        return name;
    }
 }

class Dog extends Animal {
    /**
     * Creates a new dog with the given name.
     */
    Dog(String name) {
        this.name = name;
    }

    String getName() {
        return name;
    }
}

class Cat extends Animal {
    /**
     * Creates a new cat with the given name.
     */
    Cat(String name) {
        this.name = name;
    }

    String getName() {
        return name;
    }
}

class Application {

    /**
     * @return the name of the given animal
     */
    static String getAnimalName(Animal a) {
        String name = null;
        if (a instanceof Dog) {
            name = a.getName();
        } else if (a instanceof Cat) {
            name = a.getName();
        }

        return name;
    }
}
```

Résultat

Le comportement est conservé et Animal.getName est utilisée
Modélisation +160pts

Application.getAnimalName utilise désormais Animal.getName
Fiabilité +40pts

Problème de lisibilité de code détecté

The utility class name 'Application' doesn't match '[A-Z][a-zA-Z0-9]+(Utils?|Helper)'

Question 16: Jumeaux

Java 11:13 / 12:00 0 / 200 pts

Question

Le jumeau (twin) d'un mot est un mot écrit avec exactement les mêmes lettres (indépendamment de la casse), mais pas nécessairement dans le même ordre.

Par exemple ***Marion*** est le jumeau de ***Romain***.

La méthode ***isTwin(a, b)*** retourne ***true*** si ***b*** est le jumeau de ***a*** ou ***false*** si ce n'est pas le cas. ***a*** et ***b*** sont deux chaînes de caractères non null.

Écrivez le corps de la méthode *isTwin(a, b)* .

Réponse

```java
// Java code below
import java.util.*;
import java.io.*;
import java.nio.*;
import java.math.*;

class Solution {
  public static boolean isTwin(String a, String b) {
    if(a.length() == b.length()){
      String aLower = a.toLowerCase();
      String bLower = b.toLowerCase();
      char[] co = String.valueOf(aLower).toCharArray();
      char[] bo = String.valueOf(bLower).toCharArray();
      for(int i = 0; i < co.length; i++){
        if(co[i] != bo[i]){
          return false;
        }
      }
      return true;
    }
        return false;
  }
}
```

Résultat

Tests simples
Résolution de problèmes ~~+70pts~~

Insensible à la casse
Résolution de problèmes ~~+45pts~~

Chaines courtes ou vides
Fiabilité ~~+30pts~~

Même lettres mais pas jumeaux
Fiabilité ~~+55pts~~

Question 17: Combinaisons possibles dans un tournoi

Java 01:42 / 15:00 300 / 300 pts

Question

Vous organisez un tournoi d'échecs dans lequel les joueurs s'affronteront en duel.

Pour former les duels on procède ainsi : d'abord on tire au hasard un joueur, ensuite on tire au hasard son opposant parmi les joueurs restants. Cette paire forme un des duels du tournoi. On procède de la même manière pour former toutes les paires.

Dans cet exercice, on souhaiterait connaître **le nombre de paires** qu'il est possible d'obtenir sachant que l'ordre des opposants dans une paire n'a pas d'importance.

Par exemple, avec 4 joueurs nommés A, B, C et D, il est possible d'obtenir 6 paires différentes : AB, AC, AD, BC, BD, CD.

Implémentez ***count*** pour retourner le nombre de paires possibles. Le paramètre ***n*** correspond au nombre de joueurs.

Essayez d'optimiser votre solution pour que, dans l'idéal, la durée de traitement soit la même quel que soit ***n***.

Données : ***2 <= n <= 10000***

Réponse

```java
import java.util.*;
import java.io.*;
import java.nio.*;
import java.math.*;

class Solution {

  /** Counts the number of pairs for n players. */
    static int count(int n) {
        return ((n*n)-(n*1))/2;
    }
}
```

Résultat

La solution fonctionne avec un jeu de données simple
Résolution de problèmes +128pts

La solution n'utilise pas excessivement la mémoire (pas de cache)
Résolution de problèmes +43pts

La solution retourne le bon résultat en un temps constant
Résolution de problèmes +86pts

La solution fonctionne avec des données proches de 10000
Résolution de problèmes +43pts

Problème de lisibilité de code détecté

The utility class name 'Solution' doesn't match '[A-Z][a-zA-Z0-9]+(Utils?|Helper)'

Question 18: Tendre vers zéro

Java 08:49 / 15:00 300 / 300 pts

Question

Implémentez la méthode ***closestToZero*** pour renvoyer l'entier du tableau ***ints*** le plus proche de zéro. S'il y a deux entiers tout aussi proches de zéro, considérez l'entier positif comme étant le plus proche de zéro (par exemple si ***ints*** contient -5 et 5, retournez 5). Si ***ints*** est ***null*** ou vide, retournez 0 (zero).

Données : les entiers dans ***ints*** ont des valeurs allant de -2147483647 à 2147483647.

Réponse

```java
import java.util.Arrays;

class A {

  /** @return the number that is closest to zero */
    static int closestToZero(int[] ints) {
        if(ints == null || ints.length == 0) return 0;
    int T;
    int min = ints[0];
    Arrays.sort(ints);
    for(int i =0; i < ints.length; i++){
      T = ints[i];
      if(Math.abs(T)< Math.abs(min) || (T == -min && T > 0)){
        min = T;
      }
    }
    return min;
  }
}
```

Résultat

- Les résultats sont corrects avec un jeu de données simple [7, 5, 9, 1, 4]
  Résolution de problèmes +120pts
  - La solution fonctionne avec -2147483647 ou 2147483647
    Fiabilité +20pts
  - La solution fonctionne quand le tableau ne contient que des entiers négatifs
    Fiabilité +20pts
  - Quand 2 entiers sont aussi proches de 0, alors le positif l'emporte
    Fiabilité +20pts
  - La solution fonctionne quand le tableau ne contient que deux entiers négatifs égaux
    Fiabilité +20pts
  - La solution utilise java.lang.Math.abs()
    Connaissance du langage +60pts
  - La solution fonctionne avec un tableau vide
    Fiabilité +20pts
  - La solution fonctionne avec un tableau null
    Fiabilité +20pts

Problème de lisibilité de code détecté

- The utility class name 'A' doesn't match '[A-Z][a-zA-Z0-9]+(Utils?|Helper)'
- Variables should start with a lowercase character, 'T' starts with uppercase character.

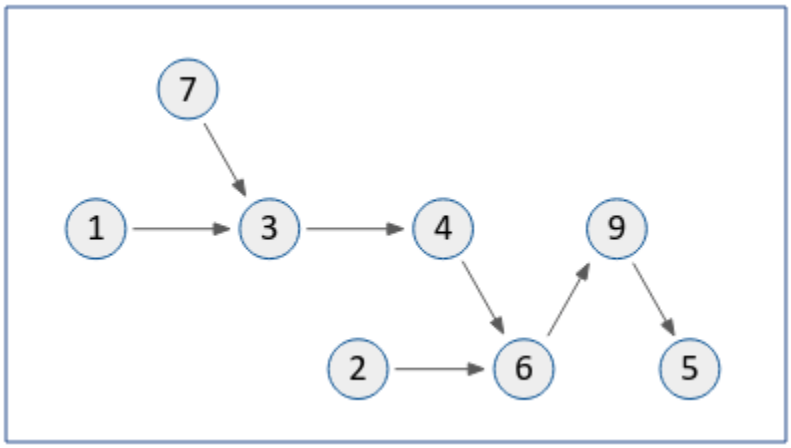
Question 19: Détection de boucles

Cross 20:26 / 25:00 355 / 450 pts

Question

L'objectif de cet exercice est de trouver le noeud terminal d'un réseau simple.

Dans ce réseau simple, chaque noeud est lié à au plus un noeud sortant de manière unidirectionnelle.



Un exemple de réseau simple

Implémentez la méthode ***findNetworkEndpoint(startNodeId, fromIds, toIds)*** qui retourne l'id du dernier noeud du réseau trouvé en partant du noeud ***startNodeId*** et en suivant les liens du réseau.

Dans l'exemple ci-dessus, le noeud terminal en partant du noeud n°2 (ou de n'importe quel noeud) est le noeud n°5.

fromIds et ***toIds*** sont deux tableaux de la même longueur qui décrivent les liens unidirectionnels du réseau (***fromIds[i]*** est lié à ***toIds[i]***).

Dans le cas où vous rencontrez une boucle lors du parcours du réseau, la méthode doit renvoyer l'id du dernier noeud traversé avant de clore la boucle.

Constraintes:
0 < nombre de liens < 10000 Un noeud ne peut pas être directement lié à lui-même.

Réponse

```java
import java.util.*;
import java.io.*;
import java.math.*;

class Solution {

    public static int findNetworkEndpoint(int startNodeId, int[] fromIds, int[] toIds) {
        // Write your code here
        // To debug: System.err.println("Debug messages...");

        List<Integer> listFromId = new ArrayList<Integer>();
        List<Integer> listToId = new ArrayList<Integer>();

        for(int fro : fromIds){
            listFromId.add(fro);
        }


        for(int fro : toIds){
            listToId.add(fro);
        }

        for(Integer elementToId : listToId){
            if(listFromId.indexOf(elementToId)==-1) return elementToId;
        }

        int indexStarTnode = listToId.indexOf(startNodeId);

        return listFromId.get(indexStarTnode);
    }

    /* Ignore and do not change the code below */
    // #region main
    public static void main(String args[]) {
        Scanner in = new Scanner(System.in);
        int startNodeId = in.nextInt();
        int n = in.nextInt();
        int[] fromIds = new int[n];
        for (int i = 0; i < n; i++) {
            fromIds[i] = in.nextInt();
        }
        int[] toIds = new int[n];
        for (int i = 0; i < n; i++) {
            toIds[i] = in.nextInt();
        }
        PrintStream outStream = System.out;
        System.setOut(System.err);
        int endPointId = findNetworkEndpoint(startNodeId, fromIds, toIds);
        System.setOut(outStream);
        System.out.println(endPointId);
    }
    // #endregion
}
```

Résultat

Réseau simple
Résolution de problèmes +85pts

Grand réseau linéaire
Résolution de problèmes +70pts

Simple boucle
Résolution de problèmes ~~+70pts~~

Réseau en arbre
Résolution de problèmes +70pts

Cercle
Résolution de problèmes +65pts

Réseaux disjoints
Résolution de problèmes +65pts

Noeud isolé
Fiabilité ~~+25pts~~

Question 20: Point de jointure

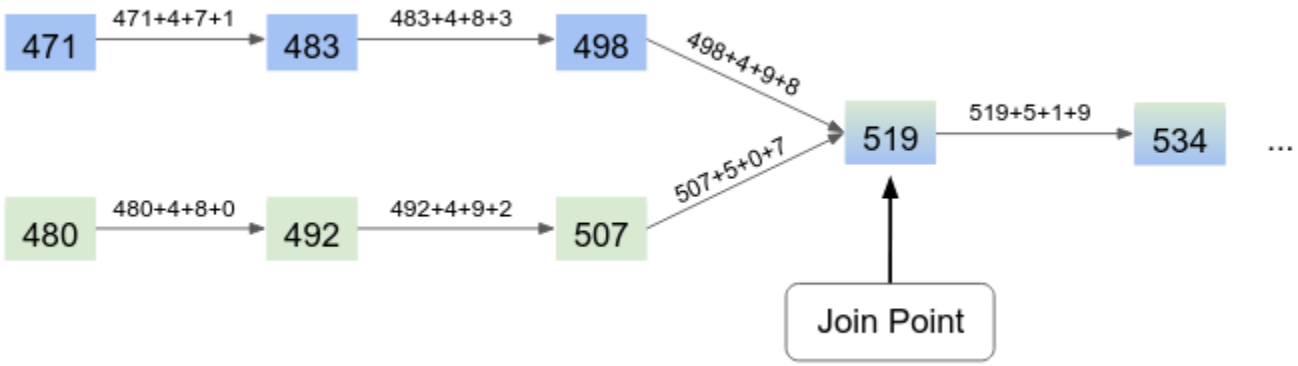
Cross 20:39 / 30:00 530 / 600 pts

Question

On considére la suite de nombres dans laquelle un nombre est suivi du même nombre plus la somme de ces chiffres.

Par exemple ***34*** est suivi de ***41*** (41 = 34 + (3 + 4)). ***41*** est lui-même suivi de ***46*** (46 = 41 + (4 + 1)).

Deux séquences avec des points de départ différents peuvent finir par se rejoindre. Par exemple la séquence qui commence à ***471*** et la séquence qui commence à ***480*** partagent le nombre ***519*** (le point de jointure). Bien évidemment après le point de jointure, les séquences sont identiques.



Un example de deux séquences se rejoignant en 519.

Implémentez la méthode ***int computeJoinPoint(int s1, int s2)*** qui prend les points de départ de deux séquences et renvoie le point de jointure de ces deux séquences.

Contraintes:
Les deux séquences se rejoignent toujours 0 < ***s1, s2*** < 20000000 0 < joint point < 20000000

Réponse

```java
import java.util.*;
import java.io.*;
import java.math.*;

class Solution {
    private static int computeJoinPoint(int s1, int s2){
        List<Integer> listJointPoint = new ArrayList<Integer>();
        while(true){
            if(listJointPoint.indexOf(s1) == -1){
                listJointPoint.add(s1);
                s1 = calculateNextNumber(s1);
            }else
                return s1;

            if(listJointPoint.indexOf(s2) == -1){
                listJointPoint.add(s2);
                s2 = calculateNextNumber(s2);
            }else
                return s2;
        }
    }

    private static int calculateNextNumber(int s){
        char chaineNumber[] = String.valueOf(s).toCharArray();
        Integer nextNumber = s;
        for (char c : chaineNumber) {
            nextNumber+=Integer.parseInt(String.valueOf(c));
        }
        return nextNumber;
    }

    /* Ignore and do not change the code below */
    // #region main
    public static void main(String args[]) {
        Scanner in = new Scanner(System.in);
        int s1 = in.nextInt();
        int s2 = in.nextInt();
        PrintStream outStream = System.out;
        System.setOut(System.err);
        int res = computeJoinPoint(s1, s2);
        System.setOut(outStream);
        System.out.println(res);
    }
    // #endregion
}
```

Résultat

- s1 < s2
  Résolution de problèmes +65pts
- s2 > s1
  Résolution de problèmes +65pts
- Jointure rapide
  Résolution de problèmes +65pts
- Plus grande jointure
  Résolution de problèmes +65pts
- Jointure asymétrique 1
  Résolution de problèmes +65pts
- Jointure asymétrique 2
  Résolution de problèmes +65pts
- Nombres premiers
  Résolution de problèmes +70pts
- Grands nombres
  Résolution de problèmes ~~+70pts~~
- Même point de départ
  Fiabilité +70pts

Question 21: C'est au fruit que l'on connaît l'arbre

Java 25:00 / 25:00 0 / 300 pts

Le temps alloué à cette question s'est écoulé. La réponse du candidat a été automatiquement récupérée à la fin du décompte.

Question

Un arbre est composé de noeuds qui respectent les règles suivantes : Un noeud tient une valeur correspondant à un entier. Hormis le noeud à la racine de l'arbre, un noeud a toujours un seul autre noeud qui le référence. Un noeud n'a pas plus de deux enfants, appelés noeud à gauche et noeud à droite. Si un noeud n'a pas d'enfant à droite ou à gauche, alors la référence correspondante est ***null***. La valeur tenue par tout enfant du sous arbre à gauche est inférieure à la valeur de son parent et la valeur tenue par tout enfant du sous arbre à droite est supérieure à la valeur de son parent. Voici un exemple d'arbre qui respecte ces règles (la racine vaut 9) :

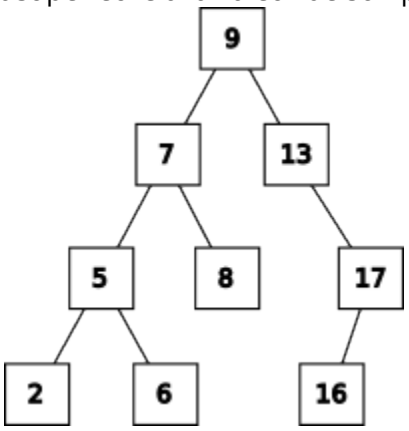


Fig. 1 Pour simplifier le code, tout est combiné dans une simple classe nommée ***Node***. La hauteur de l'arbre (la distance entre le noeud le plus éloigné et la racine) est comprise entre 0 et 100 000 noeuds. Question : Implémentez une nouvelle méthode de ***Node*** nommée ***find(int v)*** qui retourne le noeud tenant la valeur ***v***. Si ce noeud n'existe pas alors ***find*** devra retourner ***null***. **Important :** Essayez de privilégier un faible usage de la mémoire RAM. Pour tester votre algorithme, vous pouvez vous exercer à partir de deux exemples d'arbres : L'arbre présenté ci-dessus (Fig. 1) : la variable ***small*** correspond au noeud racine. Un arbre d'une hauteur de 100 000 noeuds : la variable ***large*** correspond au noeud racine, sachant que le noeud le plus éloigné tient la valeur ***0***.

Réponse

```
class Node {

    // keep these fields
    Node left, right;
  int value;

    public Node find(int num){
      //  if(this.value == num) return ;
      if(this.left.value == num) return this.left;

      if(this.right.value == num) return this.right;
        if(num < this.value){
            if(this.left != null){
                this.left.find(num);
            }else return null;
        }else{
            if(this.right != null){
                this.right.find(num);
            }else return null;
        }
        return this;

    }
}
```

Résultat

✗ Les résultats sont correctes avec un arbre d'une 'petite' hauteur
Résolution de problèmes ~~+90pts~~

NA Les résultats sont correctes avec un arbre d'une 'grande' hauteur
Résolution de problèmes ~~+195pts~~

NA Si le noeud demandé n'existe pas 'null' est retourné
Fiabilité ~~+15pts~~

Question 22: Mangez à volonté. Payez ce que vous souhaitez.

Cross 01:24 / 25:00 300 / 300 pts

Question

Pour attirer les touristes, un Casino de Las Vegas propose un buffet à volonté où chacun paye ce qu'il souhaite payer.

Sachant que vous connaissez ce que chaque touriste est prêt à payer, on vous demande de calculer les gains du restaurant pour la journée :
En début de journée le restaurant est vide. Un touriste arrive, est placé, mange, paye et s'en va. Il y a seulement ***nbSeats*** places disponibles. Un touriste peut manger et payer seulement s'il peut être placé. Si un touriste arrive au restaurant alors qu'il n'y a plus de places disponibles : soit il attend son tour jusqu'à ce qu'une place se libère soit il attend un moment et part avant qu'une place se libère Un touriste peut venir plusieurs fois dans la journée, dans ce cas il ne paye au maximum qu'une seule fois.

Implémentez la méthode ***computeDayGains(nbSeats, payingGuests, guestMovements)*** qui renvoie les gains de la journée :
Le tableau ***payingGuests*** contient ce que chaque touriste est prêt à payer. Par exemple si ***payingGuests[5]*** vaut 25, cela veut dire que le touriste avec l'idenfiant ***5*** est prêt à payer 25$ pour le buffet. Le tableau ***guestMovements*** donne l'ordre des arrivées et départs des touristes. La première fois que vous trouvez un identifiant dans le tableau, il s'agit d'une arrivée. La deuxième fois, il s'agit d'un départ. Un arrivée est toujours suivie d'un départ dans la journée.

Réponse

```
import java.util.*;
import java.io.*;
import java.math.*;

class Solution {

    public static int computeDayGains(int nbSeats, int[] payingGuests, int[] guestMovements
) {
        // Write your code here
        // To debug: System.err.println("Debug messages...");

       int dayGains=0;
        List<Integer> currentAttendancy= new ArrayList<Integer>();
        List<Integer> currentQueue= new ArrayList<Integer>();
        List<Integer> alreadyAte = new ArrayList<Integer>();

         for (int i = 0; i < guestMovements.length; i++)
        {

             if (!currentAttendancy.isEmpty() && currentAttendancy.contains(guestMovements[
i]))
```

```
            {
                if (!currentQueue.isEmpty())
                {
                    if (!alreadyAte.contains(currentQueue.get(0)))
                    {
                        dayGains += payingGuests[currentQueue.get(0)];
                        alreadyAte.add(currentQueue.get(0));
                    }

                    currentAttendancy.add(currentQueue.get(0));
                    currentQueue.remove(currentQueue.get(0));

                }
                int ind=currentAttendancy.indexOf(guestMovements[i]);

                currentAttendancy.remove(ind);
            } else if (currentQueue.contains(guestMovements[i]))
            {
                int ind=currentQueue.indexOf(guestMovements[i]);
                currentQueue.remove(ind);
            }
            else
            {
                if (currentAttendancy.size() < nbSeats)
                {
                    if (!alreadyAte.contains(guestMovements[i]))
                    {
                        dayGains += payingGuests[guestMovements[i]];
                        alreadyAte.add(guestMovements[i]);
                    }

                    currentAttendancy.add(guestMovements[i]);
                }
                else
                {
                    currentQueue.add(guestMovements[i]);
                }
            }

        }




        return dayGains;
    }

    /* Ignore and do not change the code below */
    // #region main
    public static void main(String args[]) {
        Scanner in = new Scanner(System.in);
        int nbSeats = in.nextInt();
        int nbGuests = in.nextInt();
        int nbMovements = in.nextInt();
        int[] payingGuests = new int[nbGuests];
        for (int i = 0; i < nbGuests; i++) {
            payingGuests[i] = in.nextInt();
        }
        int[] guestMovements = new int[nbMovements];
        for (int i = 0; i < nbMovements; i++) {
            guestMovements[i] = in.nextInt();
        }
        PrintStream outStream = System.out;
        System.setOut(System.err);
        int res = computeDayGains(nbSeats, payingGuests, guestMovements);
        System.setOut(outStream);
        System.out.println(res);
    }
    // #endregion
```

```
89 }
```

Résultat

Assez de places
Résolution de problèmes +50pts

Les touristes reviennent
Résolution de problèmes +50pts

File d'attente
Résolution de problèmes +50pts

Trop d'attente
Résolution de problèmes +50pts

Buffet fermé
Fiabilité +25pts

Un touriste paye à la 2ème visite
Fiabilité +25pts

Plusieurs venues
Résolution de problèmes +50pts

Problème de lisibilité de code détecté

Avoid really long methods.

Question 23: Le danseur

Java 03:55 / 30:00 400 / 400 pts

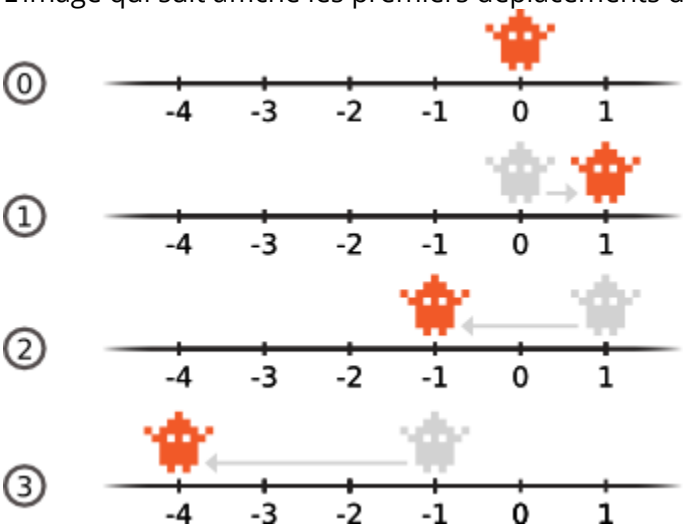
Question

Apprenons une nouvelle danse !

La première chose qu'on doit savoir c'est qu'il faut toujours se déplacer soit en avant, soit en arrière et que dans ce cas, le nombre de pas s'exprime négativement.

Cette danse impose de respecter un enchaînement précis d'étapes : Etape 0 : S'éloigner des obstacles environnant en s'installant à la position 0 Etape 1 : Effectuer un pas en avant (+1 pas) Etape 2 : Effectuer deux pas en arrière (-2 pas) Pour les étapes suivantes, les pas à effectuer ainsi que leur direction, sont donnés par le résultat de l'étape précédente moins le résultat de l'étape qui précède la précédente. Ainsi, à l'étape 3 par exemple, le danseur doit faire 3 pas en arrière (-2 - 1).

L'image qui suit affiche les premiers déplacements du danseur :



A l'étape 3, le danseur se trouve à la position -4.

Implémentez la méthode ***int Algorithm.getPositionAt(int n)*** qui retournera la position du danseur à l' étape ***n***.

Données : ***0 <= n <= 2147483647***
Important : Essayez de limiter l'usage de la mémoire RAM et du CPU.

Exemples :
Algorithm.getPositionAt(3) retourne ***-4***
Algorithm.getPositionAt(100000) retourne ***-5***
Algorithm.getPositionAt(2147483647) retourne ***1***

Réponse

```
class Algorithm {

    /** Computes the position of the dancer. */
  static int getPositionAt(int n) {
        switch (n%6)
            {
                case 0:
                    return 0;
                case 1:
                    return 1;
                case 2:
                    return -1;
                case 3:
                    return -4;
                case 4:
                    return -5;
                case 5:
                    return -3;
            }
            return 0;

  }
}
```

Résultat

Les positions sont correctes avec un jeu de données simple
Résolution de problèmes +133pts

Les positions aux étapes 0, 1 et 2 sont correctes
Fiabilité +15pts

La complexité algorithmique est d'ordre O(1)
Résolution de problèmes +59pts

La position reste correcte pour n autour de 100 000
Résolution de problèmes +193pts

Problème de lisibilité de code détecté

The utility class name 'Algorithm' doesn't match '[A-Z][a-zA-Z0-9]+(Utils?|Helper)'

Question 24: String parser

Java 02:04 / 30:00 328 / 400 pts

Question

Cet exercice consiste à identifier une chaine de caractères composée de parenthèses () et de crochets []. Une chaine de ce type est considérée comme correcte : si c'est une chaine vide ou ***null*** si la chaine A est correcte, (A) et [A] sont correctes si les chaines A et B sont correctes, la concaténation AB est également correcte Données : La chaine contient au plus 10 000 caractères.

Exemples : ***[()]*** est correcte, ***(()[])*** est correcte, ***([)]*** n'est pas correcte, ***((*** n'est pas correcte.

Implémentez la méthode ***check(String str)*** qui devra retourner ***true*** si la chaine passée en paramètre est correcte, sinon elle retournera ***false***.

Réponse

```java
import java.util.*;

class A {

    /**
     * Checks that the given string is correct.
     */
  static boolean check(String str) {
    Deque<String> pile = new ArrayDeque<>();
        char[] tableau = String.valueOf(str).toCharArray();
        List<String> lStrings = new ArrayList<>();

        for (char c : tableau) {
            lStrings.add(Character.toString(c));
        }
        for (String element : lStrings) {
            if(element.equals("(") || element.equals("[")){
                pile.add(element);
            }else{
                String correspondant = pile.getLast();
                pile.removeLast();
                if(correspondant.equals("(") && element.equals(")")) {
                    continue;
                }else if(correspondant.equals("[") && element.equals("]")){
                    continue;
                }else{
                    return false;
                }
            }
        }

        if(pile.size()>0){
            return false;
        }
        return true;
  }
}
```

Résultat

L'algorithme fonctionne avec un jeu de données simple
Résolution de problèmes +204pts

- Une chaine vide retourne true
  Fiabilité +29pts
- Fonctionne dans les temps et sans débordement de pile avec une longue chaîne
  Résolution de problèmes +95pts
- Fonctionne toujours si plus de fermeture que d'ouverture (pile vide)
  Fiabilité ~~+43pts~~
- 'null' retourne true
  Fiabilité ~~+29pts~~

Question 25: Rendu de monnaie

Java 22:23 / 40:00 240 / 400 pts

Question

Les supermarchés s'équipent de plus en plus de caisses automatiques. La plupart de ces caisses n' acceptent que le paiement par carte bancaire bien qu'une part non négligeable de consommateurs paye encore en espèces (avec des billets et des pièces).

Une des problématiques rencontrées avec le paiement en espèces est le rendu de monnaie : comment rendre une somme donnée de façon optimale, c'est-à-dire avec le nombre minimal de pièces et billets ? C'est un problème qui se pose à chacun de nous quotidiennement, à fortiori aux caisses automatiques.

Dans cet exercice, on vous demande d'essayer de trouver une solution optimale pour rendre la monnaie dans un cas précis : quand une caisse automatique ne contient que des pièces de **2€**, des billets de **5€** et de **10€**.

Pour simplifier le problème, nous considérerons que toutes ces pièces et billets sont disponibles en **quantité illimitée**.

Voici quelques exemples de rendu de monnaie :
Monnaie à rendre Solutions possibles Solution optimale 1 Impossible Impossible 6 2 + 2 + 2 2 + 2 + 2 10 2 + 2 + 2 + 2 + 2 5 + 5 10 10 9223372036854775807 … (10 * 922337203685477580) + 5 + 2

Le rendu de monnaie est exprimé par un objet ***Change***. Cet objet a 3 propriétés : ***coin2***, ***bill5*** et ***bill10*** qui, respectivement, stockent le nombre de pièces de 2€, de billets de 5€ et de billets de 10€.

Par exemple, si on reprend l'exemple n°2 du tableau (6€), on devrait obtenir un objet ***Change*** avec : ***coin2*** vaut 3 (3 pièces de 2€) ***bill5*** vaut 0 (pas de billet de 5€) ***bill10*** vaut 0 (pas de billet de 10€) Implémentez la méthode ***optimalChange(long s)*** qui retourne un objet ***Change*** contenant les pièces et billets dont la somme vaut ***s***. S'il est impossible de rendre la monnaie (comme dans l'exemple n°1), retournez ***null***.

Pour obtenir un maximum de points votre solution devra toujours rendre la monnaie quand c'est possible et avec le nombre minimal de pièces et billets.

Données : ***s*** est toujours un entier (***long***) strictement positif inférieur ou égal à 9223372036854775807

Réponse

```java
import java.util.*;
import java.math.*;

// Do not modify Change
class Change {
    long coin2 = 0;
    long bill5 = 0;
    long bill10 = 0;
}

class Solution {

    // Do not modify this method signature
    static Change optimalChange(long s) {
        Change change = new Change();
        if(s%2==0 || s%5==0 || s%10==0){
            if(s%10==0){
                change.bill10 = s/10;
                s = s%10;
            }
            if(s%5==0){
                change.bill5 = s/5;
                s = s%5;
            }
            if(s%2==0){
                change.coin2 = s/2;
                s = s%2;
            }
        return change;
        }else return null;
    }
}
```

Résultat

La monnaie est correcte pour une somme de 10€
Résolution de problèmes +40pts

- La monnaie est optimale pour une somme de 10€ (1*10)
  Résolution de problèmes +40pts
- ***null*** est retourné quand la somme vaut 1
  Fiabilité +40pts
- Le programme rend correctement la monnaie quand la somme vaut 31
  Résolution de problèmes ~~+40pts~~
- La monnaie est optimale pour une somme de 31€ (2*10 +5 + 3*2)
  Résolution de problèmes ~~+40pts~~
- ***null*** est retourné quand la somme vaut 3
  Fiabilité +40pts
- Le programme rend correctement la monnaie quand la somme vaut 8 (4 * 2)
  Résolution de problèmes +40pts
- Résultat correct et dans les temps avec 92233720368547758O2€
  Résolution de problèmes +40pts
- Résultat correct et dans les temps avec 92233720368547758O3€
  Résolution de problèmes ~~+40pts~~
- La monnaie est optimale avec 92233720368547758O3€
  Résolution de problèmes ~~+40pts~~

Problème de lisibilité de code détecté

The utility class name 'Solution' doesn't match '[A-Z][a-zA-Z0-9]+(Utils?|Helper)'

Question 26: Fermeture sécurisée d'un flux

Java 05:00 / 05:00 200 / 300 pts

Le temps alloué à cette question s'est écoulé. La réponse du candidat a été automatiquement récupérée à la fin du décompte.

Question

StreamPrinter.print(Reader reader) n'est pas robuste.

Améliorez ***StreamPrinter.print(Reader reader)***.

Réponse

```java
import java.io.IOException;
import java.io.Reader;

/**
 * This class defines a stream printer.
 */
class StreamPrinter {

    /**
     * Reads from the given reader and print to stdout.
     */
  void print(Reader reader) throws IOException {
    try{
      int code = -1;
      while (reader.read()!=-1) {
        System.out.print((char)code);
      }
    }catch(IOException e){
      e.printStackTrace();
    }finally{
      try{
        reader.close();
      }catch(IOException e){
        e.printStackTrace();
      }
    }
  }

}
```

Résultat

Le flux est fermé en cas d'exception.
Connaissance du langage +100pts

L'exception levée est passée à l'appelant.
Fiabilité ~~+100pts~~

Le flux est fermé en cas d'erreur (utilisation probable de finally).
Connaissance du langage +100pts

Question 27: Objet immuable référençant une Date

Java 05:49 / 06:00 0 / 300 pts

Question

Modifiez le corps de certaines méthodes (cela peut inclure le constructeur) de la classe ***Person*** pour que les instances ne soient pas altérables : A partir du moment où une instance de ***Person*** est créée, il ne devrait plus être possible de la modifier.

Réponse

```java
import java.util.Date;

class Person {

  private String name;
  private Date birthDate;

    /**
     * Creates a Person with the given name and birth date.
     */
  public Person(String name, Date birthDate) {
    this.name = name;
    this.birthDate = birthDate;
  }

  public String getName() {
    return name;
  }

  public Date getBirthDate() {
    return birthDate;
  }
}
```

Résultat

birthDate est copié dans le constructeur et dans le getter
Connaissance du langage ~~+300pts~~

Question 28: Calculatrice défaillante

Java 01:36 / 10:00 300 / 300 pts

Question

Calculator.sum(String... numbers) retourne la somme des nombres appartenant au tableau ***numbers***.

numbers contient toujours des nombres flottants valides exprimés sous la forme de chaine de caractères.

L'implémentation actuelle retourne parfois un résultat inexact, par exemple ***Calculator.sum("99.35", "1.10")*** retourne 100.44999999999999.

Modifiez la méthode ***Calculator.sum(String... numbers)*** afin qu'elle retourne toujours la somme exacte.

Réponse

```java
import java.math.*;
import java.util.ArrayList;
import java.util.List;

class Calculator {

    /**
     * Sums an array of numbers.
     *
     * @return the exact sum of the given numbers
     */
  static String sum(String... numbers) {
    BigDecimal total = new BigDecimal("0");
        List<BigDecimal> bigDecimals = new ArrayList<BigDecimal>();
        for (String string : numbers) {
            bigDecimals.add(new BigDecimal(string));
        }

        for (BigDecimal bigDecimal : bigDecimals) {
            total = total.add(bigDecimal);
        }
        return String.valueOf(total);
  }

}
```

Résultat

Retourne la somme exacte en toute circonstance
Connaissance du langage +300pts

Problème de lisibilité de code détecté

The utility class name 'Calculator' doesn't match '[A-Z][a-zA-Z0-9]+(Utils?|Helper)'

Dont create instances of already existing BigInteger and BigDecimal (ZERO, ONE, TEN)

Question 29: L'art ASCII

Java 00:53 / 10:00 300 / 300 pts

Question

AsciiArt.printChar permet d'afficher un et un seul caractère ASCII de A à Z (inclusif) sur plusieurs lignes et colonnes.

Maintenant, on souhaite faire l'opération dans l'autre sens : obtenir un caractère à partir de sa représentation graphique.

Implémentez la méthode ***scanChar(String s)*** afin qu'elle retourne le caractère associé à la représentation graphique fournie par ***AsciiArt.printChar(char c)***. Si ***s*** ne correspond pas à un caractère entre A et Z (inclusif), alors ***scanChar*** devra retourner le caractère ***?***.

Réponse

```java
import java.util.*;
import external.*; // imports the AsciiArt class

class A {

    private static final char[] ALPHABET = {'A', 'B', 'C', 'D', 'E', 'F', 'G'
        , 'H', 'I', 'J', 'K', 'L', 'M', 'N', 'O', 'P', 'Q', 'R', 'S', 'T', 'U', 'V', 'W'
        , 'X', 'Y', 'Z'
    };

    /**
     * @return the char that is represented by s
     */
    static char scanChar(String s) {
        if (s == null) {
            return '?';
        }
        for (int i = 0 ; i < ALPHABET.length ; i++) {
            if(s.equals(AsciiArt.printChar(ALPHABET[i]))) {
                return ALPHABET[i];
            }
        }

        return '?';
  }
}
```

Résultat

Retourne 'A' si et seulement si la chaîne représente un A
Résolution de problèmes +60pts

La solution fonctionne pour tout caractère entre A et Y
Résolution de problèmes +180pts

Le programme retourne '?' si aucun caractère ne correspond à s
Fiabilité +30pts

La solution fonctionne avec 'Z'
Fiabilité +30pts

Problème de lisibilité de code détecté

The utility class name 'A' doesn't match '[A-Z][a-zA-Z0-9]+(Utils?|Helper)'

Question 30: Approximation de π

Java | 11:06 / 12:00 | 300 / 300 pts

Question

Dans cet exercice nous allons calculer une estimation du nombre π (Pi).

La technique est la suivante :
On prend un point P au hasard de coordonnées (x, y) tel que $0 \leq x \leq 1$ et $0 \leq y \leq 1$. Si $x^2 + y^2 \leq 1$, alors le point est à l'intérieur du quart de disque de rayon ***1***, sinon le point est à l'extérieur.

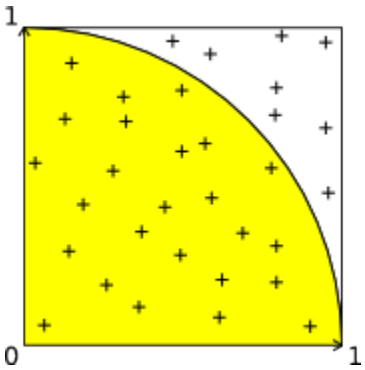


Fig 1. Exemple avec 33 points aléatoires.

On sait que la probabilité que le point se situe à l'intérieur du quart de disque est égale à $\pi/4$.

Écrivez la méthode ***double approx(double[][] pts)*** qui va utiliser les points ***pts*** (tirés au hasard) pour retourner une estimation du nombre π.

Données :
Chaque item de ***pts*** contient un point. Un point est représenté par un tableau contenant exactement deux nombres, respectivement ***x*** et ***y*** tels que $0 \leq x \leq 1$ et $0 \leq y \leq 1$. ***pts*** n'est jamais null et contient toujours au moins un item.

Réponse

```java
/**
 * This class defines a method to approximate pi
 */
class Pi {

    /**
     * Approximate pi using the given points.
     */
    static double approx(double[][] pts) {
        int nbre = 0;
        for(double[] element : pts){
            if(Math.pow(element[0], 2) + Math.pow(element[1], 2) <= 1){
                nbre+=1;
            }
        }
        return (double)(((double)4*(double)nbre)/(double)pts.length);
  }
}
```

Résultat

L'estimation de π est valide (liée aux points fournis)
Résolution de problèmes +257pts

Le point P(1, 0) est à l'intérieur du quart de disque
Fiabilité +43pts

Problème de lisibilité de code détecté

The utility class name 'Pi' doesn't match '[A-Z][a-zA-Z0-9]+(Utils?|Helper)'

Question 31: Chaine en colonne

Java | 11:26 / 12:00 | 265 / 300 pts

Question

La méthode ***reshape(n, str)*** retourne la chaine ***str*** sans les espaces et formatée en lignes de ***n*** caractères maximum.

Exemples :

```
reshape(3, "abc de fghij") => renvoie "abc\ndef\nghi\nj"
reshape(2, "1 23 456") => renvoie "12\n34\n56"
```

Écrivez le corps de la méthode ***reshape(n, str)***.

Note : n'ajoutez pas de caractère ***\n*** final.

Réponse

```java
// Java code below
import java.util.*;
import java.io.*;
import java.nio.*;
import java.math.*;

class Solution {
    public static String reshape(int n, String str) {
        String bg = str.replace(" ", "");
        int j = 1;
        String resultat = "";
        for(int i = 0; i < bg.length(); i++){
            resultat+=bg.charAt(i);
            if(j == n){
                resultat+="\n";
                j = 1;
            }else{
                j++;
            }
        }
        return resultat;
    }

}
```

Résultat

Tests simples
Résolution de problèmes +165pts

Tests espaces et lignes uniques
Résolution de problèmes +100pts

Tests caractère final
Résolution de problèmes ~~+35pts~~

Problème de lisibilité de code détecté

The utility class name 'Solution' doesn't match '[A-Z][a-zA-Z0-9]+(Utils?|Helper)'

Question 32: Communication entre threads

Java 02:15 / 12:00 0 / 300 pts

Question

Un ***WaterTankMonitor*** (un surveillant de réservoir d'eau) a pour objectif de surveiller l'accès à un ***WaterTank*** (un réservoir d'eau).
Une instance de ***WaterTankMonitor*** est partagée par des threads concurrents, opérant à des périodes de temps aléatoires et dans un ordre imprévisible.
Ces threads appartiennent à l'une de ces deux familles : les ***consommateurs*** (qui veulent vider le réservoir d'eau) ou, les ***producteurs*** (qui veulent remplir le réservoir d'eau). Le problème consiste à s'assurer que les producteurs n'essayeront pas de remplir le réservoir s'il est déjà rempli et que les consommateurs n'essayeront pas de retirer de l'eau du réservoir s'il est vide.

Corriger ***WaterTankMonitor*** en implémentant une solution élégante.

Réponse

```
class WaterTankMonitor {

  WaterTank tank; // water tank monitored

  WaterTankMonitor(WaterTank tank) {
    this.tank = tank;
  }

  void empty() throws InterruptedException {
    while (tank.isEmpty()) {
    }

    tank.setEmpty(true);
  }

  void fill() throws InterruptedException {
    while (!tank.isEmpty()) {
    }

    tank.setEmpty(false);
  }
}
```

Résultat

Les threads communiquent correctement par notifications
Résolution de problèmes ~~+300pts~~

Question 33: FizzBuzz

Java 07:41 / 12:00 0 / 300 pts

Question

Écrivez le corps de la méthode ***fizzBuzz(number, map)***.

number est un entier et ***map*** est une map non null.

La méthode ***fizzBuzz*** retourne une chaine de caractères. Cette chaîne est la concaténation des valeurs de ***map*** associées aux clés qui sont des diviseurs de l'entier ***number*** passé en paramètre (par ordre croissant des diviseurs).

S'il n'y a pas de clés qui peuvent diviser ***number***, la méthode doit retourner la représentation en chaine de caractères de ***number***.

On a ***1 <= number <= 100***.

Réponse

```java
import java.util.*;
import java.io.*;
import java.nio.*;
import java.math.*;

class Solution {

    /**
     * Concatenates values associated with divisors of number.
     */
    public static String fizzBuzz(int number, Map<Integer, String> map) {
        List<Integer> cle = new ArrayList<Integer>();



        return Integer.toString(number);
    }
}
```

Résultat

Un élément dans la map
Résolution de problèmes ~~+85pts~~

Deux éléments dans la map
Résolution de problèmes ~~+86pts~~

Cinq éléments dans la map
Résolution de problèmes ~~+43pts~~

Élements non premier
Fiabilité ~~+86pts~~

Question 34: Pile avec fuite mémoire

Java 02:14 / 15:00 0 / 450 pts

Question

Cette pile a une fuite de mémoire. Trouvez la cause et appliquez un correctif.

Réponse

```java
import java.util.EmptyStackException;

class Stack {

    // keep these two fields as they are
    private Object[] elements;
  private int size = 0;

  public Stack(int initialCapacity) {
    elements = new Object[initialCapacity];
  }

  public void push(Object object) {
    ensureCapacity();
    elements[size++] = object;
  }

  public Object pop() {
    if (size == 0) {
      throw new EmptyStackException();
    }

    return elements[--size];
  }

  private void ensureCapacity() {
    if (elements.length == size) {
      Object[] old = elements;
      elements = new Object[2 * size + 1];
      System.arraycopy(old, 0, elements, 0, size);
    }
  }
}
```

Résultat

pop() met à null la référence de l'objet dans le tableau
Résolution de problèmes ~~+450pts~~

Question 35: Chiffres différents

Java 02:32 / 25:00 0 / 450 pts

Question

La méthode ***next*** retourne le plus petit entier, supérieur à ***n*** ayant tous ses chiffres différents de ceux de ***n***.

Par exemple ***next(654321)*** doit retourner ***700000***.

Si un tel entier n'existe pas, alors la méthode doit retourner ***-1***.

Écrivez le corps de la méthode ***next(n)***.

Note : n est un entier strictement positif inférieur à 2^31 .

Réponse

```java
// Java code below
import java.util.*;
import java.io.*;
import java.nio.*;
import java.math.*;

class Solution {

    public static int next(int n) {
        // Your code goes here
        return 0;
    }

}
```

Résultat

Petits nombres
Résolution de problèmes ~~+204pts~~

Pas de solution (1234567890)
Résolution de problèmes ~~+123pts~~

Grands nombres
Fiabilité ~~+123pts~~

Question 36: Design pattern 02

Java 03:05 / 25:00 0 / 450 pts

Question

Un de vos collègues a réalisé le programme affiché dans l'éditeur de réponses. ***Program*** sert à obtenir des informations sur du texte, telles que le nombre de mots qu'il contient. Le code actuel fonctionne correctement mais on vous demande de le reprendre pour réaliser une solution qui sera capable d' accepter des nouvelles commandes sans que la classe ***Program*** soit modifiée. En conservant le point d' entrée ***exec(String cmd, String text)*** de la classe ***Program***, élaborez une nouvelle solution que vous appliquerez à la commande ***"CountWords"***. Modifiez en conséquence la classe ***ProgramTester***. Ajoutez des nouvelles classes / interfaces si nécessaire.

Réponse

```
1  class Program {
2
3    String exec(String command, String text) { // keep this method
4      if (!"CountWords".equals(command)) {
5        throw new IllegalArgumentException(command);
6      }
7
8      return WordCounter.count(text);
9    }
10 }
11
12 class ProgramTester {
13   Program p;
14
15   void init() {
16     p = new Program();
17   }
18
19   void testCountWords() {
20     String r = p.exec("CountWords", "Yes we code");
21     assert (r.equals("3"));
22   }
23 }
```

Résultat

Le programme utilise WordCounter mais n'a pas de dépendance vers WordCounter.
Modélisation ~~+397pts~~

Une commande inconnue lève une IllegalArgumentException
Fiabilité ~~+53pts~~

Question 37: File finder

Java 02:03 / 30:00 550 / 600 pts

Question

Vous ne vous rappelez plus de l'endroit où vous avez enregistré le fichier ***universe-formula***. La seule chose dont vous vous souvenez, c'est que vous l'avez mis quelque part dans un sous-répertoire de ***/tmp /documents***.

Implémentez la méthode ***String locateUniverseFormula()*** qui devra retrouver le fichier ***universe-formula*** et retourner son chemin complet (depuis la racine du système de fichiers). ***/tmp/documents*** peut contenir des sous-répertoires imbriqués les uns dans les autres et ***universe-formula*** peut se trouver dans n'importe lequel de ces sous-répertoires. Si ***universe-formula*** n'existe pas, alors votre programme devra retourner ***null***. Exemple :
locateUniverseFormula() retourne ***/tmp/documents/a/b/c/universe-formula*** si ***universe-formula*** est trouvé dans le répertoire ***/tmp/documents/a/b/c***.

Réponse

```java
import java.io.*;

class A {

    /**
     * Locates the universe-formula file.
     */
  public static String locateUniverseFormula() {
    String path = "/tmp/documents/";
    String name = "universe-formula";

    File file = find(path, name);
    return file.getAbsolutePath();
  }

  private static File find(String path, String name) {
    File file = new File(path);
    if (name.equalsIgnoreCase(file.getName())) {
      return file;
    }

    if (file.isDirectory()) {
      for (String aChild : file.list()) {
        File found = find(path + File.separator + aChild, name);
        if (found != null)
          return found;
      }
    }
    return file;
  }

  void print(Reader reader) throws IOException {
    try {
        int code = -1;
        while ((reader.read() != -1)) {
          System.out.print((char) code);
        }
     } catch (IOException e) {
          e.printStackTrace();
     } finally {
        try {
            reader.close();
        } catch (IOException e) {
            e.printStackTrace();
        }
      }
  }

}
```

Résultat

universe-formula est trouvé
Résolution de problèmes +350pts

null est retourné quand universe-formula n'existe pas
Fiabilité ~~+50pts~~

L'algorithme s'arrête de chercher dès que universe-formula est trouvé
Résolution de problèmes +200pts

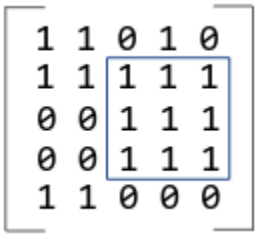
Question 38: Sous-matrice

Cross 06:32 / 30:00 195 / 600 pts

Question

Le but de cet exercice est de chercher et trouver la plus grande sous-matrice carrée composée uniquement de uns (***1***) dans une table donnée.

Par exemple, dans la table suivante, le carré montre le résultat de recherche attendu, c'est-à-dire une sous-matrice de 3 par 3.



Implémentez la méthode ***int findLargestSquare(int[][] matrix)*** qui retourne la taille de la plus grande sous-matrice carrée composée uniquement de uns dans la table à deux dimensions ***matrix*** .

Constraintes:
matrix est une table carrée `0 # taille de` ***matrix*** `# 100`

Réponse

```java
import java.util.*;
import java.io.*;
import java.math.*;

class Solution {

    public static int findLargestSquare(int[][] matrix) {
        // Write your code here
        // To debug: System.err.println("Debug messages...");

        return 0;
    }

    /* Ignore and do not change the code below */
    // #region main
    public static void main(String args[]) {
        Scanner in = new Scanner(System.in);
        int n = in.nextInt();
        int[][] matrix = new int[n][n];
        for (int i = 0; i < n; i++) {
            for (int j = 0; j < n; j++) {
                matrix[i][j] = in.nextInt();
            }
        }
        PrintStream outStream = System.out;
        System.setOut(System.err);
        int size = findLargestSquare(matrix);
        System.setOut(outStream);
        System.out.println(size);
    }
    // #endregion
}
```

Résultat

Matrice simple
Résolution de problèmes ~~+135pts~~

Que des uns
Résolution de problèmes ~~+135pts~~

Un trou de côté
Résolution de problèmes ~~+135pts~~

Que des zéros
Résolution de problèmes +130pts

Matrice vide
Fiabilité +65pts

Question 39: Parsing de fonction

Cross 02:32 / 35:00 60 / 600 pts

Question

Pour une fonction polynome du second degrée donnée sous la forme $f(x)=ax^2+bx+c$, on souhaite calculer le résultat de cette fonction pour une valeur de x.

Par exemple, pour la fonction $simple(x)=x^2+x+1$, et $x=4$, le résultat attendu est 21.

Note : certains coefficients peuvent être manquant, comme dans cet exemple : $linear(x)=-3x+2$

Implémentez la méthode ***int applyFunction(String quadraticFunction, int x)*** qui prend une fonction polynome et une valeur ***x*** en paramètres et retourne la valeur de cette fonction pour cette valeur x.

Constraintes:
-1000 < a,b,c,x < 1000 Les produits du polynome sont toujours donnés dans le même ordre : les plus grands exposants d'abord. Par exemple $k(x)=-2+3x-4x^2$ n'est pas un paramètre valide

Réponse

```java
import java.util.*;
import java.io.*;
import java.math.*;

/**
 * Compute the value of a function for a given value of x.
 * Help: the character 'square' used is this one: ² (U+00B2)
 **/
class Solution {

    public static int applyFunction(String quadraticFunction, int x) {
        // Write your code here
        // To debug: System.err.println("Debug messages...");

        return 0;
    }

    /* Ignore and do not change the code below */
    // #region main
    public static void main(String args[]) {
        Scanner in = new Scanner(System.in);
        String quadraticFunction = in.next();
        int x = in.nextInt();
        PrintStream outStream = System.out;
        System.setOut(System.err);
        int result = applyFunction(quadraticFunction, x);
        System.setOut(outStream);
        System.out.println(result);
    }
    // #endregion
}
```

Résultat

- h(x)=3x²+4x+5
  Résolution de problèmes ~~+60pts~~
- solve(x)=8x²+x+1
  Résolution de problèmes ~~+60pts~~
- k(x)=5x²-8
  Résolution de problèmes ~~+60pts~~
- notcube(x)=x²
  Résolution de problèmes ~~+60pts~~
- c(x)=4
  Résolution de problèmes ~~+60pts~~
- fct(x)=-48x²-27x+12
  Résolution de problèmes ~~+60pts~~
- min(x)=-x²-x-1
  Résolution de problèmes ~~+60pts~~
- strait(x)=-90x+200
  Résolution de problèmes ~~+60pts~~
- big(x)=998x²+999x+998
  Résolution de problèmes ~~+60pts~~
- z(x)=0
  Fiabilité +60pts

Question 40: Tas de sable

Java 01:24 / 40:00 0 / 600 pts

Question

Un tas de sable est représenté par une grille composée de nombres entiers allant de 0 à 3, ces nombres entiers représentent le nombre de grains de sable par case de la grille.

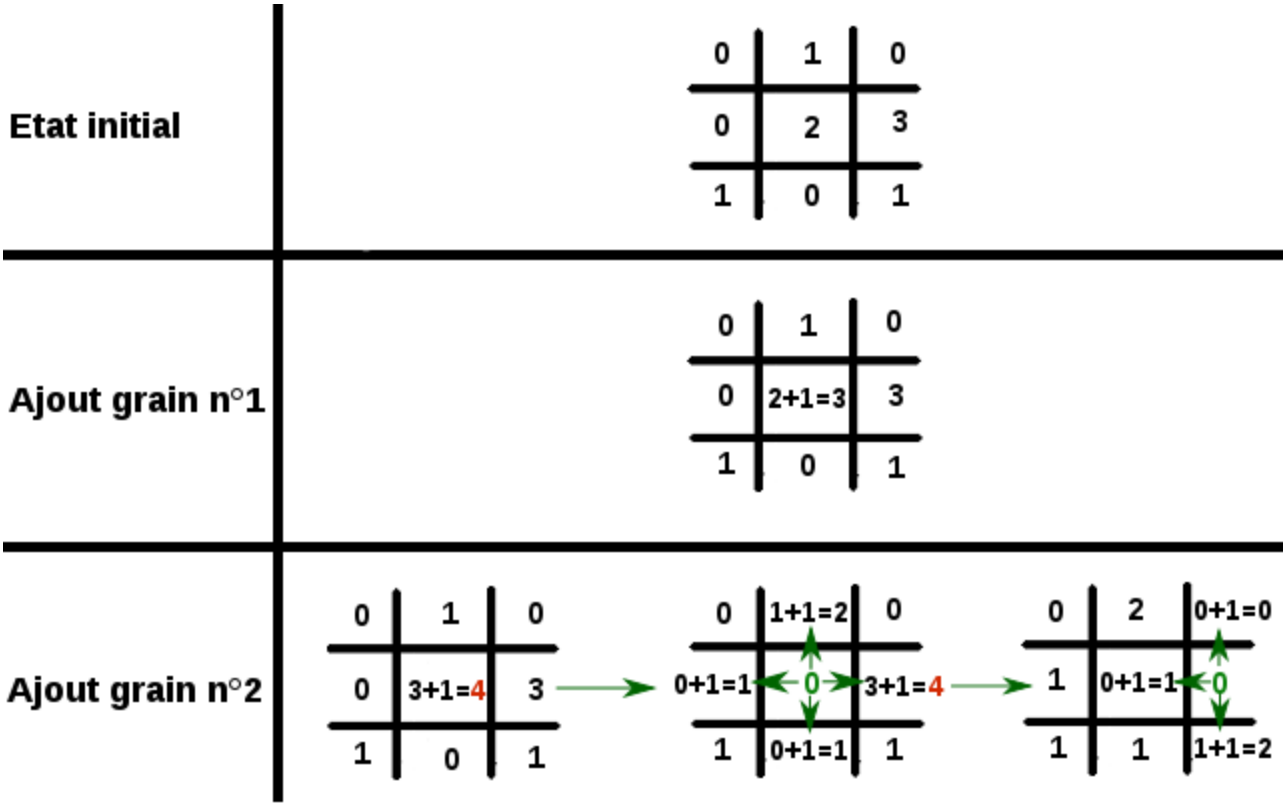
La méthode ***sandpile(pile, n)*** ajoute ***n*** grains de sable au tas ***pile***. ***pile*** est un tableau d'entiers à 2 dimensions égales et impaires.

Les ***n*** grains sont ajoutés un par un au centre du tas.

A chaque fois qu'un grain est ajouté à une case, si le nombre de grains d'une case atteint 4, la case distribue un de ses grains à chacun de ses voisins situés au-dessus et en dessous, ainsi qu'à gauche et à droite.
Si la case n'a pas 4 voisins, elle perd quand meme l'ensemble de ses grains (les grains sont perdus sur les bords du tas).

Les cases voisines qui atteignent 4 grains redistribuent à leur tour les grains dans leurs cases voisines. Les grains se propagent ainsi vers les bords du tas.

Par exemple pour ***n = 2*** :



La méthode ***sandpile()*** doit renvoyer un tableau d'entiers à 2 dimensions représentant le nouveau tas de sable après ***n*** itérations.

Écrivez le corps de la méthode ***sandpile(pile, n)***.

1 <= pile width <= 10
1 <= n <= 100

Réponse

```java
import java.util.*;
import java.io.*;
import java.nio.*;
import java.math.*;

class Solution {

    /**
     * Returns the new state of the sand pile after adding n grains
     */
  public static int[][] sandpile(int[][] pile, int n) {
        return pile;
    }

}
```

Résultat

1 case
Résolution de problèmes ~~+43pts~~

9 cases
Résolution de problèmes ~~+129pts~~

25 cases
Résolution de problèmes ~~+214pts~~

81 cases
Résolution de problèmes ~~+214pts~~

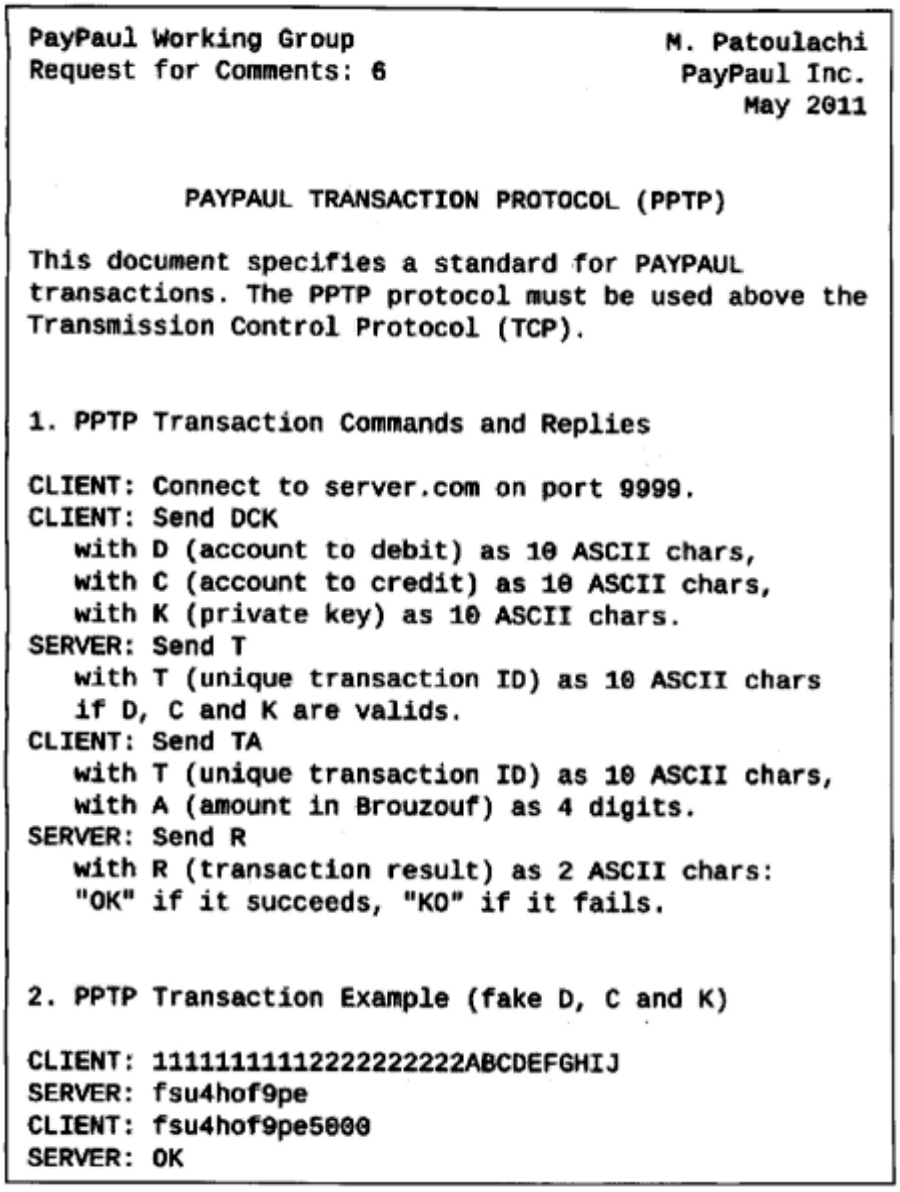
Question 41: Attaque sur PayPaul !

Java 03:26 / 120:00 0 / 600 pts

Question

Vous avez été retenu pour mener une attaque informatique visant à déstabiliser un sombre groupuscule de trafiquants de lance-missiles USB. Les malfrats ont réussi à amasser une importante quantité de brouzoufs (monnaie locale) en envahissant les espaces de travail de millions d'informaticiens. Le résultat est accablant : les projets informatiques sont retardés et la tension est à son comble dans les open-spaces. C'est toute l'industrie du logiciel qui est aujourd'hui menacée. Votre mission, si toutefois vous

l'acceptez, consiste à rétablir l'ordre en menant une attaque de front sur le compte bancaire des trafiquants. La stratégie adoptée est de vider leur compte en effectuant des transactions vers un autre compte afin d'anéantir leur puissance financière. Les numéros de compte vous seront transmis à la dernière minute. **Mission #1 : Identifier une faille de sécurité dans le système PayPaul.** La première étape de votre mission consiste à identifier une faille dans le système de transactions bancaires fourni par PayPaul, entreprise de paiements en ligne de renommée internationale. Pour ce faire, vous disposez de peu d'informations : 1. Un listing de quelques exemples de transactions dérobées chez PayPaul. ACCOUNT TO DEBIT (D) ACCOUNT TO CREDIT (C) PRIVATE KEY (K) 374c5a446f 5872303531 **7LZDoXr051** 6339314b67 4c48483673 **c91KgLHH6s** 356e5a7263 4841473737 **5nZrcHAG77** 2. Un document technique également dérobé chez PayPaul.



PayPaul Working Group — M. Patoulachi
Request for Comments: 6 — PayPaul Inc.
May 2011

PAYPAUL TRANSACTION PROTOCOL (PPTP)

This document specifies a standard for PAYPAUL transactions. The PPTP protocol must be used above the Transmission Control Protocol (TCP).

1. PPTP Transaction Commands and Replies

CLIENT: Connect to server.com on port 9999.
CLIENT: Send DCK
with D (account to debit) as 10 ASCII chars,
with C (account to credit) as 10 ASCII chars,
with K (private key) as 10 ASCII chars.
SERVER: Send T
with T (unique transaction ID) as 10 ASCII chars
if D, C and K are valids.
CLIENT: Send TA
with T (unique transaction ID) as 10 ASCII chars,
with A (amount in Brouzouf) as 4 digits.
SERVER: Send R
with R (transaction result) as 2 ASCII chars:
"OK" if it succeeds, "KO" if it fails.

2. PPTP Transaction Example (fake D, C and K)

CLIENT: 1111111111222222222ABCDEFGHIJ
SERVER: fsu4hof9pe
CLIENT: fsu4hof9pe5000
SERVER: OK

Mission #2 : Transférer un maximum de brouzoufs en un minimum de temps. Pour mener à bien votre mission, vous devez impérativement considérer les points suivants : Pour ne pas alerter la sécurité PayPaul, chaque montant de transaction ne doit pas dépasser 5000 Brouzoufs. Quand les comptes vous seront donnés, vous ne disposerez que de 1 seconde maximum pour effectuer des transactions ! Entre le moment où vous ouvrirez une transaction et le moment où elle sera terminée, il faudra compter une

durée d'environ 600 millisecondes (échanges réseau + traitement bancaire). D'après nos informateurs, le compte bancaire des trafiquants cumule 25 000 brouzoufs. Pour anéantir leur commerce frauduleux il faudrait essayer leur extorquer la totalité de leur argent. Implémentez la méthode ***start(String debitAcc, String creditAcc)*** qui sera le point d'entrée de votre programme (ne pas modifier la signature de cette méthode). L'avenir de l'informatique étant en jeu, il est crucial de vous assurer que votre programme fonctionnera le moment voulu. ***Entraînez-vous autant que vous le voulez à partir des numéros de comptes d'essais qui sont fournis dans l'éditeur de test.*** Bonne chance.

Réponse

```
import java.io.*;
import java.net.*;

class MissionImpossible {

    final static String SERVER = "server.com";
  final static int PORT = 9999;

    // do not modify the signature of this method
  static void start(String debitAcc, String creditAcc)
  throws IOException {

  }
}
```

Résultat

✗ Des brouzoufs sont crédités en utilisant les comptes de test
Résolution de problèmes ~~+165pts~~

NA Commerce frauduleux anéanti (25000 brouzoufs)
Résolution de problèmes ~~+141pts~~

NA Pas d'alerte sur le server PayPaul
Fiabilité ~~+12pts~~

NA La clé privée est toujours correcte
Résolution de problèmes ~~+188pts~~

NA Le programme fonctionne avec des packets fragmentés
Fiabilité ~~+94pts~~

Glossaire

Connaissance du langage

La mesure de cette compétence permet de déterminer l'expérience du candidat dans la pratique d'un langage de programmation. **Privilégiez cette compétence si, par exemple, vous recherchez un développeur qui devra être rapidement opérationnel.**

Modélisation

Cette mesure fournit une indication sur la capacité du candidat à appliquer des solutions standard pour résoudre des problèmes récurrents. Un développeur ayant un bon niveau dans cette compétence augmentera la qualité (maintenabilité, évolutivité) de vos applications. Cette compétence ne dépend pas spécifiquement d'une technologie. **Privilégiez cette compétence si, par exemple, vous recherchez un développeur qui sera amené à travailler sur les briques qui structurent vos applications, à anticiper les besoins de demain pour développer des solutions pérennes.**

Résolution de problèmes

Cette compétence correspond aux aptitudes du candidat à comprendre et à structurer son raisonnement pour trouver des solutions à des problèmes complexes. Cette compétence ne dépend pas spécifiquement d'une technologie. **Privilégiez cette compétence si, par exemple, vos applications ont une composante technique importante (R&D, innovation).**

Fiabilité

La fiabilité caractérise la capacité du candidat à réaliser des solutions qui prennent en compte les cas particuliers. Plus cette compétence est élevée, plus vos applications sont robustes (moins de bugs).