Prénom :
Nom :

Pour chaque programme, vous devez indiquer l'affichage produit en répondant sur l'énoncé, à droite du programme. **Soyez très précis dans vos réponses.**

```
3  import java.util.Arrays;
4
5  public class Exo11 {
6
7    public static void main(String[] args) {
8      int[][] x = { { 1, 2, 3 }, { 6 }, { 4, 5 } };
9      System.out.println(x.length);
10     for (int i = 0; i < x.length; i++) {
11       System.out.printf("x[%d] = ", i);
12       System.out.printf("%s%n", Arrays.toString(x[i]));
13     }
14     System.out.println(Arrays.deepToString(x));
15     int k = 0;
16     for (int[] u : x) {
17       System.out.printf("{ ");
18       for (int j : u) {
19         System.out.printf("%d ", k + j);
20         k++;
21       }
22       k = k - u.length + 1;
23       System.out.println("}");
24     }
25     System.out.println(k);
26   }
27
28 }
```

Affichage produit

Programmation (1)

Écrire un programme qui réalise les opérations suivantes :

1. demande à l'utilisateur une taille de matrice (c'est-à-dire un nombre de lignes et un nombre de colonnes) ;
2. crée une matrice de la taille indiquée (sous forme d'un tableau de type `int[][]`) et la remplit de chiffres aléatoires (entiers entre 0 et 9 inclus) ;
3. affichage la matrice **sans** utiliser les méthodes `Arrays.toString` et `Arrays.deepToString` ;
4. détermine et affiche la position du plus grand élément de la matrice : si la valeur en question apparaît plusieurs fois, on retiendra comme position la première ligne et la dernière colonne qui prend cette valeur.

```java
public class Exo21 {

  public static void main(String[] args) {
    StringBuilder bla = new StringBuilder();
    bla.append("D");
    System.out.println(bla);
    String s = bla.toString();
    StringBuilder pouic = bla;
    char[] c = { 'A', 'B', 'C' };
    String t = "abcd";
    for (int i = t.length()-1; i >= 0; i--) {
      if (i % 2 == 0) {
        bla.append(t.charAt(i));
      } else {
        s = s + t.charAt(i);
      }
      if (i < c.length) {
        pouic.append(c[i]);
      }
      System.out.printf("%d %s %s %s%n", i, bla, s, pouic);
    }
  }

}
```

Affichage produit

Programmation (2)

Écrire un programme qui affiche un « sapin de noël » de la forme suivante :

```
    +
   *++
  ++*++
 +******
++++++++*
    H
```

On procédera de la façon suivante :

1. le programme doit demander à l'utilisateur la taille du sapin (un entier `n` positif, 5 dans l'exemple ci-dessus) et une valeur réelle comprise entre 0 et 1, `p` (0, 2 dans l'exemple ci-dessus) ;
2. chaque ligne du sapin doit être construite complètement grâce à un `StringBuilder` puis affichée en entier avec une méthode `println` ou `printf` ;
3. le sapin comprend `n+1` lignes qu'on peut numéroter de 1 à `n+1` :
   — les lignes 1 à `n` contiennent uniquement des espaces et les symboles `+` et `*` ;
   — la ligne numéro `i` contient `2i-1` symboles `+` et `*` et débute par `n-i` espaces ;
   — le choix entre les symboles `+` et `*` se fait aléatoirement avec la probabilité `p` de choisir `*` : on utilisera donc un objet `Random` et sa méthode `nextDouble` qui renvoie un nombre réel de $[0, 1]$ choisi uniformément. Il suffit de comparer la valeur obtenue avec `p` pour choisir le bon symbole ;
   — la dernière ligne du sapin contient la lettre `H` centrée.

ATTENTION : vous ne devez pas afficher les lignes caractère par caractère.

Programmation (3)

On souhaite choisir au hasard un nombre premier arbitrairement grand, en utilisant les `BigInteger`. Pour ce faire on écrit un programme qui réalise les opérations suivantes :

1. demande à l'utilisateur un nombre de chiffres, `p`, strictement supérieur à 21 et un entier `k` indiquant le niveau de certitude souhaité (cf plus bas) ;
2. trouve un nombre premier à `p` chiffres grâce à la méthode ci-dessous :
   (a) construit une chaîne de caractères de longueur `p` contenant des chiffres choisis aléatoirement (entre 0 et 9), et se terminant par un chiffre impair (on utilisera un `StringBuilder`) ;
   (b) construit un `BigInteger` à partir de la chaîne en utilisant le constructeur adapté ;
   (c) teste si le nombre obtenu est premier en utilisant sa méthode `isProbablePrime` avec le paramètre `k` ;
   (d) recommence ces opérations si le nombre n'est pas premier ;
3. affiche le nombre obtenu ainsi que le nombre de tentatives nécessaires à sa construction.