Commencé le	mardi 7 décembre 2021, 13:31
État	Terminé
Terminé le	mardi 14 décembre 2021, 15:18
Temps mis	7 jours 1 heure
Points	15,00/17,00
Note	**17,65** sur 20,00 (**88**%)

Question **1**

Correct

Note de 1,00 sur 1,00

Soit E le sous ensemble de $\{a,b\}^*$ défini inductivement par
la base B={ε} et
les opérateurs Ω ={ω_a, ω_{bb}} avec $\omega_a(m)$=ma et $\omega_{bb}(m)$=mbb

Cochez les affirmations exactes et elles seules.

Veuillez choisir au moins une réponse :

- ☐ Tous les mots de $\{a,b\}^*$ ayant un nombre pair de b sont dans E
- ☑ Tous les mots de E ont un nombre pair de b ✔
- ☐ Tous les mots de E ont plus de b que de a
- ☐ Tous les mots de $\{a,b\}^*$ se terminant par bb sont dans E
- ☐ Aucune des autres réponses n'est vraie

Votre réponse est correcte.

La réponse correcte est : Tous les mots de E ont un nombre pair de b

Question **2**

Correct

Note de 1,00 sur 1,00

Soit E le sous ensemble de $\{a,b\}^*$ défini inductivement par
la base B={ε} et
les opérateurs $\Omega = \{\omega_a, \omega_b\}$ avec $\omega_a(m)$=aam et $\omega_b(m)$=bmb

Cochez les affirmations exactes et elles seules

Veuillez choisir au moins une réponse :

- [ ] Aucune des autres réponses n'est vraie
- [ ] Tous les mots de $\{a,b\}^*$ ayant un nombre pair de b sont dans E
- [x] Tous les mots de E ont un nombre pair de a
- [ ] Tous les mots de $\{a,b\}^*$ ayant un nombre pair de a sont dans E
- [x] Tous les mots de E ont un nombre pair de b

Votre réponse est correcte.

Les réponses correctes sont : Tous les mots de E ont un nombre pair de a, Tous les mots de E ont un nombre pair de b

Question 3

Correct

Note de 1,00 sur 1,00

Soit ***P*** l'ensemble défini sur l'alphabet {+,-} par
la base B = {ε} et
les constructeurs Ω = {***ω***} avec ***ω(u,v) = +u-v***

On peut voir ***P*** comme un ensemble de chemins montagneux avec des "pas" montants de 1 notés + et des "pas" descendants de 1 notés -, par exemple ***++-+++- - - -*** est un chemin où l'on commence par 2 "pas" montants, suivis d'un descendant, de 3 montants, et enfin de 4 descendants.

On veut définir **inductivement** la fonction ***altMax***, telle que ***altMax(u)*** est l'altitude du point culminant du chemin ***u***, par exemple ***altMax(++-+++- - - -)*** = 4.

Dans les réponses ci-dessous ***max{x,y}*** représente le maximum des 2 nombres ***x*** et ***y***.

Cocher les définitions correctes et elles seules

Veuillez choisir au moins une réponse :

- [ ] Aucune des autres réponses proposées
- [x] ***altMax(ε) = 0*** et ***altMax(ω(u,v)) = max{altMax(u)+1, altMax(v)}*** ✔
- [ ] ***altMax(ε) = 0*** et ***altMax(ω(u,v)) = altMax(u) + 1 + altMax(v)***
- [ ] ***altMax(ε) = 0*** et ***altMax(ω(u,v)) = altMax(u) + altMax(v)***
- [ ] ***altMax(ε) = 0*** et ***altMax(ω(u,v)) = max{altMax(u), altMax(v)}***

Votre réponse est correcte.

La réponse correcte est : ***altMax(ε) = 0*** et ***altMax(ω(u,v)) = max{altMax(u)+1, altMax(v)}***

Question **4**

Correct

Note de 1,00 sur 1,00

Soit ***P*** l'ensemble défini sur l'alphabet {+,-} par
la base B = {ε} et
les constructeurs Ω = {***ω***} avec ***ω(u,v) = -u+v***

On peut voir ***P*** comme un ensemble de plongées avec des "pas" descendants de 1 notés - et des "pas" montants de 1 notés +, par exemple **- -+- - - ++++** est une plongée où l'on commence par 2 "pas" descendants, suivis d'un montant, de 3 descendants, et enfin de 4 montants.

On veut définir **inductivement** la fonction ***profMax***, telle que ***profMax(u)*** est la profondeur du point le plus bas de la plongée ***u***, par exemple ***profMax(- -+- - - ++++)*** = 4.

Dans les réponses ci-dessous ***max{x,y}*** représente le maximum des 2 nombres ***x*** et ***y***.

Cocher les définitions correctes et elles seules

Veuillez choisir au moins une réponse :

- ☐ ***profMax(ε) = 0*** et ***profMax(ω(u,v)) = profMax(u) + 1 + profMax(v)***
- ☐ ***profMax(ε) = 0*** et ***profMax(ω(u,v)) = profMax(u) + profMax(v)***
- ☐ Aucune des autres réponses proposées
- ☐ ***profMax(ε) = 0*** et ***profMax(ω(u,v)) = max{profMax(u), profMax(v)}***
- ☑ ***profMax(ε) = 0*** et ***profMax(ω(u,v)) = max{profMax(u)+1, profMax(v)}*** ✔

Votre réponse est correcte.

La réponse correcte est : ***profMax(ε) = 0*** et ***profMax(ω(u,v)) = max{profMax(u)+1, profMax(v)}***

Question **5**

Correct

Note de 1,00 sur 1,00

Soit ***P*** l'ensemble défini sur l'alphabet {+,-} par
la base B = {ε} et
les constructeurs Ω = {***ω***} avec ***ω(u,v) = +u-v***

On peut voir ***P*** comme un ensemble de chemins montagneux avec des "pas" montants de 1 notés + et des "pas" descendants de 1 notés -, par exemple **++-+++- - - -** est un chemin où l'on commence par 2 "pas" montants, suivis d'un descendant, de 3 montants, et enfin de 4 descendants.

On veut définir **inductivement** la fonction ***altCumulee***, telle que ***altCumulee(u)*** est la somme des dénivelés positifs du chemin ***u***, par exemple ***altCumulee(++-+++- - - -)*** = 5.

Dans les réponses ci-dessous ***max{x,y}*** représente le maximum des 2 nombres ***x*** et ***y***.

Cocher les définitions correctes et elles seules

Veuillez choisir au moins une réponse :

- [ ] Aucune des autres réponses proposées
- [ ] ***altCumulee(ε) = 0*** et ***altCumulee(ω(u,v)) = max{altCumulee(u), altCumulee(v)}***
- [x] ***altCumulee(ε) = 0*** et ***altCumulee(ω(u,v)) = 1 + altCumulee(u)+ altCumulee(v)*** ✔
- [ ] ***altCumulee(ε) = 0*** et ***altCumulee(ω(u,v)) = 1 + max{altCumulee(u), altCumulee(v)}***
- [ ] ***altCumulee(ε) = 0*** et ***altCumulee(ω(u,v)) = altCumulee(u)+ altCumulee(v)***

Votre réponse est correcte.

La réponse correcte est : ***altCumulee(ε) = 0*** et ***altCumulee(ω(u,v)) = 1 + altCumulee(u)+ altCumulee(v)***

Question **6**

Correct

Note de 1,00 sur 1,00

Soit ***P*** l'ensemble défini sur l'alphabet {+,-} par
la base B = {ε} et
les constructeurs Ω = {***ω+***, ***ω-***} avec :

- ***ω+(u) = u+***
- ***ω-(u) = u-***

On peut voir ***P*** est un ensemble de chemins montagneux avec des "pas" montants de 1 notés + et des "pas" descendants de 1 notés -, par exemple **++-+++- -** est un chemin où l'on commence par 2 "pas" montants, suivant d'un descendant, de 3 montants, et enfin de 2 descendants.

On veut définir **inductivement** la fonction ***altFinale***, telle que ***altFinale(u)*** est l'altitude atteinte à la fin du chemin ***u***, par exemple ***altFinale(++-+++- -) =*** 2.

Cocher les définitions correctes et elles seules

Veuillez choisir au moins une réponse :

- ☐ Aucune des autres réponses proposées
- ☐ ***altFinale(ε) = 0*** et ***altFinale(ω+(u)) = altFinale(u) + 1*** et ***altFinale(ω-(u)) = altFinale(u)***
- ☐
- ☐
- ☑ ***altFinale(ε) = 0*** et ***altFinale(ω+(u)) = altFinale(u) + 1*** et ***altFinale(ω-(u)) = altFinale(u) - 1*** ✔

Votre réponse est correcte.

La réponse correcte est : ***altFinale(ε) = 0*** et ***altFinale(ω+(u)) = altFinale(u) + 1*** et ***altFinale(ω-(u)) = altFinale(u) - 1***

Question **7**

Correct

Note de 1,00 sur 1,00

Soit f la fonction de ℕ dans ℕ définie inductivement par:

- f(0)=0
- f(1)=1
- pour tout n non nul, f(2n)=2*f(n)
- pour tout n non nul, f(2n+1)=2*f(n)

Cochez les affirmations exactes et elles seules

Veuillez choisir au moins une réponse :

- [x] Pour tout entier n, il y a toujours soit 0 soit exactement n entiers qui sont solutions de f(x)=n ✔
- [x] il existe une infinité d'entiers k qui vérifient f(k)=k ✔
- [x] pour tout n, f(n) est inférieur ou égal à n ✔
- [x] pour tout n> 0 : f(n) est une puissance de 2 ✔
- [ ] f(1357)=569
- [ ] Pour tout entier pair 2k, f(2k)=2k

Votre réponse est correcte.

Les réponses correctes sont : pour tout n> 0 : f(n) est une puissance de 2, il existe une infinité d'entiers k qui vérifient f(k)=k, pour tout n, f(n) est inférieur ou égal à n, Pour tout entier n, il y a toujours soit 0 soit exactement n entiers qui sont solutions de f(x)=n

Question 8

Correct

Note de 1,00 sur 1,00

Soit f la fonction de ℕ dans ℕ définie inductivement par :

- f(0)=0
- f(1)=f(2)=1
- pour tout n non nul, f(3n)=3*f(n)
- pour tout n non nul, f(3n+1)=3*f(n)
- pour tout n non nul, f(3n+2)=3*f(n)

Cochez les affirmations exactes et elles seules.

Veuillez choisir au moins une réponse :

- [x] Pour tout n entier strictement positif, il y a toujours soit 0 soit exactement 2*n entiers qui sont solutions de f(x)=n ✔
- [x] pour tout n > 0, f(n) est une puissance de 3 ✔
- [ ] f(1357)=566
- [x] il existe une infinité d'entiers k qui vérifient f(k)=k ✔
- [x] pour tout n, f(n) est inférieur ou égal à n ✔
- [ ] Pour tout multiple de 3 (n=3k), f(3k)=3k

Votre réponse est correcte.

Les réponses correctes sont : pour tout n > 0, f(n) est une puissance de 3, il existe une infinité d'entiers k qui vérifient f(k)=k, pour tout n, f(n) est inférieur ou égal à n, Pour tout n entier strictement positif, il y a toujours soit 0 soit exactement 2*n entiers qui sont solutions de f(x)=n

Question 9

Correct

Note de 1,00 sur 1,00

Soit f la fonction de l'ensemble des entiers naturels $\mathbb{N}$ dans $\mathbb{N}$, définie inductivement par :

- f(0)=0
- f(1)=1
- pour tout n non nul, f(2n)=2*f(n)
- pour tout n non nul, f(2n+1)=2*f(n)

Cochez les affirmations exactes et elles seules

Veuillez choisir au moins une réponse :

- [ ] f(1023)=511
- [x] pour tout n, f(n) est inférieur ou égal à n
- [x] pour tout n>0, f(n) est une puissance de 2
- [x] f(1025) = 1024
- [x] pour tout entier k, $f(2^k)=2^k$
- [ ] pour tout entier pair 2k, f(2k)=2k

Votre réponse est correcte.

Les réponses correctes sont : pour tout n>0, f(n) est une puissance de 2, pour tout entier k, $f(2^k)=2^k$, pour tout n, f(n) est inférieur ou égal à n, f(1025) = 1024

Question **10**

Correct

Note de 1,00 sur 1,00

On souhaite définir inductivement la fonction S de ℕ dans ℕ,telle que S(n) est égal à la somme des n premiers entiers non nuls.

On a donc S(0)=0, S(1)=1, S(2)=3, S(3)=6, S(4)=10, S(5)=15, etc....

Sélectionnez toutes les définitions correctes de S et elles seules

Veuillez choisir au moins une réponse :

- [x] • S(0)=0 ✔
  • S(n+1)=n+1+S(n)
- [ ] aucune des définitions proposées n'est correcte
- [x] • S(0)=0 ✔
  • S(1)=1
  • S(n+2)=2n+3+S(n)
- [ ] • S(0)=0
  • S(1)=1
  • S(n+3)=3n+6+S(n)
- [x] • S(0)=0 ✔
  • Pour tout n strictement positif S(2n)=4*S(n)-n
  • S(2n+1)=4S(n)+n+1

Votre réponse est correcte.

Les réponses correctes sont :

- S(0)=0
- S(n+1)=n+1+S(n)

,

- S(0)=0
- S(1)=1
- S(n+2)=2n+3+S(n)

,

- S(0)=0
- Pour tout n strictement positif S(2n)=4*S(n)-n
- S(2n+1)=4S(n)+n+1

Question **11**

Correct

Note de 1,00 sur 1,00

On souhaite définir inductivement la fonction SP de ℕ dans ℕ,telle que SP(n) est égal à la somme des n premiers entiers pairs non nuls.

On a donc SP(0)=0, SP(1)=2, SP(2)=6, SP(3)=12, SP(4)=20, SP(5)=30 etc....

Sélectionnez toutes les définitions correctes de SP et elles seules.

Veuillez choisir au moins une réponse :

- [ ] aucune des définitions proposées n'est correcte
- [x] • SP(0)=0 ✔
  • SP(1)=2
  • SP(n+2)=4n+6+SP(n)
- [x] • SP(0)=0 ✔
  • Pour tout n strictement positif SP(2n)=4*SP(n)-2n
  • SP(2n+1)=4SP(n)+2n+2
- [x] • SP(0)=0 ✔
  • SP(n+1)=2(n+1)+SP(n)
- [ ] • SP(0)=0
  • SP(1)=2
  • SP(2)=6
  • SP(n+3)=6n+10+SP(n)

Votre réponse est correcte.

Les réponses correctes sont :

- SP(0)=0
- SP(n+1)=2(n+1)+SP(n)

,

- SP(0)=0
- SP(1)=2
- SP(n+2)=4n+6+SP(n)

,

- SP(0)=0
- Pour tout n strictement positif SP(2n)=4*SP(n)-2n
- SP(2n+1)=4SP(n)+2n+2

Question **12**

Correct

Note de 1,00 sur 1,00

Soit la méthode récursive Java ***fRec*** :

```
public static int fRec(int n) {
   if(n == 0) return 1;
   return n*fRec(n-2);
}
```

Cochez les propotions exactes et elles seules

Veuillez choisir au moins une réponse :

- [x] la méthode ***fRec*** appliquée à 18 retourne 185 794 560 (soit 512 * 9!) ✔
- [ ] la méthode ***fRec*** appliquée à 512 retourne 61 440 (soit 512 * 5!)
- [ ] la méthode ***fRec*** appliquée à tout entier qui n'est pas une puissance de 2, "ne s'arrête pas"
- [x] la méthode ***fRec*** appliquée à 511 "ne s'arrête pas" ✔
- [x] la méthode ***fRec*** appliquée à 6 retourne 48 ✔
- [ ] cette méthode n'est pas syntaxiquement correcte

Votre réponse est correcte.

Les réponses correctes sont : la méthode ***fRec*** appliquée à 6 retourne 48, la méthode ***fRec*** appliquée à 18 retourne 185 794 560 (soit 512 * 9!), la méthode ***fRec*** appliquée à 511 "ne s'arrête pas"

Question **13**

Correct

Note de 1,00 sur 1,00

Pour que la **méthode qui est à compléter**

```
public static void sans00(int k)
```

affiche tous les mots de longueur k ne contenant pas le facteur 00, **écrire la méthode suivante** de signature

```
private static void sans00(String pref, int reste)
```

où :

- pref est un préfixe d'un mot sans facteur 00
- reste est la différence entre k et la longueur de pref.

Les mots affichés sont séparés par un espace (" ").

Par exemple:

Test	Résultat
sans00(1)	1 0
sans00(2)	11 10 01
sans00(3)	111 110 101 011 010

Réponse : (régime de pénalités : 0 %)

Réinitialiser la réponse

```
/**
 * Print out all binary words of length x
 * without '00'
 * Complexity: THETA( C(x) = C(x-1) + C(x-2) )
*/
public static void sans00(int x) {
    for(int i = (int)Math.pow(2,x) ; i>= 0 ;  i--){
        String val = Integer.toString(i, 2);
        if (!val.contains("00")){
            if (val.length() == x-1){
                val = "0" + val ;
                if (val.length() == x){
                    if (!val.contains("00")){
                        System.out.print(val + " ");
                    }
                }
            } else {
                if(val.length() == x){
                    System.out.print(val + " ");
                }
            }
        }
    }
}
```

	Test	Résultat attendu	Résultat obtenu	
✔	sans00(1)	1 0	1 0	✔
✔	sans00(2)	11 10 01	11 10 01	✔
✔	sans00(3)	111 110 101 011 010	111 110 101 011 010	✔

Tous les tests ont été réussis ! ✔

Solution de l'auteur de la question (Java):

```java
/**
 * Print out all binary words of length x
 * without '00'
 * Complexity: THETA( C(x) = C(x-1) + C(x-2) )
*/
public static void sans00(int x) {
    sans00(x,"");
    if (x>0){
        sans00(x-1,"0");
    }
}

/**
 * Print out all binary words of length x
 * without '00'
 * Complexity: THETA( C(x) = C(x-1) + C(x-2) )
*/
private static void sans00(int x, String s) {
    if (x == 0) {
        out.print(s + " ");
    }
    else if (x == 1) {
        out.print(s + "1 ");
    }
```

Correct

Note pour cet envoi : 1,00/1,00.

Question **14**

Correct

Note de 1,00 sur 1,00

On travaille sur les mots binaires qui ont autant de '0' que de '1', par exemple "001011" ou "1001", mais pas "010" ni "0010".

Compléter la classe ***GenMEg*** pour que la méthode

```
public static void genMEg(int n)
```

affiche tous les mots binaires de longueur n ayant autant de '0" que de '1' , à partir d'appel(s) de la méthode suivante

```
private static void genMEg(String pref, int nb0, int nb1, int k)
```

où :

- pref est un préfixe d'un mot binaire ayant autant de '0" que de '1'
- nb0 est le nombre de '0' de pref
- nb1 est le nombre de '1' de pref
- k est la différence entre n et la longueur de pref.

Les mots affichés sont séparés par le caractère '|'
Par exemple:

Test	Résultat
`GenMEg.genMEg(0)`	`|`
`GenMEg.genMEg(1)`	`|`
`GenMEg.genMEg(2)`	`01|10|`

Réponse : (régime de pénalités : 0 %)

Réinitialiser la réponse

```
1  public class GenMEg { //TO COMPLETE
2
3      public static void genMEg(int n){
4          if (n % 2 == 1 || n == 0){
5               System.out.print("|");
6          } else {
7              for (int i = 0; i < (int) Math.pow(2,n) ; i++){
8             String nbBinaire = Integer.toString(i,2);
9             if (nbBinaire.length() != n){
10                nbBinaire = padLeftZeros(nbBinaire, n);
11            }
12             if (countOf(nbBinaire, '0') == countOf(nbBinaire, '1')){
13                 System.out.print(nbBinaire + "|");
14             }
15
16         }
17          }
18     }
19     private static void genMEg(String pref, int  nb0, int  nb1, int k){
20     }
21
22     private static int countOf(String entree, char caractere){
23         int count = 0;
24
```

	Test	Résultat attendu	Résultat obtenu	

	Test	Résultat attendu	Résultat obtenu	
✔	`GenMEg.genMEg(0)`	`|`	`|`	✔
✔	`GenMEg.genMEg(1)`	`|`	`|`	✔
✔	`GenMEg.genMEg(2)`	`01|10|`	`01|10|`	✔
✔	`GenMEg.genMEg(4)`	`0011|0101|0110|1001|1010|1100|`	`0011|0101|0110|1001|1010|1100|`	✔

Tous les tests ont été réussis ! ✔

Solution de l'auteur de la question (Java):

```java
public class GenMEg {
    private static final String SEP = "|";

    public static void genMEg(int n){
        if ( n % 2 == 1){
            System.out.print(SEP);
        }
        else{
            genMEg("",n/2,n/2,n);
        }
    }
    private static void genMEg(String pref, int  nb0, int  nb1, int k){
        if (k == 0) {
            System.out.print(pref + SEP);
            return;
        }
        if ( 0 < nb0 ) genMEg(pref+"0", nb0-1, nb1,k-1);
        if ( 0 < nb1 ) genMEg(pref+"1",  nb0, nb1-1,k-1);
    }

}
```

Correct

Note pour cet envoi : 1,00/1,00.

Question **15**

Correct

Note de 1,00 sur 1,00

On travaille sur les mots **miroirs** écrit sur l'alphabet {a,b,c}, un mot u est **miroir** si il est de longueur paire et si l'une des 2 conditions suivantes est vérifiée :

1. u est le mot vide
2. la première lettre de u est égale à la dernière de u et u privé de sa première lettre et de sa dernière lettre est un mot miroir.

Par exemple "abccba" et "bbaabb" sont miroirs, mais "abcbca" et "ab" ne sont pas miroirs.

Compléter la classe ***GenMirror*** pour que la méthode

```
public static void genMirror(int n)
```

affiche tous les mots miroirs de longueur n sur l'alphabet {a,b,c}, à partir d'appel(s) de la méthode suivante

```
private static void genMirror(String u, int k)
```

où :

- u est un mot miroir
- k est la différence entre n et la longueur de u.

Les mots affichés sont séparés par un espace " ".
Par exemple:

Test	Résultat
`GenMirror.genMirror(0)`	
`GenMirror.genMirror(1)`	
`GenMirror.genMirror(2)`	`aa bb cc`
`GenMirror.genMirror(4)`	`aaaa baab caac abba bbbb cbbc acca bccb cccc`

Réponse : (régime de pénalités : 0 %)

Réinitialiser la réponse

```
1  public class GenMirror {//TO COMPLETE
2      private static final String[] ALPHABET = {"a","b","c"};
3      private static final String SEP = " ";
4
5
6      public static void genMirror(int n){
7          if (!(n % 2 == 1 || n == 0)){
8              genMirror("", n);
9          }
10     }
11
12     private static void genMirror(String u, int k){
13         if (k == 0){
14             System.out.print(u + SEP);
15         }else {
16             for(int i = 0 ; i < ALPHABET.length ; i++){
17                 genMirror(ALPHABET[i] + u + ALPHABET[i], k-2);
18             }
19         }
20     }
21 }
22
23
```

	Test	Résultat attendu	Résultat obtenu	
✔	`GenMirror.genMirror(0)`			✔
✔	`GenMirror.genMirror(1)`			✔
✔	`GenMirror.genMirror(2)`	`aa bb cc`	`aa bb cc`	✔
✔	`GenMirror.genMirror(4)`	`aaaa baab caac abba bbbb cbbc acca bccb cccc`	`aaaa baab caac abba bbbb cbbc acca bccb cccc`	✔
✔	`GenMirror.genMirror(5)`			✔

Tous les tests ont été réussis ! ✔

Solution de l'auteur de la question (Java):

```java
public class GenMirror {
    private static final String[] ALPHABET = {"a","b","c"};
    private static final String SEP = " ";


    public static void genMirror(int n){
        if(n%2 == 1) {
            System.out.println("");
            return ;
        }
        genMirror("",n);
    }

    private static void genMirror(String m, int k){
        if (k == 0) {
            System.out.print(m + SEP);
            return;
        }
        for (String lettre  : ALPHABET) {
            genMirror(lettre+m+lettre,k-2);
        }
    }
}
```

Correct

Note pour cet envoi : 1,00/1,00.

Question **16**

Incorrect

Note de 0,00 sur 1,00

Compléter la classe ***GenMBP*** pour que la méthode

```
public static void genMBP(int n)
```

affiche tous les mots bien parenthésés de longueur 2*n, à partir d'appel(s) de la méthode suivante

```
private static void genMBP(String pref, int aFermer, int reste)
```

où :

- pref est un préfixe d'un mot bien parenthésé
- aFermer est la différence entre le nombre de '(' et le nombre de ')' de pref
- reste est la différence entre 2*n et la longueur de pref.

Les mots affichés sont séparés par un espace.
Par exemple:

Test	Résultat
`GenMBP.genMBP(0)`	
`GenMBP.genMBP(1)`	`()`
`GenMBP.genMBP(2)`	`()() (())`

Réponse : (régime de pénalités : 0 %)

Réinitialiser la réponse

```
1  public class GenMBP {        //TO COMPLETE
2      private static final String OPENING = "(";
3      private static final String CLOSING = ")";
4      private static final String SEP = " ";
5
6      public static void genMBP(int n){
7          if (n != 0) {
8
9          }
10     }
11
12     private static void genMBP(String pref, int aFermer, int reste){
13         //TO COMPLETE
14     }
15 }
```

	Test	Résultat attendu	
✔	`GenMBP.genMBP(0)`		✔
✖	`GenMBP.genMBP(1)`	`()`	✖

	Test	Résultat attendu	
✖	`GenMBP.genMBP(2)`	`()() (())`	✖
✖	`GenMBP.genMBP(3)`	`()()() ()(()) (())() (()()) ((()))`	✖

Votre code doit réussir tous les tests pour gagner des points. Recommencer.

Solution de l'auteur de la question (Java):

```java
public class GenMBP {
    private static final String OPENING = "(";
    private static final String CLOSING = ")";
    private static final String SEP = " ";

    public static void genMBP(int n){
        genMBP("",0,2*n);
    }

    private static void genMBP(String pref, int aFermer, int reste){
        if (reste == 0){
            System.out.print(pref + SEP);
            return;
        }
    if (0 < aFermer) genMBP(pref+CLOSING,aFermer-1,reste-1);
    if (aFermer+1 < reste) genMBP(pref+OPENING,aFermer+1,reste-1);
    }

}
```

Incorrect

Note pour cet envoi : 0,00/1,00.

Question 17

Incorrect

Note de 0,00 sur 1,00

◄ QCM_entrainement_def_inductives

Aller à...

QCM_entrainement_def_inductives_fcts_prog_Java ►

Écrire une méthode Java de signature

qui retourne *true* si on peut écrire l'entier s comme somme des entiers contenus dans le tableau tab. Ces valeurs peuvent être prises au plus une fois.

Par exemple:

Test	Résultat
`sum(new int[]{5,11,3},8)`	`true`
`sum(new int[]{5,11,-3},14)`	`false`

Réponse : (régime de pénalités : 10, 20, ... %)

Réinitialiser la réponse

```
1  ▾    public static boolean sum(int[] tab, int s) {
2           return sum(tab, tab.length, s);
3       }
4
5  ▾    private static boolean sum(int[] tab, int i, int s) {//to complete
6           return true;
7       }
```

	Test	Résultat attendu	Résultat obtenu	
✔	`sum(new int[]{5,11,3},8)`	`true`	`true`	✔
✔	`sum(new int[]{5,11,-3},13)`	`true`	`true`	✔
✖	`sum(new int[]{5,11,-3},14)`	`false`	`true`	✖
✔	`sum(new int[]{5,11,-3},0)`	`true`	`true`	✔

Votre code doit réussir tous les tests pour gagner des points. Recommencer.

Montrer les différences

Solution de l'auteur de la question (Java):

```
1  ▾    public static boolean sum(int[] tab, int s) {
```

```
        return sum(tab, tab.length-1, s);
    }

    private static boolean sum(int[] tab, int n, int s) {
        if (s == 0) return true;
        if (n<0) return false;
        return sum(tab,n-1,s) || sum(tab,n-1,s-tab[n]);
    }
```

Incorrect

Note pour cet envoi : 0,00/1,00.