Programmation
Licence 1 UPEC 2022/2023
TM8 : Tester ses fonctions dans un programme principal

Testez vos fonctions du **TM7** en utilisant les programmes suivants, et vérifiez que vous obtenez les résultats correspondants. Vous devez recopier les programmes ci-dessous (un programme par question), puis y intégrer votre fonction, puis le lancer en exécution et vérifier que son résultat est identique à l'affichage correspondant.

Exercice 1 : Fonctions simples Voici le programme `main` pour tester votre fonction `somme` :

```
void main(){
        test_somme();
}
void test_somme() {
        int[] tab = new int[0];
        println(somme(tab));
        tab = new int[] {34};
        println(somme(tab));
        tab = new int[] {5, -12, 32, 21, 25};
        println(somme(tab));
        tab = new int[] {100, -21, 20, 23, 1, -2, 0, 7, 0, 13};
        println(somme(tab));
        tab = new int[] {-5, -2, -32, -14, -54, -34};
        println(somme(tab));
        tab = new int[] {-45, -32, -51, -4, -13, -55, -45, -100, -12, -49};
        println(somme(tab));
        tab = new int[] {
                12, 12, 12, 12, 12, 12, 12, 12, 12, 12, 12, 12, 12, 12, 12, 12,
                12, 12, 12, 12, 12, 12, 12, 12, 12, 12, 12, 12, 12, 12, 12, 12
        };
        println(somme(tab));
        tab = new int[] {
                4, 3, 5, 7, 1, 2, -3, 2, 7, 11, 4, 5, 3, 9, 12, 76, 31, 31, 223,
                1, 5, 0, -23, -23, -100
        };
        println(somme(tab));
        tab = new int[] {2, 3};
        println(somme(tab));
        tab = new int[10];
        println(somme(tab));
}
```

Le résultat devrait être l'affichage suivant :

```
0
34
71
141
-141
-406
384
293
5
0
```

Voici le programme main pour la fonction `appartient` :

```
void main(){
        test_appartient();
}
void test_appartient() {
        int[] tab = new int[0];
        println(appartient(tab, 5));
        tab = new int[] {34};
        println(appartient(tab, 5));
        tab = new int[] {5, -12, 32, 21, 25};
        println(appartient(tab, 25));
        tab = new int[] {100, -21, 20, 23, 1, -2, 0, 7, 0, 13};
        println(appartient(tab, 100));
        tab = new int[] {-14, -2, -32, -14, -54, -34};
        println(appartient(tab, -14));
        tab = new int[] {-45, -32, -51, -4, -13, -55, -45, -100, -12, -49};
        println(appartient(tab, 45));
        tab = new int[] {
                12, 12, 12, 12, 12, 12, 12, 12, 12, 12, 12, 12, 12, 12, 12, 12,
                12, 12, 12, 12, 12, 12, 12, 12, 12, 12, 12, 12, 12, 12, 12, 12
        };
        println(appartient(tab, 11));
        tab = new int[] {
                4, 3, 5, 7, 1, 2, -3, 2, 7, 11, 4, 5, 3, 9, 12, 76, 31, 31, 223,
                1, 5, 0, -23, -23, -100
        };
        println(appartient(tab, 6));
        tab = new int[] {2, 3};
        println(appartient(tab, -3));
        tab = new int[10];
        println(appartient(tab, 0));
}
```

Le résultat devrait être le suivant :

```
false
false
true
true
true
false
false
false
false
true
```

Voici maintenant le programme pour les tests pour la fonction `paire` :

```
void main(){
        test_paire();
}
void test_paire() {
        int[] tab = new int[0];
        println(paire(tab));
        tab = new int[] {34};
        println(paire(tab));
        tab = new int[] {5, -12, 5, 21, 25};
        println(paire(tab));
        tab = new int[] {100, -21, 21, 23, 1, -2, 0, 7, 0, 13};
```

```
        println(paire(tab));
        tab = new int[] {-5, -32, -32, -14, -54, -34};
        println(paire(tab));
        tab = new int[] {-32, -32, -51, -4, -13, -55, -45, -100, -12, -49};
        println(paire(tab));
        tab = new int[] {
                12, 12, 12, 12, 12, 12, 12, 12, 12, 12, 12, 12, 12, 12, 12, 12,
                12, 12, 12, 12, 12, 12, 12, 12, 12, 12, 12, 12, 12, 12, 12, 12
        };
        println(paire(tab));
        tab = new int[] {
                4, 3, 5, 7, 1, 2, -3, 2, 7, 11, 4, 5, 3, 9, 12, 76, 31, 31, 223,
                1, 5, 0, -23, -23
        };
        println(paire(tab));
        tab = new int[] {2, 3};
        println(paire(tab));
        tab = new int[10];
        println(paire(tab));
}
```

Et voici le résultat attendu :

```
false
false
false
false
true
true
true
true
false
true
```

Et enfin le programme test pour la fonction `brelan` :

```
void main(){
        test_brelan();
}
void test_brelan() {
        int[] tab = new int[0];
        println(brelan(tab));
        tab = new int[] {34};
        println(brelan(tab));
        tab = new int[] {5, -12, 5, 5, 21};
        println(brelan(tab));
        tab = new int[] {100, -21, 21, 21, 1, -21, 0, 21, 0, 13};
        println(brelan(tab));
        tab = new int[] {-5, -32, -32, -32, -54, -34};
        println(brelan(tab));
        tab = new int[] {-32, -32, -32, -4, -13, -55, -45, -100, -12, -49};
        println(brelan(tab));
        tab = new int[] {
                12, 12, 12, 12, 12, 12, 12, 12, 12, 12, 12, 12, 12, 12, 12, 12,
                12, 12, 12, 12, 12, 12, 12, 12, 12, 12, 12, 12, 12, 12, 12, 12
        };
        println(brelan(tab));
        tab = new int[] {
                4, 3, 5, 7, 1, 2, -3, 2, 7, 11, 4, 5, 3, 9, 12, 76, 31, 31, 223,
                1, 5, -23, -23, -23
```

```
        };
        println(brelan(tab));
        tab = new int[] {2, 2};
        println(brelan(tab));
        tab = new int[10];
        println(brelan(tab));
}
```

Et le résultat attendu :

```
false
false
false
false
true
true
true
true
false
true
```

Exercice 2 : Operations Mêmes questions que pour les questions 2.1-2.2 du **TM7** et les programmes suivants :

Programme pour tester `sommeMultiple` (bien remarquer la façon utilisée pour créer un paramètre tableau) :

```
void main(){
        test_sommeMultiple();
}
void test_sommeMultiple() {
        println(sommeMultiple(new int[] {5, -12, 5, 5, 25}, 5));
        println(sommeMultiple(new int[] {100, -21, 21, 21, -21, -2, 0, 7, 0, 13}, 3));
        println(
        sommeMultiple(
        new int[] {
                4, 3, 5, 7, 1, 2, -3, 2, 7, 11, 4, 5, 3, 9, 12, 76, 31, 31, 223, 1, 5,
                -23, -23, -23
        },
        6));
        println(
        sommeMultiple(
        new int[] {
                4, 3, 5, 7, 1, 2, -3, 2, 7, 11, 4, 5, 3, 9, 12, 76, 31, 31, 223, 1, 5,
                -23, -23, -23, 31
        },
        6));
        println(
        sommeMultiple(
        new int[] {
                4, 3, 5, 7, 1, 2, -3, 2, 7, 11, 4, 5, 3, 9, 12, 76, 31, 31, 223, 1, 5,
                -23
        },
        6));
        println(sommeMultiple(new int[0], 5));
        println(sommeMultiple(new int[] {23}, 5));
        println(sommeMultiple(new int[] {5, -12, 5, 5, 25}, -3));
        println(
        sommeMultiple(
```

```
        new int[] {
                4, 3, 5, 7, 1, 2, -3, 2, 7, 11, 4, 5, 3, 9, 12, 76, 31, 31, 223, 1, 5,
                -23, -23, -23
        },
        -4));
        println(sommeMultiple(new int[13], 3));
        println(
        sommeMultiple(
        new int[] {
                4, 3, 5, 7, 1, 2, -3, 2, 7, 11, 4, 5, 3, 9, 12, 76, 31, 31, 223, 1, 5,
                -23, -23, -23
        },
        0));
        println(
        sommeMultiple(
        new int[] {
                4, 3, 5, 7, 1, 2, -3, 2, 7, 11, 4, 5, 3, 9, 12, 76, 31, 31, 223, 1, 5,
                -23, -23, -23
        },
        100));
}
```

Résultats attendus :

```
5
134
227
258
227
0
23
10
51
0
0
4
```

Programme principal pour `tabMultiple` :

```
void main(){
        test_tabMultiple();
}
void test_tabMultiple() {
        affiche(tabMultiple(new int[] {5, -12, 5, 5, 25}, 5));
        affiche(tabMultiple(new int[] {100, -21, 21, 21, -21, -2, 0, 7, 0, 13}, 3));
        affiche(
        tabMultiple(
        new int[] {
                4, 3, 5, 7, 1, 2, -3, 2, 7, 11, 4, 5, 3, 9, 12, 76, 31, 31, 223, 1, 5,
                -23, -23, -23
        },
        6));
        affiche(
        tabMultiple(
        new int[] {
                4, 3, 5, 7, 1, 2, -3, 2, 7, 11, 4, 5, 3, 9, 12, 76, 31, 31, 223, 1, 5,
                -23, -23, -23, 31
        },
        6));
```

```
        affiche(
        tabMultiple(
        new int[] {
                4, 3, 5, 7, 1, 2, -3, 2, 7, 11, 4, 5, 3, 9, 12, 76, 31, 31, 223, 1, 5,
                -23
        },
        6));
        affiche(tabMultiple(new int[0], 5));
        affiche(tabMultiple(new int[] {23}, 5));
        affiche(tabMultiple(new int[] {5, -12, 5, 5, 25}, -3));
        affiche(
        tabMultiple(
        new int[] {
                4, 3, 5, 7, 1, 2, -3, 2, 7, 11, 4, 5, 3, 9, 12, 76, 31, 31, 223, 1, 5,
                -23, -23, -23
        },
        -4));
        affiche(tabMultiple(new int[13], 3));
        affiche(
        tabMultiple(
        new int[] {
                4, 3, 5, 7, 1, 2, -3, 2, 7, 11, 4, 5, 3, 9, 12, 76, 31, 31, 223, 1, 5,
                -23, -23, -23
        },
        0));
        affiche(
        tabMultiple(
        new int[] {
                4, 3, 5, 7, 1, 2, -3, 2, 7, 11, 4, 5, 3, 9, 12, 76, 31, 31, 223, 1, 5,
                -23, -23, -23
        },
        100));
}
void affiche(int[] tab) {
        print("[");
        for (int i = 0; i < tab.length - 1; i++) {
                print(tab[i] + ",");
        }
        if (tab.length > 0) {
                print(tab[tab.length - 1]);
        }
        println("]");
}
```

Et le résultat attendu :

```
[5]
[100,21,0,13]
[4,-3,3,223]
[4,-3,3,223,31]
[4,-3,3,223]
[]
[23]
[5,5]
[4,1,7,3,31,5]
[0,0,0,0,0]
[]
[4]
```

Question 2 supplémentaire : Pourquoi on peut pas faire des exercices similaires pour les fonctions `tabHasard` et `sommeHasard` (c'est à dire, questions 2.3 et 2.4 du TM7) ?

Exercice 3 : Modification d'un tableau Et maintenant même activité pour les questions 3.1–3.5.

Le programme main pour la fonction `augmente2` (récupérer la fonction `affiche` de l'exo précédent !) :

```
void main(){
        test_augmente2();
}
void test_augmente2() {
        int[] tab;
        augmente2(tab = new int[] {5, -12, 5, 5, 25});
        affiche(tab);
        augmente2(tab = new int[] {100, -21, 21, 21, -21, -2, 0, 7, 0, 13});
        affiche(tab);
        augmente2( tab = new int[] {
                4, 3, 5, 7, 1, 2, -3, 2, 7, 11, 4, 5, 3, 9, 12, 76,
                31, 31, 223, 1, 5, -23, -23, -23
        });
        affiche(tab);
        augmente2( tab = new int[] {
                4, 3, 5, 7, 1, 2, -3, 2, 7, 11, 4, 5, 3, 9, 12, 76,
                31, 31, 223, 1, 5, -23, -23, -23, 31
        });
        affiche(tab);
        augmente2( tab = new int[] {
                4, 3, 5, 7, 1, 2, -3, 2, 7, 11, 4, 5, 3, 9, 12, 76,
                31, 31, 223, 1, 5, -23
        });
        affiche(tab);
        augmente2(tab = new int[0]);
        affiche(tab);
        augmente2(tab = new int[] {23});
        affiche(tab);
        augmente2(tab = new int[] {5, -12, 5, 5, 25});
        affiche(tab);
        augmente2( tab = new int[] {
                4, 3, 5, 7, 1, 2, -3, 2, 7, 11, 4, 5, 3, 9, 12, 76, 31, 31,
                223, 1, 5, -23, -23, -23
        });
        affiche(tab);
        augmente2(tab = new int[13]);
        affiche(tab);
        augmente2( tab = new int[] {
                4, 3, 5, 7, 1, 2, -3, 2, 7, 11, 4, 5, 3, 9, 12, 76, 31, 31,
                223, 1, 5, -23, -23, -23
        });
        affiche(tab);
        augmente2( tab = new int[] {
                4, 3, 5, 7, 1, 2, -3, 2, 7, 11, 4, 5, 3, 9, 12, 76, 31, 31,
                223, 1, 5, -23, -23, -23
        });
}
```

Et l'affichage attendu :

```
[7,-10,7,7,27]
[102,-19,23,23,-19,0,2,9,2,15]
[6,5,7,9,3,4,-1,4,9,13,6,7,5,11,14,78,33,33,225,3,7,-21,-21,-21]
[6,5,7,9,3,4,-1,4,9,13,6,7,5,11,14,78,33,33,225,3,7,-21,-21,-21,33]
[6,5,7,9,3,4,-1,4,9,13,6,7,5,11,14,78,33,33,225,3,7,-21]
```

```
[]
[25]
[7,-10,7,7,27]
[6,5,7,9,3,4,-1,4,9,13,6,7,5,11,14,78,33,33,225,3,7,-21,-21,-21]
[2,2,2,2,2,2,2,2,2,2,2,2,2]
[6,5,7,9,3,4,-1,4,9,13,6,7,5,11,14,78,33,33,225,3,7,-21,-21,-21]
```

Le programme main pour la fonction `augmente` :

```
void main(){
        test_augmente();
}
void test_augmente() {
        int[] tab;
        augmente(tab = new int[] {5, -12, 5, 5, 25}, 4);
        affiche(tab);
        augmente(tab = new int[] {100, -21, 21, 21, -21, -2, 0, 7, 0, 13}, -10);
        affiche(tab);
        augmente( tab = new int[] {
                4, 3, 5, 7, 1, 2, -3, 2, 7, 11, 4, 5, 3, 9, 12, 76, 31, 31, 223, 1, 5,
                -23, -23, -23
        },
        100);
        affiche(tab);
        augmente( tab = new int[] {
                4, 3, 5, 7, 1, 2, -3, 2, 7, 11, 4, 5, 3, 9, 12, 76, 31, 31, 223, 1, 5,
                -23, -23, -23, 31
        },
        2);
        affiche(tab);
        augmente( tab = new int[] {
                4, 3, 5, 7, 1, 2, -3, 2, 7, 11, 4, 5, 3, 9, 12, 76, 31, 31, 223, 1, 5,
                -23
        },
        15);
        affiche(tab);
        augmente(tab = new int[0], 4);
        affiche(tab);
        augmente(tab = new int[] {23}, 23);
        affiche(tab);
        augmente(tab = new int[] {5, -12, 5, 5, 25}, -23);
        affiche(tab);
        augmente( tab = new int[] {
                4, 3, 5, 7, 1, 2, -3, 2, 7, 11, 4, 5, 3, 9, 12, 76, 31, 31, 223, 1, 5,
                -23, -23, -23
        },
        -23);
        affiche(tab);
        augmente(tab = new int[13], -10);
        affiche(tab);
        augmente( tab = new int[] {
                4, 3, 5, 7, 1, 2, -3, 2, 7, 11, 4, 5, 3, 9, 12, 76, 31, 31, 223, 1, 5,
                -23, -23, -23
        },
        -10);
        affiche(tab);
        augmente( tab = new int[] {
                4, 3, 5, 7, 1, 2, -3, 2, 7, 11, 4, 5, 3, 9, 12, 76, 31, 31, 223, 1, 5,
                -23, -23, -23
```

```
        },
        -3);
}
```

Et l'affichage attendu :

```
[9,-8,9,9,29]
[90,-31,11,11,-31,-12,-10,-3,-10,3]
[104,103,105,107,101,102,97,102,107,111,104,105,103,109,112,176,131,131,323,101,105,77,77,77]
[6,5,7,9,3,4,-1,4,9,13,6,7,5,11,14,78,33,33,225,3,7,-21,-21,-21,33]
[19,18,20,22,16,17,12,17,22,26,19,20,18,24,27,91,46,46,238,16,20,-8]
[]
[46]
[-18,-35,-18,-18,2]
[-19,-20,-18,-16,-22,-21,-26,-21,-16,-12,-19,-18,-20,-14,-11,53,8,8,200,-22,-18,-46,-46,-46]
[-10,-10,-10,-10,-10,-10,-10,-10,-10,-10,-10,-10,-10]
[-6,-7,-5,-3,-9,-8,-13,-8,-3,1,-6,-5,-7,-1,2,66,21,21,213,-9,-5,-33,-33,-33]
```

Le programme main pour la fonction `echange` :

```
void main(){
        test_echange();
}
void test_echange() {
        int[] tab;
        echange(tab = new int[] {5, -12, 5, 5, 25}, 4, 2);
        affiche(tab);
        echange(tab = new int[] {100, -21, 21, 21, -21, -2, 0, 7, 0, 13}, 0, 3);
        affiche(tab);
        echange( tab = new int[] {
                4, 3, 5, 7, 1, 2, -3, 2, 7, 11, 4, 5, 3, 9, 12, 76, 31, 31, 223, 1, 5,
                -23, -23, -23
        },
        1,
        1);
        affiche(tab);
        echange( tab = new int[] {
                4, 3, 5, 7, 1, 2, -3, 2, 7, 11, 4, 5, 3, 9, 12, 76, 31, 31, 223, 1, 5,
                -23, -23, -23, 31
        },
        2,
        16);
        affiche(tab);
        echange( tab = new int[] {
                4, 3, 5, 7, 1, 2, -3, 2, 7, 11, 4, 5, 3, 9, 12, 76, 31, 31, 223, 1, 5, -23
        },
        15,
        14);
        affiche(tab);
        echange(tab = new int[0], 4, 5);
        affiche(tab);
        echange(tab = new int[] {23}, 1, 0);
        affiche(tab);
        echange(tab = new int[] {5, -12, 5, 5, 25}, 0, 5);
        affiche(tab);
        echange( tab = new int[] {
                4, 3, 5, 7, 1, 2, -3, 2, 7, 11, 4, 5, 3, 9, 12, 76, 31, 31, 223, 1, 5,
                -23, -23, -23
        },
        -2,
        3);
```

```
        affiche(tab);
        echange(tab = new int[13], 1, 0);
        affiche(tab);
        echange( tab = new int[] {
                4, 3, 5, 7, 1, 2, -3, 2, 7, 11, 4, 5, 3, 9, 12, 76, 31, 31, 223, 1, 5,
                -23, -23, -23
        },
        0,
        345);
        affiche(tab);
        echange( tab = new int[] {
                4, 3, 5, 7, 1, 2, -3, 2, 7, 11, 4, 5, 3, 9, 12, 76, 31, 31, 223, 1, 5,
                -23, -23, -23
        },
        -3,
        4);
}
```

Et le résultat attendu :

```
[5,-12,25,5,5]
[21,-21,21,100,-21,-2,0,7,0,13]
[4,3,5,7,1,2,-3,2,7,11,4,5,3,9,12,76,31,31,223,1,5,-23,-23,-23]
[4,3,31,7,1,2,-3,2,7,11,4,5,3,9,12,76,5,31,223,1,5,-23,-23,-23,31]
[4,3,5,7,1,2,-3,2,7,11,4,5,3,9,76,12,31,31,223,1,5,-23]
[]
[23]
[5,-12,5,5,25]
[4,3,5,7,1,2,-3,2,7,11,4,5,3,9,12,76,31,31,223,1,5,-23,-23,-23]
[0,0,0,0,0,0,0,0,0,0,0,0,0]
[4,3,5,7,1,2,-3,2,7,11,4,5,3,9,12,76,31,31,223,1,5,-23,-23,-23]
```

Le programme main pour la fonction `rotation` :

```
void main(){
        test_rotation();
}
void test_rotation() {
        int[] tab;
        rotation(tab = new int[] {5, -12, 5, 5, 25});
        affiche(tab);
        rotation(tab = new int[] {100, -21, 21, 21, -21, -2, 0, 7, 0, 13});
        affiche(tab);
        rotation( tab = new int[] {
                4, 3, 5, 7, 1, 2, -3, 2, 7, 11, 4, 5, 3, 9, 12, 76, 31, 31, 223, 1, 5,
                -23, -23, -23
        });
        affiche(tab);
        rotation( tab = new int[] {
                4, 3, 5, 7, 1, 2, -3, 2, 7, 11, 4, 5, 3, 9, 12, 76, 31, 31, 223, 1, 5,
                -23, -23, -23, 31
        });
        affiche(tab);
        rotation( tab = new int[] {
                4, 3, 5, 7, 1, 2, -3, 2, 7, 11, 4, 5, 3, 9, 12, 76, 31, 31, 223, 1, 5,
                -23
        });
        affiche(tab);
        rotation(tab = new int[0]);
```

```
        affiche(tab);
        rotation(tab = new int[] {23});
        affiche(tab);
        rotation(tab = new int[] {5, -12, 5, 25, 5});
        affiche(tab);
        rotation( tab = new int[] {
                4, 3, 5, 7, 1, 2, -3, 2, 7, 11, 4, 5, 3, 9, 12, 76, 31, 31, 223, 1, 5,
                -23, -23, -23
        });
        affiche(tab);
        rotation(tab = new int[13]);
        affiche(tab);
        rotation( tab = new int[] {
                4, 3, 5, 7, 1, 2, -3, 2, 7, 11, 4, 5, 3, 9, 12, 76, 31, 31, 223, 1, 5,
                -23, -23, -23
        });
        affiche(tab);
        rotation( tab = new int[] {
                4, 3, 5, 7, 1, 2, -3, 2, 7, 11, 4, 5, 3, 9, 12, 76, 31, 31, 223, 1, 5,
                -23, -23, -23
        });
}
void affiche(int[] tab) {
        print("[");
        for (int i = 0; i < tab.length - 1; i++) {
                print(tab[i] + ",");
        }
        if (tab.length > 0) {
                print(tab[tab.length - 1]);
        }
        println("]");
}
```

Et le résultat attendu :

```
[25,5,-12,5,5]
[13,100,-21,21,21,-21,-2,0,7,0]
[-23,4,3,5,7,1,2,-3,2,7,11,4,5,3,9,12,76,31,31,223,1,5,-23,-23]
[31,4,3,5,7,1,2,-3,2,7,11,4,5,3,9,12,76,31,31,223,1,5,-23,-23,-23]
[-23,4,3,5,7,1,2,-3,2,7,11,4,5,3,9,12,76,31,31,223,1,5]
[]
[23]
[5,5,-12,5,25]
[-23,4,3,5,7,1,2,-3,2,7,11,4,5,3,9,12,76,31,31,223,1,5,-23,-23]
[0,0,0,0,0,0,0,0,0,0,0,0,0]
[-23,4,3,5,7,1,2,-3,2,7,11,4,5,3,9,12,76,31,31,223,1,5,-23,-23]
```

Le programme main pour la fonction `remplissage` :

```
void main(){
        test_remplissage();
}
void test_remplissage() {
        int[] tab;
        remplissage(new int[] {5, -12, 5, 5, 25}, new int[] {12, -3, 41}, tab = new int[10]);
        affiche(tab);
        remplissage(new int[] {5, -12, 5, 5, 25}, new int[] {12, 33, 43}, tab = new int[4]);
        affiche(tab);
        remplissage(new int[] {5, -12, 5, 5, 25}, new int[] {2, 34, 42, 32}, tab = new int[8]
```

```
        affiche(tab);
        remplissage(new int[] {5, -12, 5, 5, 25}, new int[] {2, 3, 4}, tab = new int[0]);
        affiche(tab);
        remplissage(new int[0], new int[] {2, 3, 4}, tab = new int[10]);
        affiche(tab);
        remplissage(new int[] {5, -12, 5, 5, 25}, new int[0], tab = new int[6]);
        affiche(tab);
        remplissage(new int[] {5, -12, 5, 5, 25}, new int[0], tab = new int[3]);
        affiche(tab);
        remplissage(new int[] {5, -12}, new int[] {2, 3, 4}, tab = new int[6]);
        affiche(tab);
        remplissage(new int[0], new int[0], tab = new int[2]);
        affiche(tab);
        remplissage(new int[] {5, -12, 5, 5, 25}, new int[0], tab = new int[1]);
        affiche(tab);
}
```

Et le résultat attendu :

```
[5,-12,5,5,25,12,-3,41,0,0]
[5,-12,5,5]
[5,-12,5,5,25,2,34,42]
[]
[2,3,4,0,0,0,0,0,0,0]
[5,-12,5,5,25,0]
[5,-12,5]
[5,-12,2,3,4,0]
[0,0]
[5]
```

Exercice 4 : Les permutations

(4.1) Écrire une fonction `contientApres`, qui prend en paramètre un tableau d'entiers `tab`, un entier `n`, et un entier `i` et renvoie `true` si le tableau contient `n` dans une case dont l'indice est plus grand ou égale à `i`. Le programme main pour la fonction `contientApres` :

```
void main(){
        test_contientApres();
}
void test_contientApres() {
        println(contientApres(new int[] {100, -21, 14, 5, 8, -2, 0, 7, 5, 13}, 3, 5));
        println(contientApres(new int[] {3, -21, 14, 5, 8, -2, 0, 7, 5, 13}, 3, 5));
        println(contientApres(new int[] {3, -21, 14, 5, 8, -2, 0, 7, 5, 13}, 7, 5));
        println(
        contientApres(
        new int[] {
                4, 3, 5, 7, 1, 2, -3, 2, 7, 11, 4, 5, 3, 9, 12, 76, 31, 31, 223, 1, 5,
                -23, 7, 13
        },
        31,
        6));
        println(contientApres(new int[] {3, -21, 14, 5, 8, -2, 0, 7, 5, 13}, 3, 45));
        println(contientApres(new int[] {3, -21, 14, 5, 8, -2, 0, 7, 5, 13}, 3, -1));
        println(contientApres(new int[0], 3, 1));
        println(contientApres(new int[] {23}, 3, 0));
        println(contientApres(new int[] {23}, 23, 0));
        println(contientApres(new int[] {3, -21, 14, 5, 8, -2, 0, 7, 5, 13}, 13, 9));
}
```

Le résultat attendu :

```
false
false
true
true
false
true
false
false
true
true
```

(4.2) Écrire une fonction `doublons`, qui prend en paramètre un tableau d'entier `tab` et renvoie `true` si le tableau contient un nombre plus qu'une fois. Cette fonction utilisera la fonction précédente. Le programme main pour la fonction `doublons` :

```
void main(){
        test_doublons();
}
void test_doublons() {
        println(doublons(new int[] {100, -21, 14, 5, 8, -2, 0, 7, 5, 13}));
        println(doublons(new int[] {3, -21, 14, 5, 8, -2, 0, 7, 5, 13}));
        println(doublons(new int[] {3, -21, 14, 5, 8, -2, 21, 7, -5, 13}));
        println(
        doublons(
        new int[] {
                4, 3, 5, 7, 1, 2, -3, 2, 7, 11, 4, 5, 3, 9, 12, 76, 31, 31, 223, 1, 5,
                -23, 7, 13
        }));
        println(doublons(new int[] {3, -21, 14, 5, 8, -2, 0, 7, -2, 13}));
        println(doublons(new int[] {3, -21, 13, 5, 8, -2, 0, 7, 5, 13, 13}));
        println(doublons(new int[0]));
        println(doublons(new int[] {23}));
        println(doublons(new int[] {23, 23}));
        println(doublons(new int[] {13, -2, 14, 5, 8, -2, 0, 7, 5, 13}));
}
```

Et le résultat attendu :

```
true
true
false
true
true
true
false
false
true
true
```

(4.3) Écrire une fonction `permutation`, qui prend en paramètre deux tableaux d'entiers et qui renvoie `true` si les deux tableaux ne contiennent pas de doublons et s'ils sont l'un une permutation de l'autre. On utilisera les deux fonctions précédentes. Le programme main pour la fonction `permutation` :

```
void main(){
        test_permutation();
}
void test_permutation() {
        println(
```

```
        permutation(
        new int[] {100, -21, 14, 5, 8, -2, 0, 7, 5, 13},
        new int[] {21, 100, 5, 8, -2, 14, 7, 5, 13}));
        println(
        permutation(
        new int[] {3, -21, 14, 5, 8, -2, 0, 7, 13},
        new int[] {100, -21, 14, 5, 8, -2, 0, 7, 6, 13}));
        println(
        permutation(
        new int[] {3, -21, 14, 5, 8, -2, 21, 7, -5, 13},
        new int[] {13, 14, -21, 8, -2, 5, 21, 7, -5, 3}));
        println(
        permutation(
        new int[] {
                4, 3, 5, 7, 1, 2, -3, 2, 7, 11, 4, 5, 3, 9, 12, 76, 31, 31, 223, 1, 5,
                -23, 7, 13
        },
        new int[] {
                4, 3, 5, 7, 1, 2, -3, 2, 7, 11, 4, 5, 3, 9, 12, 76, 31, 31, 223, 1, 5,
                -23, 7, 13
        }));
        println(permutation(new int[] {3, -21, 14, 5}, new int[] {5, 3, -21, 14}));
        println(permutation(new int[] {3, -21, 13}, new int[] {3, -21, 14, 13}));
        println(permutation(new int[0], new int[0]));
        println(permutation(new int[] {23}, new int[] {23}));
        println(permutation(new int[] {23}, new int[0]));
        println(
        permutation(
        new int[] {13, -2, 14, 5, 8, -2, 0, 7, 5, 13}, new int[] {3, -21, 14, 5}));
}
```

Et le résultat attendu :

```
false
false
true
false
true
false
true
true
false
false
```