EP1 Programmiertest WS2021 10. Februar 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 = {{1}, {6, 7}, {1, 2, 3, 4, 5}, {}};
int[][] test2 = {{1, 2}, {1, 2, 3, 4}, {1, 2, 3}};
int[][] test3 = {{}, {1}};
int[] seq = {-2, 5, -5, 1, 2, -6, 4, -1};
```

Implementieren Sie folgende Methoden:

- **`int[][] uniformLength(int[][] input)`** liefert ein neues Array zurück, das rechteckig ist (alle Zeilen haben dieselbe Länge). Die Länge jeder Zeile entspricht dabei der Länge der längsten Zeile von `input`, die Anzahl der Zeilen ist gleich wie bei `input`. Jede Zeile von `input` wird in die entsprechende Zeile des neuen Arrays kopiert, wobei kürzere Zeilen solange wiederholt werden, bis das Ende der Zeile erreicht ist. Leere Zeilen in `input` werden im neuen Array mit lauter Nullen befüllt.
  Vorbedingung(en): `input != null`, `input.length > 0`, `input[i] != null` für alle gültigen `i`.
  Wird die Methode z.B. mit dem Parameter `test1` aufgerufen, entsteht folgendes Array:

1	1	1	1	1
6	7	6	7	6
1	2	3	4	5
0	0	0	0	0

- **`void clear(int[][] input, int index)`** entfernt aus jeder Zeile von `input` das Element mit Spaltenindex `index`. Ist an einem Spaltenindex kein Element vorhanden, soll die Zeile nicht verändert werden.
  Vorbedingung(en): `input != null`, `input.length > 0`, `input[i] != null` für alle gültigen `i`, `index >= 0`.
  Wird die Methode z.B. mit `test2` und `index = 2` aufgerufen, entsteht folgendes Array:

1	2	
1	2	4
1	2	

- **`boolean changeOfSign(int[] seq, int i1, int i2)`** überprüft, ob die Zahlenreihe `seq` von der Stelle `i1` bis zur Stelle `i2` (jeweils inklusive) alternierend ist. Bei einer alternierenden Zahlenreihe gibt es einen ständigen Vorzeichenwechsel, d.h., es gibt keine zwei aufeinander folgenden Zahlen mit demselben Vorzeichen. Ist die Zahlenreihe alternierend, soll `true` zurückgeliefert werden, andernfalls `false`.
  Diese Methode muss rekursiv implementiert werden.
  Vorbedingung(en): `seq != null`, `seq` enthält keine 0, `i1 <= i2`, `i1` und `i2` beschreiben jeweils einen gültigen Index von `seq`.

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 = uniformLength(test1)`	[[1, 1, 1, 1, 1], [6, 7, 6, 7, 6], [1, 2, 3, 4, 5], [0, 0, 0, 0, 0]]
`result2 = uniformLength(test2)`	[[1, 2, 1, 2], [1, 2, 3, 4], [1, 2, 3, 1]]
`result3 = uniformLength(test3)`	[[0], [1]]
`result4 = uniformLength(new int[][]{{}})`	[[]]
`clear(test2, 2)`	[[1, 2], [1, 2, 4], [1, 2]]
`clear(test3, 0)`	[[], []]
`changeOfSign(seq, 0, 7)`	false
`changeOfSign(seq, 0, 3)`	true
`changeOfSign(seq, 4, 7)`	true
`changeOfSign(seq, 3, 3)`	true
`changeOfSign(seq, 3, 4)`	false

Methode	**Bewertungsgrundlage**	**Punkt(e)**
`main`	Deklarationen	/ 1
	Testfälle korrekt implementiert	/ 2
`uniformLength`	Korrekte Bestimmung der längsten Zeile	/ 2
	Korrekte Dimensionen des neuen Arrays	/ 1
	Befüllung der Zeilen: korrekter Ansatz	/ 2
	Befüllung der Zeilen: korrektes Ergebnis	/ 3
	Korrekte Behandlung leerer Zeilen	/ 1
`clear`	Korrekte Zeilenlängen abhängig von `index`	/ 2
	Entfernen des Eintrags: korrekter Ansatz	/ 3
	Entfernen des Eintrags: korrektes Ergebnis	/ 4
`changeOfSign`	Korrekter Methodenansatz (Rückgabe vorhanden)	/ 1
	Basisfall/Basisfälle vorhanden	/ 1
	Basisfall/Basisfälle korrekt	/ 1
	Fortschritt der Rekursion vorhanden	/ 1
	Fortschritt der Rekursion korrekt	/ 1
	Korrekter Rückgabewert	/ 4
Gesamt		/ 30