EP1 Programmiertest 6 WS2021 27. Jänner 2022	**Matrikelnummer**	**Familienname**	**Unterschrift**

Einschränkungen

- Sie dürfen **keine** zusätzlichen eigenen Hilfsmethoden oder globalen Variablen verwenden.
- Die vorgegebenen Methodenköpfe dürfen **nicht** erweitert oder geändert werden.
- Für die Implementierung der rekursiven Methode dürfen **keine** Schleifen verwendet werden.
- Sie dürfen Strings **nicht** per Referenz vergleichen.
- Sie dürfen **nicht** die Methoden `clone` und `System.arraycopy` verwenden.
- Sie dürfen **nur** folgende Methode(n) aus der Klasse `Arrays` verwenden: `deepToString`, `toString`
- Sie dürfen **nur** folgende Methode(n) aus der Klasse `String` verwenden: `charAt`, `equals`, `length`, `substring`, `isEmpty`
- Bis auf die Klasse `Math` dürfen **keine** weiteren Klassen der Java API verwendet werden.

Aufgabenstellung

Deklarieren und initialisieren Sie in `main` die folgende(n) Variable(n):

```
int[] test1 = {2, 0, -1, -1, 6, 1};
int[][] test2 = {{1, 7, 1, 0, 0}, {3, 0}, {-8}, {-1, 3, 0, 1}};
int[][] test3 = {{5, -6}, {0}, {0, 0}};
```

Implementieren Sie folgende Methoden:

- `boolean[][] generate(int[] input)` liefert ein neues Array zurück, in dem alle Zeilen die Mindestlänge 5 haben und Einträge an in `input` spezifizierten Positionen `true` und alle übrigen Einträge `false` sind. Jede Zeile hat maximal einen Eintrag der `true` ist. In `input[i]` steht für die Zeile mit Index `i` der Spaltenindex dieses Eintrags. Ist `input[i]` kleiner 0, hat die Zeile keinen Eintrag `true` und alle 5 Einträge sind `false`. Die Zeilen des zurückgelieferten Arrays können länger als 5 sein und zwar dann, wenn `input[i]` größer als 4 ist. In diesem Fall hat die Zeile die Länge `input[i]+1`, sodass der letzte Eintrag `true` ist.
  Vorbedingung(en): `input != null` .
  Wird die Methode z.B. mit den Parametern `test1` aufgerufen, entsteht folgendes Array:

false	false	true	false	false		
true	false	false	false	false		
false	false	false	false	false		
false	false	false	false	false		
false	false	false	false	false	false	true
false	true	false	false	false		

- `void shift(int[][] input)` ändert das angegebene Array `input` so, dass alle Einträge in den Zeilen um eine Position nach rechts (Richtung größerem Index) geschoben werden. An der ersten Position jeder Zeile rücken neue Einträge mit dem Wert 0 nach.
  Stand am Ende der Zeile der Wert 0, wird dieser nicht mehr verschoben sondern vom neuen Wert überschrieben und die Zeile ändert ihre Länge nicht (in diesem Fall wird die Zeile nicht ersetzt, sondern deren Einträge geändert). Steht im ursprünglichen Array am Ende der Zeile ein Wert ungleich 0, wird auch dieser verschoben und die entsprechende Zeile wird durch eine um einen Eintrag längere Zeile ersetzt.
  Vorbedingung(en): `input != null` und für alle gültigen `i` gilt `input[i] != null` und `input[i].length > 0`.
  Wird die Methode z.B. mit `test2` aufgerufen, entsteht folgendes Array:

0	1	7	1	0
0	3			
0	-8			
0	-1	3	0	1

- `boolean evenOccurrences(String s, char ch)` liefert `true` genau dann, wenn die Anzahl der Vorkommnisse des Zeichens `ch` in `s` eine gerade Zahl ist.
  Diese Methode muss rekursiv implementiert werden.
  Vorbedingung(en): `s != null`.

Deklarieren Sie auch neue Arrays, die für die Tests benötigt werden.
Testen Sie alle Methoden und deren Seiteneffekte in `main` mit zumindest folgenden Aufrufen und weiteren Aufrufen (z.B. mit `deepToString`) für die Ausgaben. Erzeugen Sie diese Ausgaben nur in `main`, nicht in den implementierten Methoden.

Aufruf	**Ausgabe in main auf der Konsole**
`result1 = generate(new int[]{0,-1})`	[[true, false, false, false, false], [false, false, false, false, false]]
`result2 = generate(new int[]{5})`	[[false, false, false, false, false, true]]
`result3 = generate(new int[]{})`	[]
`shift(test2)`	[[0, 1, 7, 1, 0], [0, 3], [0, -8], [0, -1, 3, 0, 1]]
`shift(test3)`	[[0, 5, -6], [0], [0, 0]]
`evenOccurrences("This is just a test!",'t')`	`false`
`evenOccurrences("This is just a test!",'T')`	`false`
`evenOccurrences("This is just a test!',' ')`	`true`
`evenOccurrences("This is just a test!",'s')`	`true`
`evenOccurrences("",'x')`	`true`

Methode	**Bewertungsgrundlage**	**Punkt(e)**
`main`	Deklarationen	/ 1
	Testfälle korrekt implementiert	/ 2
`generate`	Korrekte Anzahl der Zeilen	/ 2
	Korrekte Länge jeder Zeile	/ 4
	Richtige Einträge bei negativem Index	/ 1
	Richtige Einträge in Zeile mit Länge 5	/ 1
	Richtige Einträge in Zeile mit Überlänge 5	/ 1
`shift`	Korrektes Verschieben der Einträge	/ 3
	Korrektes Kopieren bei erweiterten Zeilen	/ 3
	Korrekte Länge aller Zeilen	/ 2
	Korrekte erste Stelle in den Zeilen	/ 1
`evenOccurrences`	Korrekter Methodenansatz (Rückgabe vorhanden)	/ 1
	Basisfall/Basisfälle vorhanden	/ 1
	Basisfall/Basisfälle korrekt	/ 1
	Forschritt der Rekursion vorhanden	/ 1
	Forschritt der Rekursion korrekt	/ 1
	Korrekter Rückgabewert	/ 4
Gesamt		/ 30

EP1 Programmiertest 6 WS2021 27. Jänner 2022	**Matrikelnummer**	**Familienname**	**Unterschrift**

Einschränkungen

- Sie dürfen **keine** zusätzlichen eigenen Hilfsmethoden oder globalen Variablen verwenden.
- Die vorgegebenen Methodenköpfe dürfen **nicht** erweitert oder geändert werden.
- Für die Implementierung der rekursiven Methode dürfen **keine** Schleifen verwendet werden.
- Sie dürfen Strings **nicht** per Referenz vergleichen.
- Sie dürfen **nicht** die Methoden `clone` und `System.arraycopy` verwenden.
- Sie dürfen **nur** folgende Methode(n) aus der Klasse `Arrays` verwenden: `deepToString`, `toString`
- Sie dürfen **nur** folgende Methode(n) aus der Klasse `String` verwenden: `charAt`, `equals`, `length`, `substring`, `isEmpty`
- Bis auf die Klasse `Math` dürfen **keine** weiteren Klassen der Java API verwendet werden.

Aufgabenstellung

Deklarieren und initialisieren Sie in `main` die folgende(n) Variable(n):

```
int[] test1 = {3, 0, 6, -1, 1};
int[][] test2 = {{0}, {6, -5}, {0, 0}, {0, 1, 2, 0}};
int[][] test3 = {{1, 2, 7, 3, 0}, {-8}, {0, 2}, {1, 4, -2, 1}};
```

Implementieren Sie folgende Methoden:

- `boolean[][] create(int[] input)` liefert ein neues Array zurück, in dem alle Zeilen eine Mindestlänge von 3 Einträgen haben und Einträge an in `input` spezifizierten Positionen `true` und alle übrigen Einträge `false` sind. Jede Zeile hat maximal einen Eintrag der `true` ist. In `input[i]` steht für die Zeile mit Index `i` der Spaltenindex dieses Eintrags. Ist `input[i]` kleiner 0, hat die Zeile keinen Eintrag `true` und alle 3 Einträge sind `false`. Die Zeilen des zurückgelieferten Arrays können länger als 3 sein und zwar dann, wenn `input[i]` größer als 2 ist. In diesem Fall hat die Zeile die Länge `input[i]+1`, sodass der letzte Eintrag `true` ist.
  Vorbedingung(en): `input != null` .
  Wird die Methode z.B. mit den Parametern `test1` aufgerufen, entsteht folgendes Array:

false	false	false	true			
true	false	false				
false	false	false	false	false	false	true
false	false	false				
false	true	false				

- `void move(int[][] input)` ändert das angegebene Array `input` so, dass alle Einträge in den Zeilen um eine Position nach rechts (Richtung größerem Index) geschoben werden. An der ersten Position jeder Zeile rücken neue Einträge mit dem Wert 0 nach. Stand am Ende der Zeile der Wert 0, wird dieser nicht mehr verschoben, sondern vom neuen Wert überschrieben und die Zeile ändert ihre Länge nicht (in diesem Fall wird die Zeile nicht ersetzt, sondern deren Einträge geändert). Steht im ursprünglichen Array am Ende der Zeile ein Wert ungleich 0, wird auch dieser verschoben und die entsprechende Zeile wird durch eine um einen Eintrag längere Zeile ersetzt.
  Vorbedingung(en): `input != null` und für alle gültigen `i` gilt `input[i] != null` und `input[i].length > 0`.
  Wird die Methode z.B. mit `test2` aufgerufen, entsteht folgendes Array:

0			
0	6	-5	
0	0		
0	0	1	2

- `boolean oddOccurrences(String s, char ch)` liefert `true` genau dann, wenn die Anzahl der Vorkommnisse des Zeichens `ch` in `s` eine ungerade Zahl ist.
  Diese Methode muss rekursiv implementiert werden.
  Vorbedingung(en): `s != null`.

Deklarieren Sie auch neue Arrays, die für die Tests benötigt werden.
Testen Sie alle Methoden und deren Seiteneffekte in `main` mit zumindest folgenden Aufrufen und weiteren Aufrufen (z.B. mit `deepToString`) für die Ausgaben. Erzeugen Sie diese Ausgaben nur in `main`, nicht in den implementierten Methoden.

Aufruf	**Ausgabe in main auf der Konsole**
`result1 = create(new int[]{3})`	[[false, false, false, true]]
`result2 = create(new int[]{-2, 0})`	[[false, false, false], [true, false, false]]
`result3 = create(new int[]{})`	[]
`move(test2)`	[[0], [0, 6, -5], [0, 0], [0, 0, 1, 2]]
`move(test3)`	[[0, 1, 2, 7, 3], [0, -8], [0, 0, 2], [0, 1, 4, -2, 1]]
`oddOccurrences("This is not a test!",'s')`	`true`
`oddOccurrences("This is not a test!",'T')`	`true`
`oddOccurrences("This is not a test!",'t')`	`true`
`oddOccurrences("This is not a test!',' ')`	`false`
`oddOccurrences("",'x')`	`false`

Methode	**Bewertungsgrundlage**	**Punkt(e)**
`main`	Deklarationen	/ 1
	Testfälle korrekt implementiert	/ 2
`create`	Korrekte Anzahl der Zeilen	/ 2
	Korrekte Länge jeder Zeile	/ 4
	Richtige Einträge bei negativem Index	/ 1
	Richtige Einträge in Zeile mit Länge 3	/ 1
	Richtige Einträge in Zeile mit Überlänge 3	/ 1
`move`	Korrektes Verschieben der Einträge	/ 3
	Korrektes Kopieren bei erweiterten Zeilen	/ 3
	Korrekte Länge aller Zeilen	/ 2
	Korrekte erste Stelle in den Zeilen	/ 1
`oddOccurrences`	Korrekter Methodenansatz (Rückgabe vorhanden)	/ 1
	Basisfall/Basisfälle vorhanden	/ 1
	Basisfall/Basisfälle korrekt	/ 1
	Forschritt der Rekursion vorhanden	/ 1
	Forschritt der Rekursion korrekt	/ 1
	Korrekter Rückgabewert	/ 4
Gesamt		/ 30