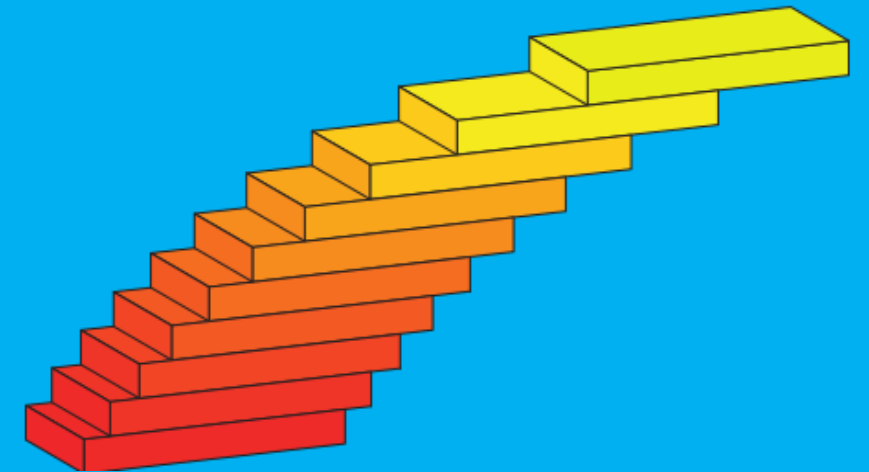


Parametrisierbare Aufgaben mit STACK: Drag-and-drop-Aufgaben



Stand: 09.10.2026

STACK // Drag-and-drop-Aufgaben

- In dieser Anleitung wird erklärt, wie STACK-Aufgaben erstellt werden können, in denen die Studierenden Elemente per Drag-and-drop verschieben müssen.
- So lassen sich beispielsweise Beweis puzzles und Zuordnungsaufgaben umsetzen.
- Wie STACK-Aufgaben mit solchen Funktionen erstellt werden können, wird im Folgenden an vier Beispielaufgaben demonstriert. Dazu werden Vorlagen aus der STACK-Fragenbibliothek genutzt.

Hinweis: Wir empfehlen, vor dem Anschauen dieser Anleitung zunächst die Klickanleitungen „Erstellung einer Aufgabe“ und im besten Fall „Die STACK Fragenbibliothek“ sowie „Fragevarianten erstellen und automatisiert testen“ durchzuarbeiten oder die gleichnamigen Videotutorials zu schauen.

STACK // Übersicht über Beispielaufgaben (1)

Wie STACK-Aufgaben mit Drag-and-drop-Funktionen Grafiken erstellt werden können, wird im Folgenden an diesen Beispielaufgaben demonstriert:

1. „Satz des Euklid“

Beweispuzzle: Die Studierenden sollen beweisen, dass es unendlich viele Primzahlen gibt. Dazu sind auf der rechten Seite dieses Bereichs verschiedene Beweisschritte aufgeführt, die die Studierenden per Drag-and-drop auf die linke Seite ziehen und dort in die richtige Reihenfolge bringen sollen.

2. „Teilbarkeit durch 6“

Auf der rechten Seite sind verschiedene Schritte zu finden, um zu überprüfen, ob eine natürliche Zahl durch 6 teilbar ist. Erneut sollen diese auf die linke Seite gezogen und dort korrekt angeordnet werden. Die Besonderheit ist hier, dass es zwei korrekte Lösungen gibt: Es spielt keine Rolle, ob zuerst die Teilbarkeit durch 2 oder erst die Teilbarkeit durch 3 überprüft wird.

Beweisen Sie, dass es unendlich viele Primzahlen gibt.

Erstellen Sie hier Ihre Lösung:

Von hier ziehen:

Dies widerspricht der Annahme, dass alle Primzahlen aufgezählt sind.

Also ist entweder Q selbst prim oder hat einen Primfaktor, der nicht in der Liste ist.

Keine der Primzahlen p_i teilt

Geben Sie das Vorgehen an um zu bestimmen, ob die Zahl $n = 150$ durch 6 teilbar ist.

Erstellen Sie hier Ihre Lösung:

Von hier ziehen:

Beides ist erfüllt, daher ist n durch 6 teilbar.

Prüfe, ob n durch 2 teilbar ist (d. h. n ist gerade).

Gehen sei eine Zahl

STACK // Übersicht über Beispielaufgaben (2)

3. „Quadratische Funktionen“

Hier sollen verschiedene Elemente kategorisiert werden: Für jede quadratische Funktion sollen die Studierenden entscheiden, ob sie keine, eine oder zwei reelle Nullstellen besitzt. Insbesondere ist die Antwort hier erneut nicht eindeutig, denn innerhalb der Kategorien können die Funktionen in beliebiger Reihenfolge angeordnet werden.

Ordnen Sie die quadratischen Funktionen in die linken Spalten ein, je nachdem, wie viele reelle Nullstellen sie besitzen.

 	Keine Nullstellen	Eine Nullstelle	Zwei Nullstellen	Von hier ziehen:
				$3x^2 - 5x - 8$
				$4x^2 + 4x + 1$
				$x^2 - x - 6$

4. „Dezimalzahlen und Brüche“

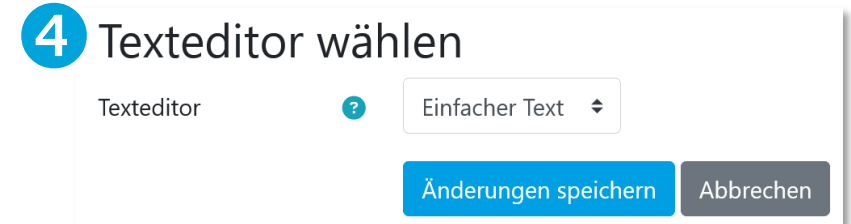
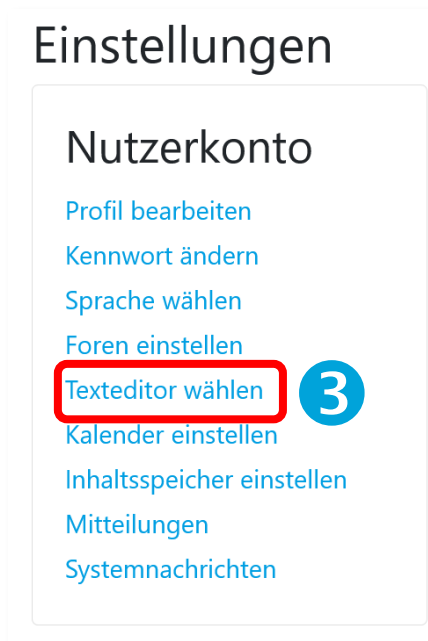
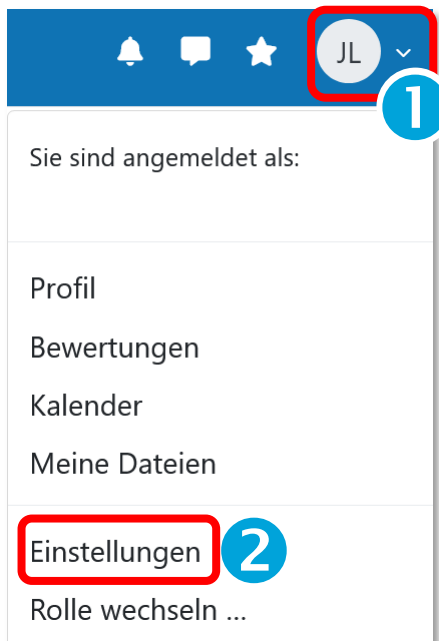
Auch hier gibt es mehrere Spalten, nur ist hier die Reihenfolge der Elemente innerhalb der Spalten ebenfalls relevant. Es geht hier darum, dass in jeder Zeile die gleiche Zahl in unterschiedlichen Schreibweisen angeordnet wird, beispielsweise können ganz links die Dezimalzahlen 3,6 und 7,75 platziert werden. Die korrekte Antwort ist hier eindeutig bis auf Vertauschen der beiden Zeilen.

Verschieben Sie die Elemente, um den Zahlen ihre anderen Schreibweisen zuzuordnen.

 	Dezimalzahl	Bruch	Gemischter Bruch	Von hier ziehen:
				$3\frac{3}{5}$
				3,6
				$\frac{18}{5}$

STACK // Änderung des Texteditors

- Für die Dropdown-Anwendungen ist der in Moodle standardmäßig eingestellte Texteditor „TinyMCE“ nicht geeignet.
- Ändern Sie zu Beginn der Aufgabenerstellung daher den Texteditor in den Moodle-Einstellungen.
- Klicken Sie dazu oben rechts auf **1 Ihr Moodle-Profilfoto oder Ihre Initialen**, dann auf **2 Einstellungen** und schließlich auf **3 Texteditor wählen**. Wählen Sie dann die Option „Einfacher Text“ aus der Liste aus und speichern Sie die Änderungen **4**.



Hinweis: Der Texteditor wird so bis auf Widerruf geändert. Wenn Sie nach der Aufgabenerstellung für andere Moodle-Inhalte wieder den ursprünglichen Texteditor verwenden möchten, können Sie die Wahl des Texteditor über den gleichen Weg zurück ändern.

STACK // Laden der Vorlagen aus der STACK-Fragenbibliothek

- Erstellen Sie zunächst eine neue STACK-Aufgabe und klicken Sie direkt unterhalb des Felds „Fragetitel“ auf **1 STACK Fragenbibliothek**.
- Wählen Sie nun die **2 Kategorie** Ihrer Moodle-Fragensammlung aus, in die Sie die Vorlagen für die Drag-and-drop-Aufgaben importieren möchten.
- Öffnen Sie dann den Ordner „Doc-Examples“ der STACK-Fragenbibliothek, klicken auf die **3 Aufgabe** „Parsons-1-minimal“ und schließlich auf **4 Importieren**.
- Wiederholen Sie dies mit den Aufgaben „Parsons-2-Interchangable-order“, „Drag_and_drop-1-grouping“ und „Drag_and_drop-2-Grid“.

Fragetitel 

1 [STACK Fragenbibliothek](#)  [STACK Dokumentation](#) | [Community-Website](#)

Fragen werden in folgende Kategorie importiert:

2

4 [Importieren](#) [Ordner importieren](#)



Aufgabe 1

„Satz des Euklid“



STACK // Aufgabe 1: Fragetitel und Aufgabenvariablen (1)

- Navigieren Sie in Ihrer Moodle-Fragensammlung zur soeben importierten Aufgabe „Parsons proofs 1- Minimal“ und öffnen Sie sie im Bearbeitungsmodus.
- Ändern Sie nun den Fragetitel der Aufgabe, zum Beispiel zu „Satz des Euklid“.
- In den **1 Aufgabenvariablen** sehen Sie eine Variable `proof_steps`, die im Folgenden angepasst wird. Dies ist eine Liste mit den einzelnen Elementen (also Beweisschritten), die beim Öffnen der Aufgabe auf der rechten Seite zu sehen sein und schließlich von den Studierenden verschoben werden sollen.
- Jedes dieser Elemente wird durch eine Unterliste definiert, deren erstes Element eine eindeutige Bezeichnung des Beweisschritts sein muss. Diese müssen alle unterschiedlich sein, können sonst aber beliebig gewählt werden.
- Dahinter wird dann jeweils der Text notiert, den die Studierenden sehen sollen. Dabei können auch LaTeX-Formeln verwendet werden (mit der Besonderheit, dass an jeder Stelle, wo sonst ein Backslash verwendet werden würde, *zwei* Backslashes benötigt werden, z. B. `\\(\\sqrt{2}\\)` statt `\(\\sqrt{2}\\)`).

1 Aufgabenvariablen ?

```

/*import parsons' code library*/
stack_include("contrib1://prooflib.mac");

/*Steps of proof*/
proof_steps: [
    ["assume", "Assume, for a contradiction, that  $\log_2(3)$  is rational."],
    ["defn_rat", "Then  $\log_2(3) = \frac{p}{q} > 0$  where"],
    ["defn_rat2", " $n$  and  $m$  are positive"],

```

STACK // Aufgabe 1: Aufgabenvariablen (2)

- Fügen Sie in der Variable `proof_steps` die einzelnen Schritte für den Beweis des Satzes von Euklid ein, zum Beispiel so (Tipp: Als Bezeichnungen der Schritte können auch einfach Zahlen verwendet werden):

```
proof_steps: [  
    ["1", "Angenommen, es gibt nur endlich viele Primzahlen  $(p_1, p_2, \dots, p_n)$ ."],  
    ["2", "Betrachte die Zahl  $(Q = p_1 \cdot p_2 \cdot \dots \cdot p_{n+1})$ ."],  
    ["3", "Keine der Primzahlen  $(p_i)$  teilt  $(Q)$ , denn bei Division bleibt immer Rest 1."],  
    ["4", "Also ist entweder  $(Q)$  selbst prim oder hat einen Primfaktor, der nicht in der Liste ist."],  
    ["5", "Dies widerspricht der Annahme, dass alle Primzahlen aufgezählt sind."],  
    ["6", "Daher gibt es unendlich viele Primzahlen."]  
];
```

- Damit die Beweisschritte den Studierenden nicht immer in der gleichen Reihenfolge angezeigt werden (die Elemente insbesondere auch nicht bereits in der korrekten Reihenfolge zu sehen sind) ist es sinnvoll, noch die Funktion `random_permutation` anzuwenden und die einzelnen Beweisschritte somit zufällig zu permutieren:

```
proof_steps: random_permutation(proof_steps);
```

- Passen Sie zuletzt noch die Variable `ta` für die Musterlösung der Aufgabe an. Hier müssen die Bezeichnungen der Beweisschritte (hier: die Zahlen in Anführungszeichen), in der korrekten Reihenfolge angegeben werden:

```
ta: proof("1", "2", "3", "4", "5", "6");
```

STACK // Aufgabe 1: Fragetext

- Passen Sie im Feld **❶ Fragetext** die Aufgabenstellung der Aufgabe an.
- Da der „Einfacher Text“-Texteditor von Moodle verwendet wird, muss hier mit HTML gearbeitet werden. Wenn Sie den Aufgabentext zwischen die vorhandenen <p>-Tags einfügen, wird dieser in einem eigenen Absatz angezeigt (eine Übersicht über die wichtigsten HTML-Befehle finden Sie auf der nächsten Folie), zum Beispiel so:

<p>Beweisen Sie, dass es unendlich viele Primzahlen gibt.</p>

- Darunter befindet sich ein sogenannter **❷ Parsons-Block**, mit dessen Hilfe der Drag-and-drop-Bereich im Fragetext realisiert wird. Im öffnenden Tag ist zu sehen, dass der Block mit dem Eingabefeld „ans1“ verknüpft ist. Hier müssen keine Änderungen vorgenommen werden, da alles bereits korrekt konfiguriert ist.

❶ Fragetext  

❷

```
<p>Beweisen Sie, dass es unendlich viele Primzahlen gibt.</p>
<p>[[parsons input="ans1"]] {# parsons_encode(proof_steps) #} [[/parsons ]]</p>
<p>[[input:ans1]] [[validation:ans1]]</p>
```

HTML-Format 

Hinweis: Der Name „Parsons“ kommt daher, dass Aufgaben, in denen Fragmente in eine richtige Reihenfolge gebracht werden müssen, vor allem im Kontext von Informatik und Programmierung häufig „Parsons-Probleme“ genannt werden.

STACK // Exkurs: Übersicht über die wichtigsten HTML-Befehle

HTML-Befehl	Beschreibung
<code><p>Text</p></code>	Text erscheint in eigenem Absatz
<code>
</code>	Zeilenumbruch
<code>Text</code>	Text wird fett gesetzt
<code><i>Text</i></code>	Text wird <i>kursiv</i> gesetzt
<code><u>Text</u></code>	Text wird <u>unterstrichen</u>
<code>~~Text~~</code>	Text wird ~~durchgestrichen~~
<code><hr></code>	Horizontale Linie
<code><h1>Überschrift</h1></code> ... <code><h6>Überschrift</h6></code>	Überschriften auf verschiedenen Ebenen
<code>Text</code>	Einfügen eines Hyperlinks
<code><!-- Kommentar --></code>	Kommentar (Text wird nicht angezeigt)
<code>Text</code>	Farbiger Text (hier: rot)

STACK // Aufgabe 1: Aufgabenbeschreibung, Aufgabenhinweis und Bereich „Eingabe“

- Passen Sie den Eintrag im Feld „Aufgabenbeschreibung“ an oder entfernen Sie ihn ganz. Das Feld ist ohnehin nur für interne Zwecke und sein Inhalt wird den Studierenden nicht angezeigt.
- Im Feld „Aufgabenhinweis“ kann der bereits eingetragene Wert `{@ map(first, proof_steps) @}` beibehalten werden.
- Im Bereich **1 Eingabe** müssen ebenfalls keine Änderungen vorgenommen werden. Hier ist als Musterlösung `[ta, proof_steps]` eingetragen und als Eingabetyp die Option „Parsons“ – ein Eingabetyp eigens für Drag-and-drop-Aufgaben – ausgewählt.

1  Eingabe: ans1

Eingabetyp		Parsons 
Musterlösung		[ta, proof_steps]

STACK // Aufgabe 1: Rückmeldebaum

- Auch im Rückmeldebaum müssen keine Änderungen vorgenommen werden.
- Hier ist allerdings interessant zu wissen, dass in den Feedbackvariablen eine Variable `pd` definiert wird, die eine Art Distanz zwischen der Antwort der Studierenden und der korrekten Lösung ist.
- Im einzigen Knoten des Rückmeldebaums wird überprüft, ob dieser Wert 0 ist, das heißt die Antwort exakt mit der korrekten Antwort übereinstimmt.
- Die Funktion `proof_assessment_display`, die im **1 Feedbacktext** für den Fall einer falschen Antwort verwendet wird, sorgt dafür, dass an dieser Stelle eine **2 Übersicht** darüber angezeigt wird, welche Elemente richtig und welche nicht richtig angeordnet wurden.

Hinweis: Es ist möglich, in einem weiteren Knoten zu überprüfen, ob die Distanz `pd` bspw. 1 ist. Dies wäre dann der Fall, wenn nur zwei Beweisschritte vertauscht sind. Die Studierenden könnten dann Teilpunkte erhalten.

1 Knoten 1 FALSCH **?** `{@proof_assessment_display(saa, proof_steps)@}`
Feedback

2 ✕ Falsche Antwort.

Angenommen, es gibt nur endlich viele Primzahlen $p_1, p_2, \dots, p_n$.	✓
Betrachte die Zahl $Q = p_1 \cdot p_2 \cdot \dots \cdot p_n + 1$.	✓
Keine der Primzahlen p_i teilt Q , denn bei Division bleibt immer Rest 1.	✓
Dies widerspricht der Annahme, dass alle Primzahlen aufgezählt sind.	↓
Also ist entweder Q selbst prim oder hat einen Primfaktor, der nicht in der Liste ist.	↑
Daher gibt es unendlich viele Primzahlen.	✓

STACK // Aufgabe 1: Standard-Feedback anpassen

- Öffnen Sie zuletzt noch den Bereich „Optionen“ und betrachten Sie dort die Felder für das ❶ **Standard-Feedback**.
- Da die Fragen aus der STACK-Fragenbibliothek englischsprachig sind, sind auch diese Texte auf Englisch. Ändern Sie die Texte daher auf die standardmäßigen deutschen Texte (siehe Screenshot).

❶ Standard Feedback für richtige Antworten	<p> <i class="fa fa-check"> </i> Richtige Antwort, gut gemacht!</p> HTML-Format
Standard Feedback für teilweise richtige Antworten	<p> <i class="fa fa-adjust"> </i> Ihre Antwort ist teilweise korrekt.</p> HTML-Format
Standard Feedback für falsche Antworten	<p> <i class="fa fa-times"> </i> Falsche Antwort.</p> HTML-Format



Aufgabe 2

„Teilbarkeit durch 6“



STACK // Aufgabe 2: Fragetitel und Aufgabenvariablen

- Navigieren Sie in Ihrer Moodle-Fragensammlung zur importierten Aufgabe „parsons proof 2-Interchangable order“ und öffnen Sie sie im Bearbeitungsmodus. Ändern Sie erneut den Fragetitel der Aufgabe, zum Beispiel zu „Teilbarkeit durch 6“.
- Tragen Sie in der Variable `proof_steps` erneut Ihre eigenen Schritte ein, zum Beispiel:

```
proof_steps: [  
    ["1", "Gegeben sei eine Zahl  $n=90$ . Wir wollen prüfen, ob  $n$  durch 6 teilbar ist."],  
    ["2a", "Prüfe, ob  $n$  durch 2 teilbar ist (d. h.  $n$  ist gerade)."],  
    ["3a", "Es gilt:  $n = 2 \cdot 45$ , daher ist  $n$  durch 2 teilbar."],  
    ["2b", "Prüfe, ob  $n$  durch 3 teilbar ist (d. h. Quersumme von  $n$  ist durch 3  
        teilbar)."],  
    ["3b", "Es gilt:  $n = 3 \cdot 30$ , daher ist  $n$  durch 3 teilbar."],  
    ["4", "Beides ist erfüllt, daher ist  $n$  durch 6 teilbar."]  
];
```

- Zur Erinnerung: Bei dieser Aufgabe ist die korrekte Reihenfolge der Schritte nicht eindeutig, weil entweder zuerst die Teilbarkeit durch 2 (Schritte 2a und 3a) oder die Teilbarkeit durch 3 (Schritte 2b und 3b) überprüft werden kann. Dies muss in der Variable `ta` für die korrekte Antwort berücksichtigt werden (siehe nächste Folie).

STACK // Aufgabe 2: Aufgabenvariablen – Musterlösung der Aufgabe

- In der Variable `ta` wird aktuell noch die Funktion `proof_iff` verwendet, mit zwei Beweisen (oder besser gesagt Teilbeweisen) als Argumenten, also nach folgendem Schema: `ta: proof_iff( proof(...), proof(...) );`
Das bedeutet, dass die Schritte innerhalb der beiden Teilbeweise genau so angeordnet werden müssen, wie es hier notiert ist, aber es egal ist, ob erst der erste oder erst der zweite Teilbeweis folgt.
- Für die Aufgabe zur Teilbarkeit durch 6 sind die Anforderungen ähnlich, aber entsprechen nicht ganz der Situation aus der Vorlage: Die Studierenden müssen unbedingt mit Schritt 1 beginnen und erst dann ist es egal, ob sie mit den Schritten 2a und 3a oder mit 2b und 3b fortfahren (dies bewirkt die Funktion `proof_c`, s. u.). Zum Schluss muss dann unbedingt Schritt 4 folgen. Diese Anforderungen können wie folgt abgebildet werden:

```
ta: proof("1", proof_c(proof("2a", "3a"), proof("2b", "3b")), "4" );
```

- Unterhalb von `ta` findet sich eine Variable `wa`. Dies ist eine beliebige falsche Antwort, die nur für einen Testfall benötigt wird (siehe Klickanleitung und Video „Fragevarianten erstellen und automatisiert testen“). Konstruieren Sie hier eine beliebige falsche Antwort, zum Beispiel:

```
wa: proof("4", proof_c(proof("2a", "3a"), proof("2b", "3b")), "1" );
```

- Darunter gibt es noch eine Variable `ta1`: Mithilfe der Funktion `proof_alternatives` wird aus der Variable `ta` eine Liste mit alle korrekten Antworten bestimmt. Diese Variable muss nicht angepasst werden, ebenso wie die Variable `tas`, die diese Liste noch in eine andere Form umwandelt.

STACK // Aufgabe 2: Weitere Felder

- Ändern Sie im Fragetext wieder die Aufgabenstellung, zum Beispiel: „*Geben Sie das Vorgehen an um zu bestimmen, ob die Zahl $(n=90)$ durch (6) teilbar ist.*“
- Der Rest des Felds „Fragetext“ inklusive des Parsons-Blocks kann unverändert bleiben.
- In der Vorlage ist im Feld „Allgemeines Feedback“ ein kurzer englischsprachiger Text eingetragen, in dem die Studierenden darauf aufmerksam gemacht werden, dass die Aufgabe zwei verschiedene Lösungen besitzt. Diesen Text kann übersetzt und angepasst werden (siehe 1).
- Passen Sie erneut die Aufgabenbeschreibung an oder entfernen Sie sie.
- Im Feld „Aufgabenhinweis“ und im Bereich „Eingabe“ müssen keine Änderungen im Vergleich zur Vorlage vorgenommen werden. Auch der Rückmeldebaum ist bereits korrekt vorkonfiguriert und kann ggf. noch durch weitere Knoten ergänzt werden.
- Ändern Sie analog zur ersten Beispielaufgabe das Standard-Feedback im Abschnitt „Optionen“ (siehe Folie 14).

1 Allgemeines Feedback ?

Hier sehen Sie das Vorgehen strukturiert aufgeschrieben:

```
{@proof_display(tal[2], proof_steps)@}
```

Beachten Sie, dass Sie zuerst die Teilbarkeit durch 2 oder die Teilbarkeit durch 3 überprüfen können, daher besitzt die Aufgabe zwei mögliche Lösungen:

```
<table><tr>
```

```
<td><div class="proof">{@proof_display_para(tal[1], proof_steps)@}</div></td>
```

```
<td><div class="proof">{@proof_display_para(tal[2], proof_steps)@}</div></td>
```

```
</tr></table>
```

Erkennen Sie den Unterschied zwischen den beiden Lösungen?

STACK // Aufgabe 2: Randomisierung der Aufgabe

- Um anstatt des festen Werts von 90 eine randomisierte Zahl zu verwenden, erstellen Sie zunächst eine entsprechende Aufgabenvariable, zum Beispiel `n`: `rand_with_step(12*6, 26*6, 6);`
- Nun muss der feste Wert 90 überall durch die neue Variable `n` ersetzt werden. Dazu gehören auch die Schritte 1, 3a und 3b in der Variable `proof_steps`. Verwenden Sie dazu die Funktion `sconcat(1, 2, 3)`, mit der in Maxima mehrere Zeichenketten wie 1, 2 und 3 zusammengefügt werden können:

```
proof_steps: [  
  1 2 3  
  ["1", sconcat("Gegeben sei eine Zahl \\\(n=", n, "\\). Wir wollen prüfen, ob \\\(n\\) durch 6  
    teilbar ist.")],  
  ["2a", "Prüfe, ob \\\(n\\) durch 2 teilbar ist (d. h. \\\(n\\) ist gerade."],  
  ["3a", sconcat("Es gilt: \\\(n = 2 \\\cdot", n/2, "\\), daher ist \\\(n\\) durch 2 teilbar.")],  
  ["2b", "Prüfe, ob \\\(n\\) durch 3 teilbar ist (d. h. Quersumme von \\\(n\\) ist durch 3  
    teilbar)."],  
  ["3b", sconcat("Es gilt: \\\(n = 3 \\\cdot", n/3, "\\), daher ist \\\(n\\) durch 3 teilbar.")],  
  ["4", "Beides ist erfüllt, daher ist \\\(n\\) durch 6 teilbar."]  
];
```

- Verwenden Sie `n` im Fragetext, also so: „Geben Sie das Vorgehen an um zu bestimmen, ob die Zahl $\backslash(n=\{@n@}\backslash)$ durch $\backslash(6)$ teilbar ist.“
- Erwähnen Sie `n` auch im Feld „Aufgabenhinweis“, damit STACK es als verschiedene Varianten der Aufgabe betrachtet, wenn die Aufgabenstellung verschiedene Werte für `n` beinhaltet, z. B. `{@ map(first, proof_steps) @}, \\\(n=\{@n@}\backslash)`. Erstellen Sie dann neue Fragevarianten.

Aufgabe 3

„Quadratische Funktionen“



STACK // Aufgabe 3: Fragetitel und Aufgabenvariablen

- Navigieren Sie in Ihrer Moodle-Fragensammlung zur importierten Aufgabe „Drag and drop - 1 - grouping example“ und öffnen Sie sie im Bearbeitungsmodus. Ändern Sie erneut den Fragetitel der Aufgabe, zum Beispiel zu „Quadratische Funktionen“.
- Die Aufgabenvariable, in der die zu sortierenden Elemente definiert werden, heißt in dieser Vorlage nicht `proof_steps`, sondern `steps`. Fügen Sie dort wieder Ihre eigenen Inhalte ein, zum Beispiel:

```
steps : [  
    ["1", "\\(x^2 + x + 1\\)"],  
    ["2", "\\(2x^2 + 3x + 5\\)"],  
    ["3", "\\(x^2 - 2x + 1\\)"],  
    ["4", "\\(4x^2 + 4x + 1\\)"],  
    ["5", "\\(x^2 - x - 6\\)"],  
    ["6", "\\(3x^2 - 5x - 8\\)"]  
];
```

- Wichtig ist als nächstes die Variable `headers`. Hier werden die **1 Spaltenüberschriften** definiert. Ersetzen Sie die Einträge aus der Vorlage wie folgt:

```
headers: [  
    "Keine Nullstellen",  
    "Eine Nullstelle",  
    "Zwei Nullstellen",  
];
```

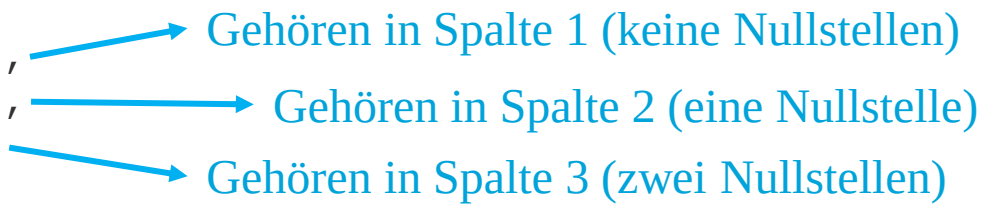
1 Aufgabenvariablen ?

```
/*Categories to drag to, can be changed*/  
headers: [  
    "Keine Nullstellen",  
    "Eine Nullstelle",  
    "Zwei Nullstellen"  
];
```

STACK // Aufgabe 3: Aufgabenvariablen – Musterlösung der Aufgabe

- Bei diesem Aufgabentyp besteht die Musterlösung ta aus einer Unterliste pro Spalte.
- In diesen Unterlisten müssen die Bezeichnungen der Elemente eingefügt werden, die von den Studierenden in die entsprechenden Kategorien sortiert werden sollen:

```
ta: [  
    ["1", "2"],  
    ["3", "4"],  
    ["5", "6"]  
];
```



The diagram shows three blue arrows pointing from the list elements to their respective column categories:

- Arrow from ["1", "2"] to "Gehören in Spalte 1 (keine Nullstellen)"
- Arrow from ["3", "4"] to "Gehören in Spalte 2 (eine Nullstelle)"
- Arrow from ["5", "6"] to "Gehören in Spalte 3 (zwei Nullstellen)"

- Konstruieren Sie wie bei der vorherigen Beispielaufgabe in der Variable wa wieder eine falsche Antwort. Diese wird wieder nur für einen vorkonfigurierten Testfall benötigt. Beispiel:

```
wa: [  
    ["1", "3"],  
    ["2", "5"],  
    ["4", "6"]  
];
```

STACK // Aufgabe 3: Weitere Felder

- Ändern Sie im Fragetext wieder die Aufgabenstellung, zum Beispiel: „*Ordnen Sie die quadratischen Funktionen in die linken Spalten ein, je nachdem, wie viele reelle Nullstellen sie besitzen.*“
- Der Rest des Felds „Fragetext“ inklusive des Parsons-Blocks kann erneut unverändert bleiben. Nur wenn Sie in den Aufgabenvariablen eine andere Zahl an Spalten als 3 festgelegt haben, müssen Sie im öffnenden Tag des Parsons-Blocks noch den Wert des **1 columns-Attributs** entsprechend anpassen.
- Passen Sie erneut die Aufgabenbeschreibung an oder entfernen Sie sie.
- Im Feld „Aufgabenhinweis“ und im Bereich „Eingabe“ müssen keine Änderungen im Vergleich zur Vorlage vorgenommen werden. Nur im Fall, dass Sie die Anzahl der Spalten verändert haben, müssen Sie im Feld **2 Musterlösung** der Wert 3 auf die tatsächliche Spaltenanzahl ändern.
- Auch der Rückmeldebaum kann unverändert aus der Vorlage übernommen werden.
- Ändern Sie analog zur ersten Beispielaufgabe das Standard-Feedback im Abschnitt „Optionen“ (siehe Folie 14).

Fragetext  

```
<p>Ordnen Sie die quadratischen Funktionen in die linken Spalten  
ein, je nachdem, wie viele reelle Nullstellen sie besitzen.</p>  
[[parsons input="ans1" columns="3"] 1  
{  
  "steps" : {#match_encode(steps)#},  
  "headers" : {#headers#}  
}  
[[/parsons]]  
<p>[[input:ans1]] [[validation:ans1]]</p>
```

2 Musterlösung  [ta, steps, 3]

Hinweis: Im Rückmeldebaum wird diesmal keine Distanz zwischen der Antwort der Studierenden und einer korrekten Lösung bestimmt. Stattdessen werden im Feld „Feedback-Variablen“ Variablen sans bzw. tans erstellt und im einzigen Knoten miteinander verglichen.



Aufgabe 4

„Dezimalzahlen und Brüche“



STACK // Aufgabe 4: Fragetitel und Aufgabenvariablen

- Navigieren Sie in Ihrer Moodle-Fragensammlung zur importierten Aufgabe „Drag and drop -2- Grid“ und öffnen Sie sie im Bearbeitungsmodus. Ändern Sie erneut den Fragetitel der Aufgabe, zum Beispiel zu „Dezimalzahlen und Brüche“.
- Analog zur dritten Beispielaufgabe heißt die Aufgabenvariable, in der die zu sortierenden Elemente definiert werden, `steps`. Fügen Sie dort wieder Ihre eigenen Inhalte ein, zum Beispiel:

```
steps : [  
    ["d1", "\\(7, \\!75\\)"],  
    ["b1", "\\(\\frac{31}{4}\\)"],  
    ["g1", "\\(7\\frac{3}{4}\\)"],  
    ["d2", "\\(3, \\!6\\)"],  
    ["b2", "\\(\\frac{18}{5}\\)"],  
    ["g2", "\\(3\\frac{3}{5}\\)"]  
];
```

- Fügen Sie erneut in der Variable `headers` die Spaltenüberschriften ein:

```
headers: [  
    "Dezimalzahl",  
    "Bruch",  
    "Gemischter Bruch",  
];
```

STACK // Aufgabe 4: Aufgabenvariablen – Musterlösung der Aufgabe

- Ähnlich wie in der dritten Beispielaufgabe besteht die Musterlösung ta auch hier aus einer Unterliste pro Spalte. Erneut müssen in den Unterlisten die Bezeichnungen der Elemente eingefügt werden, die von den Studierenden in die entsprechenden Kategorien sortiert werden sollen. Wichtig ist diesmal auch die Reihenfolge der Elemente innerhalb der Unterlisten:

```
ta: [  
    ["d1", "d2"],  
    ["b1", "b2"],  
    ["g1", "g2"]  
];
```

- Diesmal enthält die Vorlage noch eine Variable ta2 (für einen vorkonfigurierten Testfall), in der eine weitere richtige Lösung konstruiert werden soll. Konstruieren Sie außerdem wieder eine falsche Antwort (Variable wa):

```
ta2: [  
    ["d2", "d1"],  
    ["b2", "b1"],  
    ["g2", "g1"]  
];
```

```
wa: [  
    ["d2", "d1"],  
    ["g1", "b1"],  
    ["g2", "b2"]  
];
```

STACK // Aufgabe 3: Weitere Felder

- Ändern Sie im Fragetext wieder die Aufgabenstellung, zum Beispiel: „*Verschieben Sie die Elemente, um die Zahlen ihre anderen Schreibweisen zuzuordnen.*“
- Der Rest des Felds „Fragetext“ inklusive des Parsons-Blocks kann erneut unverändert bleiben. Nur wenn Sie in den Aufgabenvariablen eine andere Zahl an Spalten als 3 festgelegt haben, müssen Sie im öffnenden Tag des Parsons-Blocks noch den Wert des **1 columns-Attributs** entsprechend anpassen. Diesmal muss darüber hinaus auch das **2 rows-Attribut** angepasst werden, wenn Sie die Anzahl an Zeilen geändert haben.
- Passen Sie erneut die Aufgabenbeschreibung an oder entfernen Sie sie.
- Im Feld „Aufgabenhinweis“ und im Bereich „Eingabe“ müssen keine Änderungen im Vergleich zur Vorlage vorgenommen werden. Achten Sie nur analog zu Folie 23 darauf, dass im Feld „Musterlösung“ die korrekte Anzahl an Spalten (und diesmal auch Zeilen) eingetragen ist.
- Auch der Rückmeldebaum kann unverändert aus der Vorlage übernommen werden.
- Ändern Sie analog zur ersten Beispielaufgabe das Standard-Feedback im Abschnitt „Optionen“ (siehe Folie 14).

Fragetext ! ?

```
<p>Verschieben Sie die Elemente, um die Zahlen ihre anderen  
Schreibweisen zuzuordnen.  
[[parsons input="ans1" 1columns="3" 2rows="2"]  
{  
  "steps": {#match_encode(steps)#},  
  "headers": {#headers#}  
}  
[[/parsons]]  
<p>[[input:ans1]] [[validation:ans1]] </p>
```

STACK // Moodle-Kurs „Tutorials STACK“

- Dort finden Sie Videotutorials, Beispielaufgaben, weitere Klickanleitungen und mehr
- Link: <https://moodle.ruhr-uni-bochum.de/course/view.php?id=22225>
- Passend zu dieser Klickanleitung empfehlen wir Ihnen:
 - Videotutorial „Drag-and-drop-Aufgaben mit STACK“
 - Offizielle Dokumentation von STACK zu Drag-and-drop-Aufgaben: https://docs.stack-assessment.org/en/Specialist_tools/Drag_and_drop/Parsons



The screenshot shows the Moodle interface for the course 'Tutorials zu STACK' at the Ruhr-Universität Bochum (RUB). The header includes the RUB logo, navigation links (Dashboard, LEARN2Gether, OpenRUB, Support), and a user profile icon. The main content area features the title 'Tutorials zu STACK' with a colorful staircase graphic. Below the title, there is a paragraph explaining that STACK is a Moodle question type for digital mathematical tasks. Another paragraph mentions that the course covers various aspects of STACK through video tutorials and example tasks. A final paragraph states that the course is open and accessible without login, but an account is needed for full functionality.

RUB Dashboard LEARN2Gether OpenRUB Support Teilnehmer/in

Alles einklappen

Tutorials zu STACK

STACK ist ein Moodle-Fragetyp, mit dem sich geschlossene und offene digitale mathematische Aufgaben umsetzen lassen, die automatisch ausgewertet werden. Dabei ist es möglich, dass STACK Aufgabenstellungen zufällig generiert und den Lernenden ein differenziertes und adaptives Feedback bereitstellt.

In den Abschnitten des Kurses werden unterschiedliche Aspekte des Fragetyps STACK in Form von Videotutorials und Beispielaufgaben vermittelt. Zum Start empfehlen wir Ihnen die Inhalte des Abschnitts "Einführung in STACK" und "Eine STACK-Aufgabe erstellen", die Grundlagen vermitteln. Die weiteren Abschnitte können in beliebiger Reihenfolge bearbeitet werden.

Dieser Kurs ist komplett öffentlich und ohne Login zugänglich. Um alle Funktionen uneingeschränkt nutzen zu können, ist allerdings ein Account für das Moodle der Ruhr-Universität Bochum notwendig. Falls Sie noch keinen Account besitzen, können Sie sich über diesen Link kostenlos und unkompliziert einen Account erstellen. Danach müssen Sie sich noch in den Kurs einschreiben. Dies funktioniert unter diesem Link.



Dieses Dokument steht – ausgenommen verwendeter Logos bzw. Wort-/Bildmarken, Screenshots und anders gekennzeichnete Elemente – unter einer CC BY-SA 4.0 Lizenz. Sie dürfen die Inhalte **vervielfältigen, verwenden, verarbeiten, vermischen und verbreiten** unter der **Bedingung**, dass Sie die Originalautor:innen nennen und Beiträge, die durch Remix, Veränderung oder Aufbau auf diesem Video entstehen, nur unter derselben Lizenz wie das Original verbreiten.



Anmerkung für Nachnutzende | Version 1.0.a vom 09.10.2026

Empfohlene Zitation:

„Parametrisierbare Aufgaben mit STACK: Drag-and-drop-Aufgaben“, Team E-Learning@Hochschule Ruhr West von Jonas Lache, ist lizenziert unter einer CC BY-SA 4.0 International Lizenz (<https://creativecommons.org/licenses/by-sa/4.0/>).

Ausgenommen von der Lizenz sind die verwendeten Wort-/Bildmarken und alle anders gekennzeichneten Elemente.

Dieses Video basiert auf dem Video „Drag-and-drop-Aufgaben mit STACK“, Team E-Learning@Hochschule Ruhr West von Jonas Lache, das unter CC BY-SA 4.0 (<https://creativecommons.org/licenses/by-sa/4.0/>) im Moodle-Kurs „E-Learning@HRW – Selbstlernkurs STACK“ (<https://elearning.hs-ruhrwest.de/course/view.php?id=24999>) veröffentlicht wurde.

Zusätzliche Informationen für Nachnutzende:

Ursprungsort:

Moodle-Kurs „E-Learning@HRW – Selbstlernkurs STACK“ (<https://elearning.hs-ruhrwest.de/course/view.php?id=24999>), Stand 09.10.2026, Version 1.0.a

Urheber:in(nen):

Jonas Lache, Team E-Learning@Hochschule Ruhr West, Mail: elearning@hs-ruhrwest.de

Bildnachweise (einzeln):

- Logo des Assessment-Systems STACK: stack-assessment.org, CC BY-SA 4.0
- Screenshots der Aufgaben „Parsons-1-minimal“, „Parsons-2-Interchangable-order“, „Drag_and_drop-1-grouping“ und „Drag_and_drop-2-Grid“ aus der STACK-Fragenbibliothek, CC BY-SA 4.0