FLOATING-POINT CONSTANTS (4/4 points)

What is the result of compiling and evaluating `1.5 *. 1e3` ?

- ○ Syntax error.
- ○ Type error.
- ● `1500.` ✓
- ○ `1500`

- The `*.` operator is a floating-point multiplication. Floating-point constants must contain a dot (e.g. `1.5`), an exponential part (e.g. `1e3`), or both: `1.5e3`.
- The OCaml toplevel always prints floating-point values with the dot notation.

What is the result of compiling and evaluating `1.5 *. 1000.` ?

- ○ Syntax error.
- ○ Type error.
- ● `1500.` ✓
- ○ `1500`

The expression `1000.`, with a final dot, is a floating-point constant.

What is the result of compiling and evaluating `1.5 *. 1000` ?

○ Syntax error.

● Type error. ✓

○ `1500.`

○ `1500`

The expression `1000`, without a final dot, is an integer constant ; it cannot be used as argument for a floating-point multiplication, as there is no implicit type conversion in OCaml.

What is the result of compiling and evaluating `1.5 *. "1e3"`?

○ Syntax error.

● Type error. ✓

○ `1500.`

○ `1500`

There is no implicit type conversion in OCaml. The expression `"1e3"` is a string, it cannot be used as argument for floating-point multiplication.

Vous avez utilisé 1 essais sur 3

FLOATING-POINT CONSTANTS (BIS) (4/4 points)

What is the result of compiling and evaluating `1.5 * 1000.`?

○ Syntax error.

● Type error. ✓

- ○ `1500.`
- ○ `1500`

The `*` operator is the integer multiplication, it cannot accept floating-point values as argument.

What is the result of compiling and evaluating `1.5e3` ?

- ○ Syntax error.
- ○ Type error.
- [✓ `1500.`] ✓
- ○ `1500`

The expression `1.5e3` is a valid floating-point constant.

What is the result of compiling and evaluating `1000. +. 500. /. 2.` ?

- ○ Syntax error.
- ○ Type error.
- ○ `750.`
- [✓ `1250.`] ✓

The same usual priorities apply also with floating-point operators.

What is the result of compiling and evaluating `1000.+.500. /. 2.` ?

- ○ Syntax error.
- ○ Type error.
- ○ `750.`
- ● `1250.` ✓

This is read as `1000. +. ( 500. /. 2. )`, the spacing has no influence on operator priorities.

Vous avez utilisé 1 essais sur 3

COMPARISON EXPRESSIONS (4/4 points)

What is the result of compiling and evaluating `1.5e3 <= 1500. && 1500 <= 1500 && false <> true` ?

- ○ Syntax error.
- ○ Type error.
- ○ `false`
- ● `true` ✓

The polymorphic comparison operators are able to compare floating-point values, as well as integer or boolean values.

What is the result of compiling and evaluating `1500 < 1500.1` ?

- ○ Syntax error.

[✓ Type error.] ✓

`false`

`true`

You can't compare an integer value and a floating-point value without an explicit type conversion.

What is the result of compiling and evaluating `1500 < int_of_float 1500.1`?

Syntax error.

Type error.

[✓ `false`] ✓

`true`

The `int_of_float` function truncates its floating-point argument and returns the resulting integer. This expression is equivalent to `1500 < 1500`.

What is the result of compiling and evaluating `floor 1500.1 = 1500`?

Syntax error.

[✓ Type error.] ✓

`false`

`true`

According to the OCaml manual, the `floor` function, is a function that takes a float and returns a float.

Vous avez utilisé 1 essais sur 3

FLOATING-POINT EXPRESSIONS (4/4 points)

Warning: you only have 1 attempt (but anyway the result will not count in the final grading).

What is the result of compiling and evaluating `10. /. 3. *. 3.` ?

- ○ Syntax error.
- ○ Type error.
- ○ `9`
- ○ `9.`
- ○ `10`
- [✓ ● `10.`] ✓

This is read as `(10. /. 3.) *. 3.` .

What is the result of compiling and evaluating `10./.3.*.3.` ?

- ○ Syntax error.
- ○ Type error.
- ○ `9`
- ○ `9.`

- ○ `10`
- [✓ `10.`] ✓

This is read as `(10. /. 3.) *. 3.` .

What is the result of compiling and evaluating `sqrt 16. +. 9.` ?

- ○ Syntax error.
- ○ Type error.
- ○ `5`
- ○ `5.`
- ○ `13`
- [✓ `13.`] ✓

This is read as `(sqrt 16.) +. 9.` .

What is the result of compiling and evaluating `sqrt 16.+.9.` ?

- ○ Syntax error.
- ○ Type error.
- ○ `5`
- ○ `5.`
- ○ `13`

[✓ `13.`] ✓

This is read as `(sqrt 16.) +. 9.` . The spacing has no influence on the priority between operators.

Vous avez utilisé 1 essais sur 1

BOOLEAN EXPRESSIONS (4/4 points)

Warning: you only have 1 attempt (but anyway the result will not count in the final grading).

What is the result of compiling and evaluating `1. <> 2.5 && 3 <> 4` ?

- Syntax error.
- Type error.
- `false`
- [✓ `true`] ✓

Business as usual.

What is the result of compiling and evaluating `not 1. = 2. || not 3 = 4` ?

- Syntax error.
- [✓ Type error.] ✓
- `false`
- `true`

This is read as `((not 1.) = 2.) || ((not 3) = 4)` -, where the sub-expression `(not 1.)` and `(not 3)` are ill-typed.

What is the result of compiling and evaluating `1 <= 2.5 && 3 <= 4.5` ?

- ○ Syntax error.
- ● Type error. ✓
- ○ `false`
- ○ `true`

What is the result of compiling and evaluating
`1 <= int_of_float 2.5 && 3. <= floor 3.5` ?

- ○ Syntax error.
- ○ Type error.
- ○ `false`
- ● `true` ✓

The expression `int_of_float 2.5` evaluates to the integer value `2M`; and the expression `floor 3.5` evaluates to the floating-point value `3.`.

Vous avez utilisé 1 essais sur 1