Exercice 1 Il s'agit d'un exercice sur papier.

On vous donne des implémentations de deux méthodes avec des bugs:

1. La première méthode indexOf(int[] a, int n) renvoie l'index de la première occurrence de l'entier n dans le tableau a.
2. La seconde méthode average(int[] a) calcule et renvoie la moyenne de tous les entiers du tableau a.

```
public static int indexOf(int[] a, int n) {
if (a == null) {
  throw new
IllegalArgumentException("Array is null");
 }
int index = -1;
for (int i = 0; i <= a.length; i++) {
   if (a[i] == n) {
     index = i;
    break;
  }
}
return index;
}
```

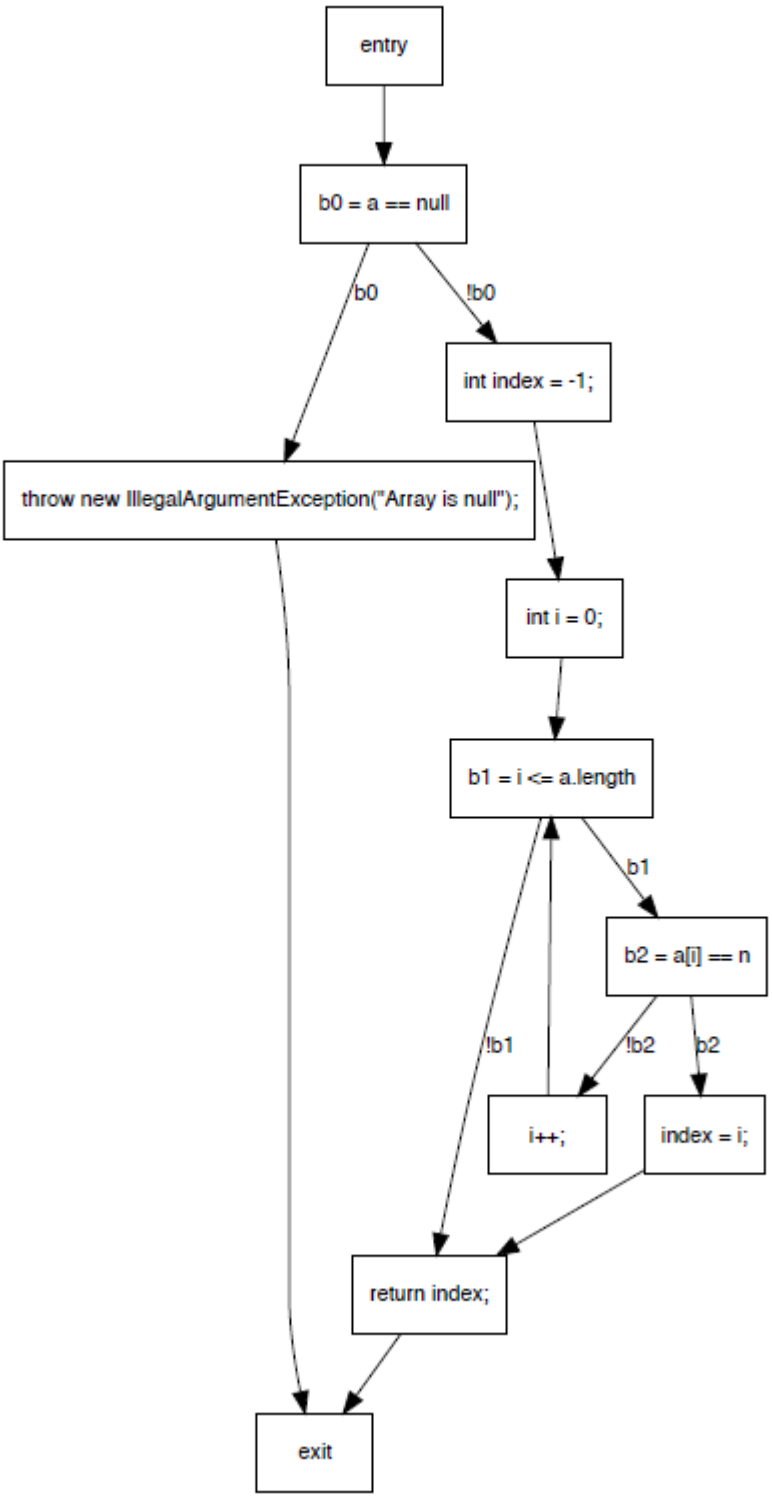
- Dessiner le graphe de flux de contrôle de la méthode **indexOf**.
- Proposer un cas de test qui révèle le bug de la méthode **indexOf**.
- Proposer une suite de tests qui atteint 100 % de couverture des blocs **indexOf** et ne trouve pas le bug.
- Existe-t-il une suite de tests qui atteint une couverture de branche à 100 % pour **indexOf** et manque toujours le bug?

```
public static int average(int[] a) {
 if (a == null) {
  throw new
IllegalArgumentException("Array is null");
}
if (a.length == 0) {
   throw new
IllegalArgumentException("Array is
empty");
}
int sum = 0;
for (int i = 1; i < a.length; i++) {
  sum += a[i];
}
```

- Proposer un cas de test qui révèle le bug dans la méthode **average**
- Proposer une suite de tests qui atteint 100 % de couverture de branche et ne trouve pas le bug

Solutions méthode indexOf :

- Dessiner le graphe de flux de contrôle de la méthode **indexOf**.



- Proposer un cas de test qui révèle le bug de la méthode **indexOf** :

Le bug peut être révélé par le calcul d'un index d'un élément qui n'existe pas dans la tableau.

```
@Test
public void testIndexOfBug() {
  int actual = Array.indexOf(new int[]{0, 1, 2, 3}, 4);
 int expected = -1;
 assertEquals(expected, actual);
}
```

- Proposer une suite de tests qui atteint 100 % de couverture des blocs **indexOf** et ne trouve pas le bug.

```
@Test(expected = IllegalArgumentException.class)
public void testIndexOfNullArgument() {
   Array.indexOf(null, 0);
}
@Test
public void testIndexOfCoverage() {
  int actual = Array.indexOf(new int[]{0, 1, 2, 3}, 2);
  int expected = 2;
assertEquals(expected, actual);
}
```

- Existe-t-il une suite de tests qui atteint une couverture de branche à 100 % pour **indexOf** et manque toujours le bug?

Solutions méthode average :

- Proposer un cas de test qui révèle le bug dans la méthode **average :**
  Le bug dans average peut être révélé si le résultat dépend du premier élément

```
@Test
 public void testAverageBug() {
  int actual = Array.average(new int[]{1});
  int expected = 1;
  assertEquals(expected, actual);
}
```

- Proposer une suite de tests qui atteint 100 % de couverture de branche et ne trouve pas le bug

```
@Test
public void testAverage() {
  int actual = Array.average(new int[]{0, 2, 4, 10});
   int expected = 4;
   assertEquals(expected, actual);
}
@Test(expected = IllegalArgumentException.class)
  public void testAverageNullArgument() {
  Array.average(null);
}
@Test(expected = IllegalArgumentException.class)
  public void testAverageEmptyList() {
  Array.average(new int[]{});
}
```