Good Practice Multiple Choice

Multiple Choice (MC) Fragen führen immer wieder zu Diskussionen. Wir möchten daher Anregungen geben, wie MC Fragen (im Kontext der Informatik) geeignet gestellt werden können. Wesentlich detaillierte, aber gleichzeitig kompakt dargestellte Informationen, zum Thema Design von MC Fragen gibt z.B. René Krebs in seiner „Anleitung zur Herstellung von MC-Fragen und MC-Prüfungen für die ärztliche Ausbildung“.

Fragen wo nur zwischen wahr und falsch gewählt wird sind problematisch, weil Antworten (leicht) erraten werden können. Bei gleicher Verteilung von wahren und falschen Aussagen führt das durchgängige Ankreuzen von wahr (oder falsch) bereits zur Hälfte der zu erreichenden Punkte. Dies durch die Vergabe von Malus-/Minuspunkten zu adressieren ist rechtlich problematisch, weil nur innerhalb von inhaltlich zusammenhängenden Fragen eine Punkteverrechnung stattfinden darf.

Gut geeignete und etablierte Typen sind z.B. „Positive Einfachwahl aus fünf Wahlantworten“ und „Vierfache Entscheidung richtig/falsch (Typ K', genannt Kprim)“, für die wir im Folgenden jeweils drei Beispiele geben. Für Fragen vom Typ Kprim zeigen wir außerdem typische Fehler auf.

Good Practice – Positive Einfachwahl aus fünf Wahlantworten

Beispiel 1

Welche Zeilen in dem folgenden JavaScript Code würden einen Fehler hervorrufen?

```
function runMe() {
  if (true) {
    var v1 = "Go ";
    let v2 = "for ";
    v3 = "it!";
  }
  console.log(v1);
  console.log(v2);
}
runMe();
console.log(v1);
console.log(v3);
```

Wählen Sie eine Antwort:

- ○ keine
- ○ 7, 8, 11, 12
- ◉ 8, 11
- ○ 5, 7, 8, 11, 12
- ○ 11, 12

[Hier geht es um die verschiedenen Arten der Variablendeklarationen (let, var, ohne keyword) in JavaScript und den Effekt davon (block scope, function scope, global scope).]

Beispiel 2

Der folgende Java Code soll alle Elemente aus dem `int` Array `arr` mit `n` Elementen nach links schieben, wobei die herausgeschobene Zahl am rechten Ende wieder eingefügt wird. Ein Array mit den Werten 7, 3, 8, 1, 0, 5 wird somit zu 3, 8, 1, 0, 5, 7 .

Entscheiden Sie, wie `<Statement1>` und `<Statement2>` gewählt werden sollten, damit der Code korrekt ausgeführt wird.

```
<Statement1>
for (int i = 0; i < n - 1; i++)
  arr[i] = arr[i + 1];
<Statement2>
```

Wählen Sie eine Antwort:

- ◉
```
int tmp = arr[0];
arr[n-1] = tmp;
```
- ○
```
int tmp = arr[0];
arr[0] = tmp;
```
- ○
```
int tmp = arr[n-1];
arr[0] = tmp;
```
- ○
```
int tmp = arr[n-1]
arr[n-1] = tmp;
```
- ○
```
int tmp = arr[0];
arr[n] = tmp;
```

[Hier geht es um Programmverständnis.]

Beispiel 3

In welcher Reihenfolge werden die Zahlen durch die Alerts im folgenden JavaScript Code ausgegeben?

```
let p = new Promise(resolve => setTimeout(() => resolve(1), 500));
let r = p.then(result => {
  alert(result);
  return result * 2;
}).then(result => {
  alert(result);
  return result * 2;
});
p.then(result => alert(result));
r.then(result => alert(result));
alert(0);
```

Wählen Sie eine Antwort:

- ◉ 0 1 1 2 4
- ○ 1 2 1 1 0
- ○ 1 2 1 4 0
- ○ 1 2 4 8 0
- ○ 0 1 2 4 8

[Hier geht es um das Prinzip der Ausführungswarteschlange in JavaScript (im Browser) und die Ausführungsreihenfolge im Zusammenhang mit asynchroner Programmierung mittels Promises.]

Good Practice – Kprim Fragen

Um Kprim (K`) Fragen richtig zu bewerten, müssen die Fragen sowie auch die Antworten richtig formuliert werden. Hierzu gibt René Krebs entsprechende Hinweise auf S. 14 in seiner „Anleitung zur Herstellung von MC-Fragen und MC-Prüfungen für die ärztliche Ausbildung“. Weiterhin gibt René Krebs folgendes Kriterium:

“Inhaltlich betrachtet ist der Typ K` angezeigt, wenn es um einen Sachverhalt geht, bei dem mehrere Aspekte bedeutsam sein können, resp. ein Problem, zu dessen richtiger Lösung mehrere Elemente gehören können. Alle Antworten müssen schwarz/weiß beurteilbar sein. Der Typ K` sollte **nicht missbraucht** werden, um völlig heterogene Aussagen zu einem breiten Thema in einem Item zusammenzuwürfeln.”

Bei Kprim ist eine sinnvolle Bewertung wie folgt: Um die volle Punktzahl zu erhalten, müssen alle vier Beurteilungen korrekt sein. Drei korrekte Beurteilungen werden mit der Hälfte der Punkte bewertet.

Beispiel 1

Gegeben sei die folgender PHP Code:

```
function test($str) {
    $i = 0;
    $j = strlen($str) - 1;
    $response = true;
    while ($i < $j) {
        if ($str[$i] != $str[$j])
            $response = false;
        $i++;
        $j--;
    }
    return $response;
}
```

Welche der folgenden Aussagen sind korrekt?

- ☐ Der Rückgabewert ist false/falsch, wenn die Länge des Eingabestring ungerade ist.
- ☒ Für eine Eingabe der Länge 7 wird die while Schleife 3 Mal durchlaufen.
- ☒ Für die Eingabe „madam“ ist der Rückgabewert true/wahr.
- ☐ Bei Eingabe eines leeren Strings ist der Rückgabewert false/falsch.

[Hier geht es um Programmverständnis. Die Aussagen stehen in engem inhaltlichem Zusammenhang mit der gegebenen Funktion.]

Beispiel 2

Gegeben sei folgendes HTML Fragment:

```
<!DOCTYPE html>
<html>
  <head></head>
  <body>
    <header>Wichtige Webseite!</header>
    <main>
      <span class="msg" id="motd"></span>
    </main>
    <footer>Powered by Web Engineering</footer>
  </body>
</html>
```

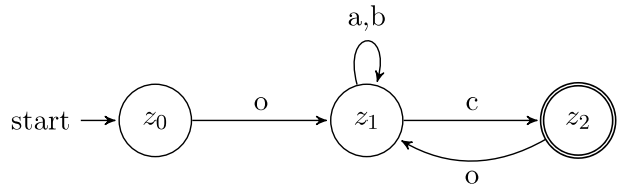
Wählen Sie alle JavaScript Fragmente aus, die den Text "Eat more veggies!" in Fettdruck in das span Tag einfügen.

- ☒ `document.getElementsByClassName("msg")[0].innerHTML = "<b>Eat more Veggies!</b>";`
- ☐ `document.getElementById("motd").textContent = "<b>Eat more Veggies!</b>";`
- ☒ `document.getElementsByTagName("span")[0].innerHTML = "<b>Eat more Veggies!</b>";`
- ☐ `document.getElementsByClassName("msg").innerHTML = "<b>Eat more Veggies!</b>";`

[Hier geht es um die Auswahl und die programmatische Veränderung von HTML Elementen und den Unterschied zwischen `innerHTML` (triggert HTML Rendering/Interpretation) und `textConent` (keine HTML Interpretation).]

Beispiel 3

Gegeben sei ein Automat $A = (Z, \Sigma, \delta, z_0, T)$ mit der Zustandsmenge $Z = \{z_0, z_1, z_2\}$, dem Eingabealphabet $\Sigma = \{a, b, c, o\}$, dem Startzustand z_0 und dem Endzustand z_2 (also $T = \{z_2\}$). Die Zustandsübergangsfunktion δ ergibt sich aus der folgenden grafischen Darstellung:



Wählen Sie aus, welche der folgenden Worte zu der von diesem Automaten akzeptierten Sprache gehören.

- ☐ oa,bc
- ☒ oc
- ☒ oabcoacobbc
- ☐ oao

[Eine einfache Frage zu der von einem Automaten akzeptierten Sprache.]

Bad Practice – Kprim

Kprim Fragen bringen typische Fallstricke mit sich, für die wir mit den folgenden Beispielen sensibilisieren möchten.

Beispiel 1

Welche Aussagen über Linux sind korrekt:

- ☒ Ist Open Source.
- ☐ Kann nur im Terminal bedient werden.
- ☒ Wurde von Linus Torwalds erfunden.
- ☐ Ist auf Desktops weiter verbreitet als Windows.

[Die Fragestellung ist problematisch, da es sich um ein zu breites Themengebiet handelt.]

Beispiel 2

Wählen Sie aus, welche Zeilen zulässigen Java Code darstellen:

- ☒ `public sealed class Animal permits Dog, Monkey, Leopard {...}`
- ☐ `System.out.println(“Hello World”)`
- ☐ `std::cout << "Hello World";`
- ☒
```
String html = """
                  <html>
                         <body>
                             <p>Hello, world</p>
                         </body>
                    </html>
                    """;
```

[Die Fragestellung ist problematisch, weil zum einen das Themenfeld wieder zu breit angelegt ist und zum anderen, weil ohne Angabe der Java Version, die Antworten nicht eindeutig falsch oder richtig sind, so wurden sealed classes erst mit Java 15 eingeführt.]

Beispiel 3

Was trifft normalerweise nicht auf die Worstcase-Laufzeit von Heapsort zu?

- ☒ O(n)
- ☐ O(n log(n))
- ☐ O(n!)
- ☒ O(log(n))

[Die Fragestellung ist aus zwei Gründen problematisch: Der Zusatz „normalerweise“ verfälscht eine sonst sehr präzise formulierte Frage. Fragen sollten ohne Verneinung formuliert werden, da das Verständnis von verneinten Aussagen schwerer ist und so nicht nur das reine Wissen dazu beträgt, ob die Frage richtig beantwortet werden kann.]