

The *Integers* module is about integers, and n/p is not, in general, an integer. The only arithmetic operations you learned in grade school that the module defines are addition (+), subtraction (-), multiplication (*), and exponentiation (a^b is typed `a^b`). There is a *Reals* module that defines ordinary division, but it is rarely used because the TLC model checker cannot evaluate the operator /. So, we define *Divides* using the operators defined in the *Integers* module.

?

←

→

C

I

S

The *Integers* module is about integers, and n/p is not, in general, an integer. The only arithmetic operations you learned in grade school that the module defines are addition (+), subtraction (-), multiplication (*), and exponentiation (a^b is typed `a^b`). There is a *Reals* module that defines ordinary division, but it is rarely used because the TLC model checker cannot evaluate the operator /. So, we define *Divides* using the operators defined in the *Integers* module.

red: .3

The *Integers* module is about integers, and n/p is not, in general, an integer. The only arithmetic operations you learned in grade school that the module defines are addition (+), subtraction (-), multiplication (*), and exponentiation (a^b is typed `a^b`). There is a *Reals* module that defines ordinary division, but it is rarely used because the TLC model checker cannot evaluate the operator /. So, we define *Divides* using the operators defined in the *Integers* module.

The *Integers* module is about integers, and n/p is not, in general, an integer. The only arithmetic operations you learned in grade school that the module defines are addition (+), subtraction (-), multiplication (*), and exponentiation (a^b is typed `a^b`). There is a *Reals* module that defines ordinary division, but it is rarely used because the TLC model checker cannot evaluate the operator /. So, we define *Divides* using the operators defined in the *Integers* module.

green: .3

The *Integers* module is about integers, and n/p is not, in general, an integer. The only arithmetic operations you learned in grade school that the module defines are addition (+), subtraction (-), multiplication (*), and exponentiation (a^b is typed `a^b`). There is a *Reals* module that defines ordinary division, but it is rarely used because the TLC model checker cannot evaluate the operator /. So, we define *Divides* using the operators defined in the *Integers* module.

The *Integers* module is about integers, and n/p is not, in general, an integer. The only arithmetic operations you learned in grade school that the module defines are addition (+), subtraction (-), multiplication (*), and exponentiation (a^b is typed `a^b`). There is a *Reals* module that defines ordinary division, but it is rarely used because the TLC model checker cannot evaluate the operator /. So, we define *Divides* using the operators defined in the *Integers* module.

blue: .3

The *Integers* module is about integers, and n/p is not, in general, an integer. The only arithmetic operations you learned in grade school that the module defines are addition (+), subtraction (-), multiplication (*), and exponentiation (a^b is typed `a^b`). There is a *Reals* module that defines ordinary division, but it is rarely used because the TLC model checker cannot evaluate the operator /. So, we define *Divides* using the operators defined in the *Integers* module.

?

←

→

C

I

S

The *Integers* module is about integers, and n/p is not, in general, an integer. The only arithmetic operations you learned in grade school that the module defines are addition (+), subtraction (-), multiplication (*), and exponentiation (a^b is typed `a^b`). There is a *Reals* module that defines ordinary division, but it is rarely used because the TLC model checker cannot evaluate the operator /. So, we define *Divides* using the operators defined in the *Integers* module.

aqua: .3

The *Integers* module is about integers, and n/p is not, in general, an integer. The only arithmetic operations you learned in grade school that the module defines are addition (+), subtraction (-), multiplication (*), and exponentiation (a^b is typed `a^b`). There is a *Reals* module that defines ordinary division, but it is rarely used because the TLC model checker cannot evaluate the operator /. So, we define *Divides* using the operators defined in the *Integers* module.

The *Integers* module is about integers, and n/p is not, in general, an integer. The only arithmetic operations you learned in grade school that the module defines are addition (+), subtraction (-), multiplication (*), and exponentiation (a^b is typed `a^b`). There is a *Reals* module that defines ordinary division, but it is rarely used because the TLC model checker cannot evaluate the operator /. So, we define *Divides* using the operators defined in the *Integers* module.

magenta: .3

The *Integers* module is about integers, and n/p is not, in general, an integer. The only arithmetic operations you learned in grade school that the module defines are addition (+), subtraction (-), multiplication (*), and exponentiation (a^b is typed `a^b`). There is a *Reals* module that defines ordinary division, but it is rarely used because the TLC model checker cannot evaluate the operator /. So, we define *Divides* using the operators defined in the *Integers* module.

The *Integers* module is about integers, and n/p is not, in general, an integer. The only arithmetic operations you learned in grade school that the module defines are addition (+), subtraction (-), multiplication (*), and exponentiation (a^b is typed `a^b`). There is a *Reals* module that defines ordinary division, but it is rarely used because the TLC model checker cannot evaluate the operator /. So, we define *Divides* using the operators defined in the *Integers* module.

yellow: .3

The *Integers* module is about integers, and n/p is not, in general, an integer. The only arithmetic operations you learned in grade school that the module defines are addition (+), subtraction (-), multiplication (*), and exponentiation (a^b is typed `a^b`). There is a *Reals* module that defines ordinary division, but it is rarely used because the TLC model checker cannot evaluate the operator /. So, we define *Divides* using the operators defined in the *Integers* module.

?

←

→

C

I

S

The *Integers* module is about integers, and n/p is not, in general, an integer. The only arithmetic operations you learned in grade school that the module defines are addition (+), subtraction (-), multiplication (*), and exponentiation (a^b is typed `a^b`). There is a *Reals* module that defines ordinary division, but it is rarely used because the TLC model checker cannot evaluate the operator /. So, we define *Divides* using the operators defined in the *Integers* module.

red: .35

The *Integers* module is about integers, and n/p is not, in general, an integer. The only arithmetic operations you learned in grade school that the module defines are addition (+), subtraction (-), multiplication (*), and exponentiation (a^b is typed `a^b`). There is a *Reals* module that defines ordinary division, but it is rarely used because the TLC model checker cannot evaluate the operator /. So, we define *Divides* using the operators defined in the *Integers* module.

The *Integers* module is about integers, and n/p is not, in general, an integer. The only arithmetic operations you learned in grade school that the module defines are addition (+), subtraction (-), multiplication (*), and exponentiation (a^b is typed `a^b`). There is a *Reals* module that defines ordinary division, but it is rarely used because the TLC model checker cannot evaluate the operator /. So, we define *Divides* using the operators defined in the *Integers* module.

green: .35

The *Integers* module is about integers, and n/p is not, in general, an integer. The only arithmetic operations you learned in grade school that the module defines are addition (+), subtraction (-), multiplication (*), and exponentiation (a^b is typed `a^b`). There is a *Reals* module that defines ordinary division, but it is rarely used because the TLC model checker cannot evaluate the operator /. So, we define *Divides* using the operators defined in the *Integers* module.

The *Integers* module is about integers, and n/p is not, in general, an integer. The only arithmetic operations you learned in grade school that the module defines are addition (+), subtraction (-), multiplication (*), and exponentiation (a^b is typed `a^b`). There is a *Reals* module that defines ordinary division, but it is rarely used because the TLC model checker cannot evaluate the operator /. So, we define *Divides* using the operators defined in the *Integers* module.

blue: .35

?
←
→
C
I
S

The *Integers* module is about integers, and n/p is not, in general, an integer. The only arithmetic operations you learned in grade school that the module defines are addition (+), subtraction (−), multiplication (*), and exponentiation (a^b is typed `a^b`). There is a *Reals* module that defines ordinary division, but it is rarely used because the TLC model checker cannot evaluate the operator /. So, we define *Divides* using the operators defined in the *Integers* module.

The *Integers* module is about integers, and n/p is not, in general, an integer. The only arithmetic operations you learned in grade school that the module defines are addition (+), subtraction (−), multiplication (*), and exponentiation (a^b is typed `a^b`). There is a *Reals* module that defines ordinary division, but it is rarely used because the TLC model checker cannot evaluate the operator /. So, we define *Divides* using the operators defined in the *Integers* module.

aqua: .35

The *Integers* module is about integers, and n/p is not, in general, an integer. The only arithmetic operations you learned in grade school that the module defines are addition (+), subtraction (−), multiplication (*), and exponentiation (a^b is typed `a^b`). There is a *Reals* module that defines ordinary division, but it is rarely used because the TLC model checker cannot evaluate the operator /. So, we define *Divides* using the operators defined in the *Integers* module.

The *Integers* module is about integers, and n/p is not, in general, an integer. The only arithmetic operations you learned in grade school that the module defines are addition (+), subtraction (−), multiplication (*), and exponentiation (a^b is typed `a^b`). There is a *Reals* module that defines ordinary division, but it is rarely used because the TLC model checker cannot evaluate the operator /. So, we define *Divides* using the operators defined in the *Integers* module.

magenta: .35

The *Integers* module is about integers, and n/p is not, in general, an integer. The only arithmetic operations you learned in grade school that the module defines are addition (+), subtraction (−), multiplication (*), and exponentiation (a^b is typed `a^b`). There is a *Reals* module that defines ordinary division, but it is rarely used because the TLC model checker cannot evaluate the operator /. So, we define *Divides* using the operators defined in the *Integers* module.

The *Integers* module is about integers, and n/p is not, in general, an integer. The only arithmetic operations you learned in grade school that the module defines are addition (+), subtraction (−), multiplication (*), and exponentiation (a^b is typed `a^b`). There is a *Reals* module that defines ordinary division, but it is rarely used because the TLC model checker cannot evaluate the operator /. So, we define *Divides* using the operators defined in the *Integers* module.

yellow: .35

The *Integers* module is about integers, and n/p is not, in general, an integer. The only arithmetic operations you learned in grade school that the module defines are addition (+), subtraction (-), multiplication (*), and exponentiation (a^b is typed `a^b`). There is a *Reals* module that defines ordinary division, but it is rarely used because the TLC model checker cannot evaluate the operator /. So, we define *Divides* using the operators defined in the *Integers* module.

?

←

→

C

I

S

The *Integers* module is about integers, and n/p is not, in general, an integer. The only arithmetic operations you learned in grade school that the module defines are addition (+), subtraction (-), multiplication (*), and exponentiation (a^b is typed `a^b`). There is a *Reals* module that defines ordinary division, but it is rarely used because the TLC model checker cannot evaluate the operator /. So, we define *Divides* using the operators defined in the *Integers* module.

red: .4

The *Integers* module is about integers, and n/p is not, in general, an integer. The only arithmetic operations you learned in grade school that the module defines are addition (+), subtraction (-), multiplication (*), and exponentiation (a^b is typed `a^b`). There is a *Reals* module that defines ordinary division, but it is rarely used because the TLC model checker cannot evaluate the operator /. So, we define *Divides* using the operators defined in the *Integers* module.

The *Integers* module is about integers, and n/p is not, in general, an integer. The only arithmetic operations you learned in grade school that the module defines are addition (+), subtraction (-), multiplication (*), and exponentiation (a^b is typed `a^b`). There is a *Reals* module that defines ordinary division, but it is rarely used because the TLC model checker cannot evaluate the operator /. So, we define *Divides* using the operators defined in the *Integers* module.

green: .4

The *Integers* module is about integers, and n/p is not, in general, an integer. The only arithmetic operations you learned in grade school that the module defines are addition (+), subtraction (-), multiplication (*), and exponentiation (a^b is typed `a^b`). There is a *Reals* module that defines ordinary division, but it is rarely used because the TLC model checker cannot evaluate the operator /. So, we define *Divides* using the operators defined in the *Integers* module.

The *Integers* module is about integers, and n/p is not, in general, an integer. The only arithmetic operations you learned in grade school that the module defines are addition (+), subtraction (-), multiplication (*), and exponentiation (a^b is typed `a^b`). There is a *Reals* module that defines ordinary division, but it is rarely used because the TLC model checker cannot evaluate the operator /. So, we define *Divides* using the operators defined in the *Integers* module.

blue: .4

The *Integers* module is about integers, and n/p is not, in general, an integer. The only arithmetic operations you learned in grade school that the module defines are addition (+), subtraction (-), multiplication (*), and exponentiation (a^b is typed `a^b`). There is a *Reals* module that defines ordinary division, but it is rarely used because the TLC model checker cannot evaluate the operator /. So, we define *Divides* using the operators defined in the *Integers* module.

?

←

→

C

I

S

The *Integers* module is about integers, and n/p is not, in general, an integer. The only arithmetic operations you learned in grade school that the module defines are addition (+), subtraction (-), multiplication (*), and exponentiation (a^b is typed `a^b`). There is a *Reals* module that defines ordinary division, but it is rarely used because the TLC model checker cannot evaluate the operator /. So, we define *Divides* using the operators defined in the *Integers* module.

aqua: .4

The *Integers* module is about integers, and n/p is not, in general, an integer. The only arithmetic operations you learned in grade school that the module defines are addition (+), subtraction (-), multiplication (*), and exponentiation (a^b is typed `a^b`). There is a *Reals* module that defines ordinary division, but it is rarely used because the TLC model checker cannot evaluate the operator /. So, we define *Divides* using the operators defined in the *Integers* module.

The *Integers* module is about integers, and n/p is not, in general, an integer. The only arithmetic operations you learned in grade school that the module defines are addition (+), subtraction (-), multiplication (*), and exponentiation (a^b is typed `a^b`). There is a *Reals* module that defines ordinary division, but it is rarely used because the TLC model checker cannot evaluate the operator /. So, we define *Divides* using the operators defined in the *Integers* module.

magenta: .4

The *Integers* module is about integers, and n/p is not, in general, an integer. The only arithmetic operations you learned in grade school that the module defines are addition (+), subtraction (-), multiplication (*), and exponentiation (a^b is typed `a^b`). There is a *Reals* module that defines ordinary division, but it is rarely used because the TLC model checker cannot evaluate the operator /. So, we define *Divides* using the operators defined in the *Integers* module.

The *Integers* module is about integers, and n/p is not, in general, an integer. The only arithmetic operations you learned in grade school that the module defines are addition (+), subtraction (-), multiplication (*), and exponentiation (a^b is typed `a^b`). There is a *Reals* module that defines ordinary division, but it is rarely used because the TLC model checker cannot evaluate the operator /. So, we define *Divides* using the operators defined in the *Integers* module.

yellow: .4

The *Integers* module is about integers, and n/p is not, in general, an integer. The only arithmetic operations you learned in grade school that the module defines are addition (+), subtraction (-), multiplication (*), and exponentiation (a^b is typed `a^b`). There is a *Reals* module that defines ordinary division, but it is rarely used because the TLC model checker cannot evaluate the operator /. So, we define *Divides* using the operators defined in the *Integers* module.

?

←

→

C

I

S

The *Integers* module is about integers, and n/p is not, in general, an integer. The only arithmetic operations you learned in grade school that the module defines are addition (+), subtraction (-), multiplication (*), and exponentiation (a^b is typed `a^b`). There is a *Reals* module that defines ordinary division, but it is rarely used because the TLC model checker cannot evaluate the operator /. So, we define *Divides* using the operators defined in the *Integers* module.

red: .45

The *Integers* module is about integers, and n/p is not, in general, an integer. The only arithmetic operations you learned in grade school that the module defines are addition (+), subtraction (-), multiplication (*), and exponentiation (a^b is typed `a^b`). There is a *Reals* module that defines ordinary division, but it is rarely used because the TLC model checker cannot evaluate the operator /. So, we define *Divides* using the operators defined in the *Integers* module.

The *Integers* module is about integers, and n/p is not, in general, an integer. The only arithmetic operations you learned in grade school that the module defines are addition (+), subtraction (-), multiplication (*), and exponentiation (a^b is typed `a^b`). There is a *Reals* module that defines ordinary division, but it is rarely used because the TLC model checker cannot evaluate the operator /. So, we define *Divides* using the operators defined in the *Integers* module.

green: .45

The *Integers* module is about integers, and n/p is not, in general, an integer. The only arithmetic operations you learned in grade school that the module defines are addition (+), subtraction (-), multiplication (*), and exponentiation (a^b is typed `a^b`). There is a *Reals* module that defines ordinary division, but it is rarely used because the TLC model checker cannot evaluate the operator /. So, we define *Divides* using the operators defined in the *Integers* module.

The *Integers* module is about integers, and n/p is not, in general, an integer. The only arithmetic operations you learned in grade school that the module defines are addition (+), subtraction (-), multiplication (*), and exponentiation (a^b is typed `a^b`). There is a *Reals* module that defines ordinary division, but it is rarely used because the TLC model checker cannot evaluate the operator /. So, we define *Divides* using the operators defined in the *Integers* module.

blue: .45

?
←
→
C
I
S

The *Integers* module is about integers, and n/p is not, in general, an integer. The only arithmetic operations you learned in grade school that the module defines are addition (+), subtraction (−), multiplication (*), and exponentiation (a^b is typed `a^b`). There is a *Reals* module that defines ordinary division, but it is rarely used because the TLC model checker cannot evaluate the operator /. So, we define *Divides* using the operators defined in the *Integers* module.

The *Integers* module is about integers, and n/p is not, in general, an integer. The only arithmetic operations you learned in grade school that the module defines are addition (+), subtraction (−), multiplication (*), and exponentiation (a^b is typed `a^b`). There is a *Reals* module that defines ordinary division, but it is rarely used because the TLC model checker cannot evaluate the operator /. So, we define *Divides* using the operators defined in the *Integers* module.

aqua: .45

The *Integers* module is about integers, and n/p is not, in general, an integer. The only arithmetic operations you learned in grade school that the module defines are addition (+), subtraction (−), multiplication (*), and exponentiation (a^b is typed `a^b`). There is a *Reals* module that defines ordinary division, but it is rarely used because the TLC model checker cannot evaluate the operator /. So, we define *Divides* using the operators defined in the *Integers* module.

The *Integers* module is about integers, and n/p is not, in general, an integer. The only arithmetic operations you learned in grade school that the module defines are addition (+), subtraction (−), multiplication (*), and exponentiation (a^b is typed `a^b`). There is a *Reals* module that defines ordinary division, but it is rarely used because the TLC model checker cannot evaluate the operator /. So, we define *Divides* using the operators defined in the *Integers* module.

magenta: .45

The *Integers* module is about integers, and n/p is not, in general, an integer. The only arithmetic operations you learned in grade school that the module defines are addition (+), subtraction (−), multiplication (*), and exponentiation (a^b is typed `a^b`). There is a *Reals* module that defines ordinary division, but it is rarely used because the TLC model checker cannot evaluate the operator /. So, we define *Divides* using the operators defined in the *Integers* module.

The *Integers* module is about integers, and n/p is not, in general, an integer. The only arithmetic operations you learned in grade school that the module defines are addition (+), subtraction (−), multiplication (*), and exponentiation (a^b is typed `a^b`). There is a *Reals* module that defines ordinary division, but it is rarely used because the TLC model checker cannot evaluate the operator /. So, we define *Divides* using the operators defined in the *Integers* module.

yellow: .45

The *Integers* module is about integers, and n/p is not, in general, an integer. The only arithmetic operations you learned in grade school that the module defines are addition (+), subtraction (-), multiplication (*), and exponentiation (a^b is typed `a^b`). There is a *Reals* module that defines ordinary division, but it is rarely used because the TLC model checker cannot evaluate the operator $/$. So, we define *Divides* using the operators defined in the *Integers* module.

The *Integers* module is about integers, and n/p is not, in general, an integer. The only arithmetic operations you learned in grade school that the module defines are addition (+), subtraction (-), multiplication (*), and exponentiation (a^b is typed `a^b`). There is a *Reals* module that defines ordinary division, but it is rarely used because the TLC model checker cannot evaluate the operator /. So, we define *Divides* using the operators defined in the *Integers* module.

red: .5

The *Integers* module is about integers, and n/p is not, in general, an integer. The only arithmetic operations you learned in grade school that the module defines are addition (+), subtraction (-), multiplication (*), and exponentiation (a^b is typed `a^b`). There is a *Reals* module that defines ordinary division, but it is rarely used because the TLC model checker cannot evaluate the operator $/$. So, we define *Divides* using the operators defined in the *Integers* module.

The *Integers* module is about integers, and n/p is not, in general, an integer. The only arithmetic operations you learned in grade school that the module defines are addition (+), subtraction (-), multiplication (*), and exponentiation (a^b is typed `a^b`). There is a *Reals* module that defines ordinary division, but it is rarely used because the TLC model checker cannot evaluate the operator /. So, we define *Divides* using the operators defined in the *Integers* module.

```
green: .5
```

The *Integers* module is about integers, and n/p is not, in general, an integer. The only arithmetic operations you learned in grade school that the module defines are addition (+), subtraction (-), multiplication (*), and exponentiation (a^b is typed `a^b`). There is a *Reals* module that defines ordinary division, but it is rarely used because the TLC model checker cannot evaluate the operator $/$. So, we define *Divides* using the operators defined in the *Integers* module.

The *Integers* module is about integers, and n/p is not, in general, an integer. The only arithmetic operations you learned in grade school that the module defines are addition (+), subtraction (−), multiplication (*), and exponentiation (a^b is typed `a^b`). There is a *Reals* module that defines ordinary division, but it is rarely used because the TLC model checker cannot evaluate the operator $/$. So, we define *Divides* using the operators defined in the *Integers* module.

blue: .5

The *Integers* module is about integers, and n/p is not, in general, an integer. The only arithmetic operations you learned in grade school that the module defines are addition (+), subtraction (-), multiplication (*), and exponentiation (a^b is typed `a^b`). There is a *Reals* module that defines ordinary division, but it is rarely used because the TLC model checker cannot evaluate the operator /. So, we define *Divides* using the operators defined in the *Integers* module.

The *Integers* module is about integers, and n/p is not, in general, an integer. The only arithmetic operations you learned in grade school that the module defines are addition (+), subtraction (-), multiplication (*), and exponentiation (a^b is typed `a^b`). There is a *Reals* module that defines ordinary division, but it is rarely used because the TLC model checker cannot evaluate the operator /. So, we define *Divides* using the operators defined in the *Integers* module.

aqua: .5

The *Integers* module is about integers, and n/p is not, in general, an integer. The only arithmetic operations you learned in grade school that the module defines are addition (+), subtraction (-), multiplication (*), and exponentiation (a^b is typed `a^b`). There is a *Reals* module that defines ordinary division, but it is rarely used because the TLC model checker cannot evaluate the operator /. So, we define *Divides* using the operators defined in the *Integers* module.

The *Integers* module is about integers, and n/p is not, in general, an integer. The only arithmetic operations you learned in grade school that the module defines are addition (+), subtraction (-), multiplication (*), and exponentiation (a^b is typed `a^b`). There is a *Reals* module that defines ordinary division, but it is rarely used because the TLC model checker cannot evaluate the operator /. So, we define *Divides* using the operators defined in the *Integers* module.

magenta: .5

The *Integers* module is about integers, and n/p is not, in general, an integer. The only arithmetic operations you learned in grade school that the module defines are addition (+), subtraction (-), multiplication (*), and exponentiation (a^b is typed `a^b`). There is a *Reals* module that defines ordinary division, but it is rarely used because the TLC model checker cannot evaluate the operator /. So, we define *Divides* using the operators defined in the *Integers* module.

The *Integers* module is about integers, and n/p is not, in general, an integer. The only arithmetic operations you learned in grade school that the module defines are addition (+), subtraction (-), multiplication (*), and exponentiation (a^b is typed `a^b`). There is a *Reals* module that defines ordinary division, but it is rarely used because the TLC model checker cannot evaluate the operator /. So, we define *Divides* using the operators defined in the *Integers* module.

yellow: .5

?

←

→

C

I

S

The *Integers* module is about integers, and n/p is not, in general, an integer. The only arithmetic operations you learned in grade school that the module defines are addition (+), subtraction (-), multiplication (*), and exponentiation (a^b is typed `a^b`). There is a *Reals* module that defines ordinary division, but it is rarely used because the TLC model checker cannot evaluate the operator /. So, we define *Divides* using the operators defined in the *Integers* module.

?

←

→

C

I

S

The *Integers* module is about integers, and n/p is not, in general, an integer. The only arithmetic operations you learned in grade school that the module defines are addition (+), subtraction (-), multiplication (*), and exponentiation (a^b is typed `a^b`). There is a *Reals* module that defines ordinary division, but it is rarely used because the TLC model checker cannot evaluate the operator /. So, we define *Divides* using the operators defined in the *Integers* module.

red: .55

The *Integers* module is about integers, and n/p is not, in general, an integer. The only arithmetic operations you learned in grade school that the module defines are addition (+), subtraction (-), multiplication (*), and exponentiation (a^b is typed `a^b`). There is a *Reals* module that defines ordinary division, but it is rarely used because the TLC model checker cannot evaluate the operator /. So, we define *Divides* using the operators defined in the *Integers* module.

The *Integers* module is about integers, and n/p is not, in general, an integer. The only arithmetic operations you learned in grade school that the module defines are addition (+), subtraction (-), multiplication (*), and exponentiation (a^b is typed `a^b`). There is a *Reals* module that defines ordinary division, but it is rarely used because the TLC model checker cannot evaluate the operator /. So, we define *Divides* using the operators defined in the *Integers* module.

green: .55

The *Integers* module is about integers, and n/p is not, in general, an integer. The only arithmetic operations you learned in grade school that the module defines are addition (+), subtraction (-), multiplication (*), and exponentiation (a^b is typed `a^b`). There is a *Reals* module that defines ordinary division, but it is rarely used because the TLC model checker cannot evaluate the operator /. So, we define *Divides* using the operators defined in the *Integers* module.

The *Integers* module is about integers, and n/p is not, in general, an integer. The only arithmetic operations you learned in grade school that the module defines are addition (+), subtraction (-), multiplication (*), and exponentiation (a^b is typed `a^b`). There is a *Reals* module that defines ordinary division, but it is rarely used because the TLC model checker cannot evaluate the operator /. So, we define *Divides* using the operators defined in the *Integers* module.

blue: .55

?
←
→
C
I
S

The *Integers* module is about integers, and n/p is not, in general, an integer. The only arithmetic operations you learned in grade school that the module defines are addition (+), subtraction (−), multiplication (*), and exponentiation (a^b is typed `a^b`). There is a *Reals* module that defines ordinary division, but it is rarely used because the TLC model checker cannot evaluate the operator /. So, we define *Divides* using the operators defined in the *Integers* module.

The *Integers* module is about integers, and n/p is not, in general, an integer. The only arithmetic operations you learned in grade school that the module defines are addition (+), subtraction (−), multiplication (*), and exponentiation (a^b is typed `a^b`). There is a *Reals* module that defines ordinary division, but it is rarely used because the TLC model checker cannot evaluate the operator /. So, we define *Divides* using the operators defined in the *Integers* module.

aqua: .55

The *Integers* module is about integers, and n/p is not, in general, an integer. The only arithmetic operations you learned in grade school that the module defines are addition (+), subtraction (−), multiplication (*), and exponentiation (a^b is typed `a^b`). There is a *Reals* module that defines ordinary division, but it is rarely used because the TLC model checker cannot evaluate the operator /. So, we define *Divides* using the operators defined in the *Integers* module.

The *Integers* module is about integers, and n/p is not, in general, an integer. The only arithmetic operations you learned in grade school that the module defines are addition (+), subtraction (−), multiplication (*), and exponentiation (a^b is typed `a^b`). There is a *Reals* module that defines ordinary division, but it is rarely used because the TLC model checker cannot evaluate the operator /. So, we define *Divides* using the operators defined in the *Integers* module.

magenta: .55

The *Integers* module is about integers, and n/p is not, in general, an integer. The only arithmetic operations you learned in grade school that the module defines are addition (+), subtraction (−), multiplication (*), and exponentiation (a^b is typed `a^b`). There is a *Reals* module that defines ordinary division, but it is rarely used because the TLC model checker cannot evaluate the operator /. So, we define *Divides* using the operators defined in the *Integers* module.

The *Integers* module is about integers, and n/p is not, in general, an integer. The only arithmetic operations you learned in grade school that the module defines are addition (+), subtraction (−), multiplication (*), and exponentiation (a^b is typed `a^b`). There is a *Reals* module that defines ordinary division, but it is rarely used because the TLC model checker cannot evaluate the operator /. So, we define *Divides* using the operators defined in the *Integers* module.

yellow: .55

The *Integers* module is about integers, and n/p is not, in general, an integer. The only arithmetic operations you learned in grade school that the module defines are addition (+), subtraction (-), multiplication (*), and exponentiation (a^b is typed `a^b`). There is a *Reals* module that defines ordinary division, but it is rarely used because the TLC model checker cannot evaluate the operator /. So, we define *Divides* using the operators defined in the *Integers* module.

?

←

→

C

I

S

The *Integers* module is about integers, and n/p is not, in general, an integer. The only arithmetic operations you learned in grade school that the module defines are addition (+), subtraction (-), multiplication (*), and exponentiation (a^b is typed `a^b`). There is a *Reals* module that defines ordinary division, but it is rarely used because the TLC model checker cannot evaluate the operator /. So, we define *Divides* using the operators defined in the *Integers* module.

red: .6

The *Integers* module is about integers, and n/p is not, in general, an integer. The only arithmetic operations you learned in grade school that the module defines are addition (+), subtraction (-), multiplication (*), and exponentiation (a^b is typed `a^b`). There is a *Reals* module that defines ordinary division, but it is rarely used because the TLC model checker cannot evaluate the operator /. So, we define *Divides* using the operators defined in the *Integers* module.

The *Integers* module is about integers, and n/p is not, in general, an integer. The only arithmetic operations you learned in grade school that the module defines are addition (+), subtraction (-), multiplication (*), and exponentiation (a^b is typed `a^b`). There is a *Reals* module that defines ordinary division, but it is rarely used because the TLC model checker cannot evaluate the operator /. So, we define *Divides* using the operators defined in the *Integers* module.

green: .6

The *Integers* module is about integers, and n/p is not, in general, an integer. The only arithmetic operations you learned in grade school that the module defines are addition (+), subtraction (-), multiplication (*), and exponentiation (a^b is typed `a^b`). There is a *Reals* module that defines ordinary division, but it is rarely used because the TLC model checker cannot evaluate the operator /. So, we define *Divides* using the operators defined in the *Integers* module.

The *Integers* module is about integers, and n/p is not, in general, an integer. The only arithmetic operations you learned in grade school that the module defines are addition (+), subtraction (-), multiplication (*), and exponentiation (a^b is typed `a^b`). There is a *Reals* module that defines ordinary division, but it is rarely used because the TLC model checker cannot evaluate the operator /. So, we define *Divides* using the operators defined in the *Integers* module.

blue: .6

?
←
→
C
I
S

The *Integers* module is about integers, and n/p is not, in general, an integer. The only arithmetic operations you learned in grade school that the module defines are addition (+), subtraction (−), multiplication (*), and exponentiation (a^b is typed `a^b`). There is a *Reals* module that defines ordinary division, but it is rarely used because the TLC model checker cannot evaluate the operator /. So, we define *Divides* using the operators defined in the *Integers* module.

The *Integers* module is about integers, and n/p is not, in general, an integer. The only arithmetic operations you learned in grade school that the module defines are addition (+), subtraction (−), multiplication (*), and exponentiation (a^b is typed `a^b`). There is a *Reals* module that defines ordinary division, but it is rarely used because the TLC model checker cannot evaluate the operator /. So, we define *Divides* using the operators defined in the *Integers* module.

aqua: .6

The *Integers* module is about integers, and n/p is not, in general, an integer. The only arithmetic operations you learned in grade school that the module defines are addition (+), subtraction (−), multiplication (*), and exponentiation (a^b is typed `a^b`). There is a *Reals* module that defines ordinary division, but it is rarely used because the TLC model checker cannot evaluate the operator /. So, we define *Divides* using the operators defined in the *Integers* module.

The *Integers* module is about integers, and n/p is not, in general, an integer. The only arithmetic operations you learned in grade school that the module defines are addition (+), subtraction (−), multiplication (*), and exponentiation (a^b is typed `a^b`). There is a *Reals* module that defines ordinary division, but it is rarely used because the TLC model checker cannot evaluate the operator /. So, we define *Divides* using the operators defined in the *Integers* module.

magenta: .6

The *Integers* module is about integers, and n/p is not, in general, an integer. The only arithmetic operations you learned in grade school that the module defines are addition (+), subtraction (−), multiplication (*), and exponentiation (a^b is typed `a^b`). There is a *Reals* module that defines ordinary division, but it is rarely used because the TLC model checker cannot evaluate the operator /. So, we define *Divides* using the operators defined in the *Integers* module.

The *Integers* module is about integers, and n/p is not, in general, an integer. The only arithmetic operations you learned in grade school that the module defines are addition (+), subtraction (−), multiplication (*), and exponentiation (a^b is typed `a^b`). There is a *Reals* module that defines ordinary division, but it is rarely used because the TLC model checker cannot evaluate the operator /. So, we define *Divides* using the operators defined in the *Integers* module.

yellow: .6

?
←
→
C
I
S

The *Integers* module is about integers, and n/p is not, in general, an integer. The only arithmetic operations you learned in grade school that the module defines are addition (+), subtraction (−), multiplication (*), and exponentiation (a^b is typed `a^b`). There is a *Reals* module that defines ordinary division, but it is rarely used because the TLC model checker cannot evaluate the operator /. So, we define *Divides* using the operators defined in the *Integers* module.

The *Integers* module is about integers, and n/p is not, in general, an integer. The only arithmetic operations you learned in grade school that the module defines are addition (+), subtraction (−), multiplication (*), and exponentiation (a^b is typed `a^b`). There is a *Reals* module that defines ordinary division, but it is rarely used because the TLC model checker cannot evaluate the operator /. So, we define *Divides* using the operators defined in the *Integers* module.

red: .65

The *Integers* module is about integers, and n/p is not, in general, an integer. The only arithmetic operations you learned in grade school that the module defines are addition (+), subtraction (−), multiplication (*), and exponentiation (a^b is typed `a^b`). There is a *Reals* module that defines ordinary division, but it is rarely used because the TLC model checker cannot evaluate the operator /. So, we define *Divides* using the operators defined in the *Integers* module.

The *Integers* module is about integers, and n/p is not, in general, an integer. The only arithmetic operations you learned in grade school that the module defines are addition (+), subtraction (−), multiplication (*), and exponentiation (a^b is typed `a^b`). There is a *Reals* module that defines ordinary division, but it is rarely used because the TLC model checker cannot evaluate the operator /. So, we define *Divides* using the operators defined in the *Integers* module.

green: .65

The *Integers* module is about integers, and n/p is not, in general, an integer. The only arithmetic operations you learned in grade school that the module defines are addition (+), subtraction (−), multiplication (*), and exponentiation (a^b is typed `a^b`). There is a *Reals* module that defines ordinary division, but it is rarely used because the TLC model checker cannot evaluate the operator /. So, we define *Divides* using the operators defined in the *Integers* module.

The *Integers* module is about integers, and n/p is not, in general, an integer. The only arithmetic operations you learned in grade school that the module defines are addition (+), subtraction (−), multiplication (*), and exponentiation (a^b is typed `a^b`). There is a *Reals* module that defines ordinary division, but it is rarely used because the TLC model checker cannot evaluate the operator /. So, we define *Divides* using the operators defined in the *Integers* module.

blue: .65

?
←
→
C
I
S

The *Integers* module is about integers, and n/p is not, in general, an integer. The only arithmetic operations you learned in grade school that the module defines are addition (+), subtraction (−), multiplication (*), and exponentiation (a^b is typed `a^b`). There is a *Reals* module that defines ordinary division, but it is rarely used because the TLC model checker cannot evaluate the operator /. So, we define *Divides* using the operators defined in the *Integers* module.

The *Integers* module is about integers, and n/p is not, in general, an integer. The only arithmetic operations you learned in grade school that the module defines are addition (+), subtraction (−), multiplication (*), and exponentiation (a^b is typed `a^b`). There is a *Reals* module that defines ordinary division, but it is rarely used because the TLC model checker cannot evaluate the operator /. So, we define *Divides* using the operators defined in the *Integers* module.

aqua: .65

The *Integers* module is about integers, and n/p is not, in general, an integer. The only arithmetic operations you learned in grade school that the module defines are addition (+), subtraction (−), multiplication (*), and exponentiation (a^b is typed `a^b`). There is a *Reals* module that defines ordinary division, but it is rarely used because the TLC model checker cannot evaluate the operator /. So, we define *Divides* using the operators defined in the *Integers* module.

The *Integers* module is about integers, and n/p is not, in general, an integer. The only arithmetic operations you learned in grade school that the module defines are addition (+), subtraction (−), multiplication (*), and exponentiation (a^b is typed `a^b`). There is a *Reals* module that defines ordinary division, but it is rarely used because the TLC model checker cannot evaluate the operator /. So, we define *Divides* using the operators defined in the *Integers* module.

magenta: .65

The *Integers* module is about integers, and n/p is not, in general, an integer. The only arithmetic operations you learned in grade school that the module defines are addition (+), subtraction (−), multiplication (*), and exponentiation (a^b is typed `a^b`). There is a *Reals* module that defines ordinary division, but it is rarely used because the TLC model checker cannot evaluate the operator /. So, we define *Divides* using the operators defined in the *Integers* module.

The *Integers* module is about integers, and n/p is not, in general, an integer. The only arithmetic operations you learned in grade school that the module defines are addition (+), subtraction (−), multiplication (*), and exponentiation (a^b is typed `a^b`). There is a *Reals* module that defines ordinary division, but it is rarely used because the TLC model checker cannot evaluate the operator /. So, we define *Divides* using the operators defined in the *Integers* module.

yellow: .65