

1. Was ist der Wert des Ausdrucks $(1 + 2)$ in der BSL?
 1. Der Wert des Ausdrucks ist 1.
 2. Der Wert des Ausdrucks ist 2.
 3. Der Wert des Ausdrucks ist 3.
 4. Der Wert des Ausdrucks ist "12".
 5. Der Ausdruck ist fehlerhaft: Zu viele Klammern.
 6. Der Ausdruck ist fehlerhaft: + erwartet mindestens 3 Argumente.
 7. Der Ausdruck ist fehlerhaft: 1 ist keine Funktion.
 8. Der Ausdruck ist fehlerhaft: 2 ist keine Funktion.

2. Was ist der Wert des Ausdrucks $(* (+ 2 3) (+ 4 5))$ in der BSL?
 1. Der Wert des Ausdrucks ist 23.
 2. Der Wert des Ausdrucks ist "23".
 3. Der Wert des Ausdrucks ist 45.
 4. Der Wert des Ausdrucks ist "45".
 5. Der Wert des Ausdrucks ist 2345.
 6. Der Wert des Ausdrucks ist "2345".
 7. Der Wert des Ausdrucks ist (list 2 3 4 5).
 8. Der Wert des Ausdrucks ist (list "23" "45").
 9. Der Ausdruck ist fehlerhaft: Zu viele Klammern.
 10. Der Ausdruck ist fehlerhaft: 2 ist keine Funktion.
 11. Der Ausdruck ist fehlerhaft: 3 ist keine Funktion.
 12. Der Ausdruck ist fehlerhaft: 4 ist keine Funktion.
 13. Der Ausdruck ist fehlerhaft: 5 ist keine Funktion.
 14. Der Ausdruck ist fehlerhaft: + erwartet mindestens 3 Argumente.
 15. Der Ausdruck ist fehlerhaft: * erwartet mindestens 3 Argumente.

3. Gegeben sei folgendes Programm in der ISL mit lambda:


```
(define x 12)
```

```
(define f
  (local [(define x 13)]
    (lambda (y) (+ x y))))
```

Was ist der Wert des Ausdrucks $(\text{local}[(\text{define } x \ 14)](f \ x))$ bezüglich diesen Programmes?

1. Der Wert des Ausdrucks ist 12.
2. Der fehlerhaft: x ist nicht definiert.
3. Das Ausdruck ist Wert des Ausdrucks ist 13.
4. Der Wert des Ausdrucks ist 24.
5. Der Wert des Ausdrucks ist 25.
6. Der Wert des Ausdrucks ist 26.
7. Der Wert des Ausdrucks ist 27.
8. Der Wert des Ausdrucks ist 28.

9. Das Programm ist fehlerhaft: f ist doppelt definiert.
10. Das Programm ist fehlerhaft: x ist doppelt definiert.
11. Das Programm ist fehlerhaft: y ist doppelt definiert.
12. Das Programm ist fehlerhaft: f ist nicht definiert.
13. Das Programm ist fehlerhaft: x ist nicht definiert.
14. Das Programm ist fehlerhaft: y ist nicht definiert.
15. Das Ausdruck ist fehlerhaft: f ist doppelt definiert.
16. Das Ausdruck ist fehlerhaft: x ist doppelt definiert.
17. Das Ausdruck ist fehlerhaft: y ist doppelt definiert.
18. Das Ausdruck ist fehlerhaft: f ist nicht definiert.
19. Das Ausdruck ist fehlerhaft: y ist nicht definiert.

4. Wozu reduziert der Ausdruck $(\text{cond}[\text{false } 3][\text{false } 4])$ in der BSL in einem Schritt?

1. $(\text{cond}[\text{false } 3][\text{false } 4]) \rightarrow (\text{cond}[\text{false } 3][\text{false } 4])$
2. $(\text{cond}[\text{false } 3][\text{false } 4]) \rightarrow (\text{cond}[\text{false } 3][\text{true } 4])$
3. $(\text{cond}[\text{false } 3][\text{false } 4]) \rightarrow (\text{cond}[\text{false } 3][\text{else } 4])$
4. $(\text{cond}[\text{false } 3][\text{false } 4]) \rightarrow (\text{cond}[\text{false } 3] \ 4)$
5. $(\text{cond}[\text{false } 3][\text{false } 4]) \rightarrow (\text{cond}[\text{false } \text{false}][\text{false } 4])$
6. $(\text{cond}[\text{false } 3][\text{false } 4]) \rightarrow (\text{cond}[\text{false } \text{false}][\text{true } 4])$
7. $(\text{cond}[\text{false } 3][\text{false } 4]) \rightarrow (\text{cond}[\text{false } \text{false}][\text{else } 4])$
8. $(\text{cond}[\text{false } 3][\text{false } 4]) \rightarrow (\text{cond}[\text{false } \text{false}] \ 4)$
9. $(\text{cond}[\text{false } 3][\text{false } 4]) \rightarrow (\text{cond}[\text{false } 4])$
10. $(\text{cond}[\text{false } 3][\text{false } 4]) \rightarrow (\text{cond}[\text{true } 4])$
11. $(\text{cond}[\text{false } 3][\text{false } 4]) \rightarrow (\text{cond}[\text{else } 4])$
12. $(\text{cond}[\text{false } 3][\text{false } 4]) \rightarrow (\text{cond } 4)$
13. $(\text{cond}[\text{false } 3][\text{false } 4]) \rightarrow 4$
14. $(\text{cond}[\text{false } 3][\text{false } 4])$ ist ein Wert und kann nicht weiter reduziert werden.
15. $(\text{cond}[\text{false } 3][\text{false } 4])$ enthält einen Fehler und kann nicht weiter reduziert werden.

5. Was ist die Signatur der folgenden Funktion?

```
(define (f x y)
  (string-append (number->string x) y))
```

1. Die Signatur ist Number Number -> Number
2. Die Signatur ist Number Number -> String
3. Die Signatur ist Number String -> Number
4. Die Signatur ist Number String -> String
5. Die Signatur ist String Number -> Number
6. Die Signatur ist String Number -> String
7. Die Signatur ist String String -> Number
8. Die Signatur ist String String -> String