True or false? (unless otherwise specified.) Assume any explicitly named package is installed. An expression that raises an error does not evaluate to `true`.

Answer grid

	1	2	3	4	5	6	7	8	9	10	11	12	13	14	15
True	☐	☐	☐	☐	☐	☐	☐	☐	☐	☐	☐	☐	☐	☐	☐
False	☐	☐	☐	☐	☐	☐	☐	☐	☐	☐	☐	☐	☐	☐	☐

1. The expression `"ab" * "cd" * "!"` returns the string `"abcd!"`.

2. The following code produces a single plot containing both curves, rather than replacing the first curve with the second.

```
using Plots
plot(x -> x^2, xlims=(-1, 1), label="square")
plot!(x -> x^3, label="cube")
```

3. The expression `eps()/2 + eps()/2 == eps()` evaluates to `true`.

4. The expression `typeof(1 + 2.0) == Float64` evaluates to `true`.

5. If `u = [1, 2]` and `v = [3, 4]`, then `[u v]` returns the matrix `[1 2; 3 4]`.

6. If `A = [1 2 3; 4 5 6]`, then `A[2, 3]` returns `6`.

7. The following code assigns the vector `[2, 5]` to `w`.

```
v = [-3, 2, 0, 5, -1]
w = v[v .> 0]
```

8. The following code assigns the vector `[1, 3, 5]` to `a`.

```
f(x, y) = x - 2y
a = f.([3, 5, 7], 1)
```

9. The expression `sum(length, ["Julia", "is", "fun"])` returns `3`.

10. The command `A = [i + 2j for i in 1:2, j in 1:3]` produces the matrix `A = [3 5 7; 4 6 8]`.

11. After the following definitions, `f(f(1))` returns `2.0`.

```
f(x::Float64) = 1
f(x::Int) = 2.0
```

12. The call `odd_sum(5)` returns `9`.

```
function odd_sum(n)
    s = 0
    for k in 1:n
        if isodd(k)
            s += k
        end
    end
    return s
end
```

13. The expression `LinRange(0, 1, 4) == [0, 0.2, 0.4, 0.6, 0.8, 1]` evaluates to `true`.

14. With the following definition, the expression `f(n)` returns the n-th Fibonacci number for positive integers n (assuming no integer overflow). Recall that the Fibonacci numbers are defined as $f_1 = f_2 = 1$ and $f_k = f_{k-1} + f_{k-2}$ for $k \geq 3$.

```
f(n) = n <= 2 ? 1 : f(n-1) + f(n-2)
```

15. If `v = [2, 4, 6, 8]`, then `v[2:end] == [2, 4, 6]` evaluates to `true`.

Bonus (2 points). Give the values of `a` and `b` after the following code, and briefly explain why the results differ.

```
v = [[1, 2, 3], [2, 3, 4], [4, 5, 6]]
a = sum.(v)
b = sum(v)
```

Answer and explanation:

Answer key

Questions 1–15: 1 point for a correct answer, 0 otherwise.

1. **True.** `*` concatenates strings. The result is `"abcd!"`.

2. **True.** `plot!` adds a series to the current plot, which already contains the square function.

3. **True.** For `Float64`, `eps()` is 2^{-52} and `eps()/2` is 2^{-53}. Both are exactly representable, and adding the two halves gives `eps()` exactly.

4. **True.** The integer is promoted for this addition; the result is `3.0`, of type `Float64`.

5. **False.** Horizontal concatenation places `u` and `v` in the columns, giving `[1 3; 2 4]`.

6. **True.** The first index selects the row and the second selects the column. The entry in row 2, column 3 is 6.

7. **True.** The mask selects the strictly positive entries, giving `[2, 5]`.

8. **True.** The scalar second argument is used in each call: `f(3, 1)`, `f(5, 1)`, and `f(7, 1)`.

9. **False.** The expression sums the string lengths, giving $5 + 2 + 3 = 10$, rather than counting the strings.

10. **True.** Evaluating `i + 2j` for each row index `i` and column index `j` gives `[3 5 7; 4 6 8]`.

11. **False.** The inner call `f(1)` returns `2.0`. The outer call therefore uses the `Float64` method and returns `1`.

12. **True.** The accumulator adds the odd integers $1 + 3 + 5 = 9$.

13. **False.** `LinRange(0, 1, 4)` has four equally spaced entries, including both endpoints. The vector in the question has six entries instead.

14. **True.** The ternary expression returns 1 for the base cases $n = 1$ and $n = 2$. For $n \geq 3$, the recursive calls implement the Fibonacci recurrence $f_n = f_{n-1} + f_{n-2}$.

15. **False.** `end` denotes the last index, which is 4. The slice selects entries 2 through 4, giving `[4, 6, 8]`.

Bonus. `a = [6, 9, 15]` and `b = [7, 10, 13]`. The broadcast `sum.(v)` applies `sum` to each inner vector independently, returning the vector of the sums 6, 9 and 15. In contrast, `sum(v)` adds the three vectors together with the `+` operator, i.e. `[1,2,3] + [2,3,4] + [4,5,6]`, yielding the single vector `[7, 10, 13]`. Award 0.5 points for each correct final value and 0.5 points for each correct explanation (elementwise sum versus vector addition).