Aerobus

v1.2

Generated by Doxygen 1.9.8

1 Concept Index	1
1.1 Concepts	. 1
2 Class Index	3
2.1 Class List	. 3
3 File Index	5
3.1 File List	. 5
4 Concept Documentation	7
4.1 aerobus::IsEuclideanDomain Concept Reference	. 7
4.1.1 Concept definition	. 7
4.1.2 Detailed Description	. 7
4.2 aerobus::IsField Concept Reference	. 7
4.2.1 Concept definition	. 7
4.2.2 Detailed Description	. 8
4.3 aerobus::IsRing Concept Reference	. 8
4.3.1 Concept definition	. 8
4.3.2 Detailed Description	. 8
5 Class Documentation	9
5.1 aerobus::polynomial< Ring, variable_name >::val< coeffN >::coeff_at< index, E > Struct Template Reference	
5.2 aerobus::polynomial< Ring, variable_name >::val< coeffN >::coeff_at< index, std::enable_if_ t<(index<0 index>0)>> Struct Template Reference	.
5.3 aerobus::polynomial< Ring, variable_name >::val< coeffN >::coeff_at< index, std::enable_if_ t<(index==0)>> Struct Template Reference	
5.4 aerobus::ContinuedFraction < values > Struct Template Reference	. 10
5.4.1 Detailed Description	. 10
5.5 aerobus::ContinuedFraction< a0 > Struct Template Reference	. 10
5.5.1 Detailed Description	. 10
5.6 aerobus::ContinuedFraction< a0, rest > Struct Template Reference	
5.6.1 Detailed Description	
5.7 aerobus::i32 Struct Reference	
5.7.1 Detailed Description	
5.7.2 Member Data Documentation	
5.7.2.1 eq_v	_
5.7.2.2 pos_v	
5.8 aerobus::i64 Struct Reference	
5.8.1 Detailed Description	
	_
5.8.2 Member Typedef Documentation	_
5.8.2.1 add_t	
5.8.2.2 div_t	
5.8.2.3 eq_t	. 16

5.8.2.4 gcd_t	16
5.8.2.5 gt_t	16
5.8.2.6 lt_t	16
$5.8.2.7~mod_t$	17
5.8.2.8 mul_t	17
5.8.2.9 pos_t	17
5.8.2.10 sub_t	17
5.8.3 Member Data Documentation	18
5.8.3.1 eq_v	18
5.8.3.2 gt_v	18
5.8.3.3 lt_v	18
5.8.3.4 pos_v	18
$5.9 \ aerobus::polynomial < Ring, \ variable_name > ::horner_evaluation < \ valueRing, \ P > ::inner < \ index,$	
stop > Struct Template Reference	19
5.10 aerobus::polynomial< Ring, variable_name >::horner_evaluation< valueRing, P >::inner< stop, stop > Struct Template Reference	19
5.11 aerobus::is_prime< n > Struct Template Reference	
5.11.1 Detailed Description	
5.12 aerobus::polynomial < Ring, variable_name > Struct Template Reference	20
5.12.1 Detailed Description	
5.12.2 Member Typedef Documentation	
5.12.2.1 add_t	
5.12.2.2 derive t	
5.12.2.3 div t	
5.12.2.4 eq_t	
5.12.2.5 gcd_t	
5.12.2.6 gt t	
5.12.2.7 lt t	23
5.12.2.8 mod_t	23
5.12.2.9 monomial_t	24
5.12.2.10 mul_t	24
5.12.2.11 pos_t	24
5.12.2.12 simplify_t	25
5.12.2.13 sub_t	25
5.13 aerobus::type_list< Ts >::pop_front Struct Reference	25
5.13.1 Detailed Description	25
5.14 aerobus::Quotient< Ring, X > Struct Template Reference	26
5.15 aerobus::type_list< Ts >::split< index > Struct Template Reference	26
5.15.1 Detailed Description	27
5.16 aerobus::type_list< Ts $>$ Struct Template Reference	28
5.16.1 Detailed Description	28
5.16.2 Member Typedef Documentation	29
5.16.2.1 at	29

5.16.2.2 concat	. 29
5.16.2.3 insert	. 29
5.16.2.4 push_back	. 30
5.16.2.5 push_front	. 30
5.16.2.6 remove	. 30
5.17 aerobus::type_list<> Struct Reference	. 30
5.18 aerobus::i32::val $<$ x $>$ Struct Template Reference	. 31
5.18.1 Detailed Description	. 31
5.18.2 Member Function Documentation	. 32
5.18.2.1 eval()	. 32
5.18.2.2 get()	. 32
5.19 aerobus::i64::val < x > Struct Template Reference	. 32
5.19.1 Detailed Description	. 33
5.19.2 Member Function Documentation	. 33
5.19.2.1 eval()	. 33
5.19.2.2 get()	. 33
$5.20 \; aerobus::polynomial < Ring, \; variable_name > ::val < coeffN, \; coeffs > Struct \; Template \; Reference \; .$. 35
5.20.1 Detailed Description	. 35
5.20.2 Member Typedef Documentation	. 36
5.20.2.1 coeff_at_t	. 36
5.20.3 Member Function Documentation	. 36
5.20.3.1 eval()	. 36
5.20.3.2 to_string()	. 37
5.21 aerobus::Quotient < Ring, X >::val < V > Struct Template Reference	. 37
5.22 aerobus::zpz::val< x > Struct Template Reference	. 37
5.23 aerobus::polynomial < Ring, variable_name >::val < coeffN > Struct Template Reference	. 38
5.23.1 Detailed Description	. 38
5.24 aerobus::zpz Struct Template Reference	. 39
5.24.1 Detailed Description	. 40
6 File Documentation	41
6.1 aerobus.h	. 41
7 Examples	123
7.1 i32::template	. 123
7.2 i64::template	. 123
7.3 polynomial	. 123
7.4 PI_fraction::val	. 124
7.5 E_fraction::val	. 124
Index	125
the state of the s	

Chapter 1

Concept Index

1.1 Concepts

Here is a list of all documented concepts with brief descriptions:

aerobus::IsEuclideanDomain	
Concept to express R is an euclidean domain	7
aerobus::IsField	
Concept to express R is a field	7
aerobus::IsRing	
Concept to express R is a Ring (ordered)	8

2 Concept Index

Chapter 2

Class Index

2.1 Class List

Here are the classes, structs, unions and interfaces with brief descriptions:

aerobus::polynomial< Ring, variable_name >::val< coeffN >::coeff_at< index, E >	9
aerobus::polynomial< Ring, variable_name >::val< coeffN >::coeff_at< index, std::enable_if_t<(index< 0	(
9	
aerobus::polynomial < Ring, variable_name >::val < coeffN >::coeff_at < index, std::enable_if_t < (index==0)	0)>>
9	
aerobus::ContinuedFraction< values >	
Continued fraction a0 + 1/(a1 + 1/())	10
aerobus::ContinuedFraction< a0 >	
Specialization for only one coefficient, technically just 'a0'	10
aerobus::ContinuedFraction< a0, rest >	
Specialization for multiple coefficients (strictly more than one)	11
aerobus::i32	
32 bits signed integers, seen as a algebraic ring with related operations	11
aerobus::i64	
64 bits signed integers, seen as a algebraic ring with related operations	14
aerobus::polynomial< Ring, variable_name >::horner_evaluation< valueRing, P >::inner< index, stop > 19	
aerobus::polynomial< Ring, variable_name >::horner_evaluation< valueRing, P >::inner< stop, stop >	19
aerobus::is_prime< n >	
Checks if n is prime	19
aerobus::polynomial < Ring, variable_name >	20
aerobus::type_list< Ts >::pop_front	
Removes types from head of the list	25
aerobus::Quotient < Ring, X >	26
aerobus::type_list< Ts >::split< index >	
Splits list at index	26
aerobus::type_list< Ts >	
Empty pure template struct to handle type list	28
aerobus::type_list<>	30
aerobus::i32::val< x >	
Values in i32, again represented as types	31
aerobus::i64::val< x >	
Values in i64	32
aerobus::polynomial< Ring, variable_name >::val< coeffN, coeffs >	
Values (seen as types) in polynomial ring	35

4 Class Index

$aerobus:: Quotient < Ring, X > :: val < V > \qquad . \qquad$	37
$aerobus::zpz ::val < x > \qquad \dots \qquad \dots \qquad \dots$	37
aerobus::polynomial < Ring, variable_name >::val < coeffN >	
Specialization for constants	38
$aerobus::zpz \dots \dots$	39

Chapter 3

File Index

•			
~~ ~	File	•	ıct
J. I	1 116	_	131

Here is a list of all documented files with brief descriptions:	
src/aerobus.h	41

6 File Index

Chapter 4

Concept Documentation

4.1 aerobus::IsEuclideanDomain Concept Reference

Concept to express R is an euclidean domain.

```
#include <aerobus.h>
```

4.1.1 Concept definition

```
template<typename R>
concept aerobus::IsEuclideanDomain = IsRing<R> && requires {
            typename R::template div_t<typename R::one, typename R::one>;
            typename R::template mod_t<typename R::one, typename R::one>;
            typename R::template gcd_t<typename R::one, typename R::one>;
            typename R::template eq_t<typename R::one, typename R::one>;
            typename R::template pos_t<typename R::one>;
            R::template pos_t<typename R::one> == true;
            R::is_euclidean_domain == true;
}
```

4.1.2 Detailed Description

Concept to express R is an euclidean domain.

4.2 aerobus::IsField Concept Reference

Concept to express R is a field.

```
#include <aerobus.h>
```

4.2.1 Concept definition

```
template<typename R>
concept aerobus::IsField = IsEuclideanDomain<R> && requires {
          R::is_field == true;
}
```

4.2.2 Detailed Description

Concept to express R is a field.

4.3 aerobus::IsRing Concept Reference

Concept to express R is a Ring (ordered)

```
#include <aerobus.h>
```

4.3.1 Concept definition

```
template < typename R>
concept aerobus::IsRing = requires {
    typename R::one;
    typename R:zero;
    typename R::template add_t < typename R::one, typename R::one>;
    typename R::template sub_t < typename R::one, typename R::one>;
    typename R::template mul_t < typename R::one, typename R::one>;
}
```

4.3.2 Detailed Description

Concept to express R is a Ring (ordered)

Chapter 5

Class Documentation

5.1 aerobus::polynomial< Ring, variable_name >::val< coeffN >::coeff_at< index, E > Struct Template Reference

The documentation for this struct was generated from the following file:

- · src/aerobus.h
- 5.2 aerobus::polynomial< Ring, variable_name >::val< coeffN >::coeff_at< index, std::enable_if_t<(index<0||index>0)>> Struct Template Reference

Public Types

• using type = typename Ring::zero

The documentation for this struct was generated from the following file:

- src/aerobus.h
- 5.3 aerobus::polynomial < Ring, variable_name >::val < coeffN >::coeff_at < index, std::enable_if_t < (index==0) > > Struct Template Reference

Public Types

• using type = aN

The documentation for this struct was generated from the following file:

• src/aerobus.h

5.4 aerobus::ContinuedFraction< values > Struct Template Reference

```
represents a continued fraction a0 + 1/(a1 + 1/(...))
#include <aerobus.h>
```

5.4.1 Detailed Description

```
template < int64_t... values > struct aerobus::ContinuedFraction < values > represents a continued fraction a0 + 1/(a1 + 1/(...)) Template Parameters
```

...values are aerobus::i64

The documentation for this struct was generated from the following file:

· src/aerobus.h

5.5 aerobus::ContinuedFraction < a0 > Struct Template Reference

Specialization for only one coefficient, technically just 'a0'.

```
#include <aerobus.h>
```

Public Types

using type = typename q64::template inject_constant_t< a0 >

Static Public Attributes

• static constexpr double **val** = type::template get<double>()

5.5.1 Detailed Description

```
template<int64_t a0> struct aerobus::ContinuedFraction< a0 >
```

Specialization for only one coefficient, technically just 'a0'.

Template Parameters

```
a0 an integer (aerobus::i64)
```

The documentation for this struct was generated from the following file:

· src/aerobus.h

5.6 aerobus::ContinuedFraction< a0, rest... > Struct Template Reference

specialization for multiple coefficients (strictly more than one)

```
#include <aerobus.h>
```

Public Types

• using **type** = q64::template add_t< typename q64::template inject_constant_t< a0 >, typename q64
::template div_t< typename q64::one, typename ContinuedFraction< rest... >::type > >

Static Public Attributes

• static constexpr double val = type::template get<double>()

5.6.1 Detailed Description

```
template<int64_t a0, int64_t... rest> struct aerobus::ContinuedFraction< a0, rest... >
```

specialization for multiple coefficients (strictly more than one)

Template Parameters

a0	an integer (aerobus::i64)
rest	integers (aerobus::i64)

The documentation for this struct was generated from the following file:

src/aerobus.h

5.7 aerobus::i32 Struct Reference

32 bits signed integers, seen as a algebraic ring with related operations

```
#include <aerobus.h>
```

Classes

• struct val values in i32, again represented as types

Public Types

```
• using inner_type = int32 t
• using zero = val< 0 >
     constant zero
• using one = val< 1 >
     constant one
• template<auto x>
  using inject_constant_t = val< static_cast< int32_t >(x)>
• template<typename v >
  using inject_ring_t = v
• template<typename v1 , typename v2 >
  using add_t = typename add< v1, v2 >::type
     addition operator
• template<typename v1 , typename v2 >
  using sub_t = typename sub < v1, v2 >::type
     substraction operator
• template<typename v1 , typename v2 >
  using mul_t = typename mul < v1, v2 >::type
     multiplication operator
• template<typename v1 , typename v2 >
  using div_t = typename div < v1, v2 >::type
     division operator
• template<typename v1 , typename v2 >
  using mod_t = typename remainder < v1, v2 >::type
     modulus operator
• template<typename v1 , typename v2 >
  using gt_t = typename gt < v1, v2 >::type
     strictly greater operator (v1 > v2)
• template<typename v1 , typename v2 >
  using It_t = typename It < v1, v2 >::type
     strict less operator (v1 < v2)
• template<typename v1 , typename v2 >
  using eq_t = typename eq< v1, v2 >::type
     equality operator (type)

    template<typename v1 , typename v2 >

  using gcd_t = gcd_t < i32, v1, v2 >
     greatest common divisor
• template<typename v >
  using pos_t = typename pos< v >::type
     positivity (type)(v > 0)
```

Static Public Attributes

```
• static constexpr bool is_field = false
```

integers are not a field

• static constexpr bool is_euclidean_domain = true

integers are an euclidean domain

template<typename v1, typename v2 >
 static constexpr bool eq_v = eq_t<v1, v2>::value
 equality operator (boolean value)

template<typename v >
 static constexpr bool pos_v = pos_t<v>::value
 positivity (boolean value)

5.7.1 Detailed Description

32 bits signed integers, seen as a algebraic ring with related operations

5.7.2 Member Data Documentation

5.7.2.1 eq v

```
template<typename v1 , typename v2 >
constexpr bool aerobus::i32::eq_v = eq_t<v1, v2>::value [static], [constexpr]
```

equality operator (boolean value)

Template Parameters

v1	
v2	

5.7.2.2 pos_v

```
template<typename v >
constexpr bool aerobus::i32::pos_v = pos_t < v > ::value [static], [constexpr]
```

positivity (boolean value)

Template Parameters



The documentation for this struct was generated from the following file:

• src/aerobus.h

5.8 aerobus::i64 Struct Reference

64 bits signed integers, seen as a algebraic ring with related operations

```
#include <aerobus.h>
```

Classes

• struct val values in i64

Public Types

```
• using inner_type = int64_t

    template<auto x>

  using inject_constant_t = val< static_cast< int64_t >(x)>
• template<typename v >
  using inject_ring_t = v

    using zero = val < 0 >

     constant zero
• using one = val< 1 >
     constant one
• template<typename v1 , typename v2 >
  using add_t = typename add< v1, v2 >::type
     addition operator
• template<typename v1 , typename v2 >
  using sub_t = typename sub< v1, v2 >::type
     substraction operator
• template<typename v1 , typename v2 >
  using mul_t = typename mul < v1, v2 >::type
     multiplication operator

    template<typename v1 , typename v2 >

  using div_t = typename div < v1, v2 >::type
     division operator
• template<typename v1 , typename v2 >
  using mod_t = typename remainder < v1, v2 >::type
     modulus operator
• template<typename v1 , typename v2 >
  using gt_t = typename gt < v1, v2 >::type
     strictly greater operator (v1 > v2) - type
• template<typename v1 , typename v2 >
  using It_t = typename It< v1, v2 >::type
     strict less operator (v1 < v2)
• template<typename v1 , typename v2 >
  using eq_t = typename eq< v1, v2 >::type
     equality operator (type)
• template<typename v1 , typename v2 >
  using gcd_t = gcd_t < i64, v1, v2 >
     greatest common divisor
• template<typename v >
  using pos_t = typename pos< v >::type
     is v posititive (type)
```

Static Public Attributes

```
    static constexpr bool is_field = false
        integers are not a field
    static constexpr bool is_euclidean_domain = true
        integers are an euclidean domain
    template<typename v1 , typename v2 >
        static constexpr bool gt_v = gt_t<v1, v2>::value
            strictly greater operator (v1 > v2) - boolean value
    template<typename v1 , typename v2 >
        static constexpr bool It_v = It_t<v1, v2>::value
            strictly smaller operator (v1 < v2) - boolean value</li>
    template<typename v1 , typename v2 >

    template<typename v1 , typename v2 >
```

template < typename v >
 static constexpr bool pos_v = pos_t < v > ::value
 positivity (boolean value)

5.8.1 Detailed Description

64 bits signed integers, seen as a algebraic ring with related operations

5.8.2 Member Typedef Documentation

5.8.2.1 add_t

```
template<typename v1 , typename v2 >
using aerobus::i64::add_t = typename add<v1, v2>::type
```

addition operator

Template Parameters

v1	: an element of aerobus::i64::val
v2	: an element of aerobus::i64::val

5.8.2.2 div_t

```
template<typename v1 , typename v2 >
using aerobus::i64::div_t = typename div<v1, v2>::type
```

division operator

v1	: an element of aerobus::i64::val
v2	: an element of aerobus::i64::val

5.8.2.3 eq_t

```
template<typename v1 , typename v2 >
using aerobus::i64::eq_t = typename eq<v1, v2>::type
```

equality operator (type)

Template Parameters

v1	: an element of aerobus::i64::val
v2	: an element of aerobus::i64::val

5.8.2.4 gcd_t

```
template<typename v1 , typename v2 >
using aerobus::i64::gcd_t = gcd_t<i64, v1, v2>
```

greatest common divisor

Template Parameters

v1	: an element of aerobus::i64::val
v2	: an element of aerobus::i64::val

5.8.2.5 gt_t

```
template<typename v1 , typename v2 >
using aerobus::i64::gt_t = typename gt<v1, v2>::type
```

strictly greater operator (v1 > v2) - type

Template Parameters

v1	: an element of aerobus::i64::val
v2	: an element of aerobus::i64::val

5.8.2.6 lt_t

```
template<typename v1 , typename v2 >
using aerobus::i64::lt_t = typename lt<v1, v2>::type
```

strict less operator (v1 < v2)

v1	: an element of aerobus::i64::val
v2	: an element of aerobus::i64::val

5.8.2.7 mod_t

```
template<typename v1 , typename v2 >
using aerobus::i64::mod_t = typename remainder<v1, v2>::type
```

modulus operator

Template Parameters

v1	: an element of aerobus::i64::val
v2	: an element of aerobus::i64::val

5.8.2.8 mul_t

```
template<typename v1 , typename v2 >
using aerobus::i64::mul_t = typename mul<v1, v2>::type
```

multiplication operator

Template Parameters

v1	: an element of aerobus::i64::val
v2	: an element of aerobus::i64::val

5.8.2.9 pos_t

```
template<typename v >
using aerobus::i64::pos_t = typename pos<v>::type
```

is v posititive (type)

Template Parameters

```
v1 : an element of aerobus::i64::val
```

5.8.2.10 sub_t

```
template<typename v1 , typename v2 >
using aerobus::i64::sub_t = typename sub<v1, v2>::type
```

substraction operator

v1	: an element of aerobus::i64::val
v2	: an element of aerobus::i64::val

5.8.3 Member Data Documentation

5.8.3.1 eq_v

```
\label{eq:constexpr} \begin{tabular}{ll} template < typename & v1 & typename & v2 & \\ constexpr & bool & aerobus::i64::eq_v & = eq_t < v1, & v2 > ::value & [static], & [constexpr] & (static) & (st
```

equality operator (boolean value)

Template Parameters

v1	: an element of aerobus::i64::val
v2	: an element of aerobus::i64::val

5.8.3.2 gt_v

```
template<typename v1 , typename v2 > constexpr bool aerobus::i64::gt_v = gt_t<v1, v2>::value [static], [constexpr]
```

strictly greater operator (v1 > v2) - boolean value

Template Parameters

v1	: an element of aerobus::i64::val
v2	: an element of aerobus::i64::val

5.8.3.3 lt_v

```
\label{template} $$ \ensuremath{\sf typename}$ v1 , typename v2 > $$ constexpr bool aerobus::i64::lt_v = lt_t < v1, v2>::value [static], [constexpr] $$
```

strictly smaller operator (v1 < v2) - boolean value

Template Parameters

v1	: an element of aerobus::i64::val
v2	: an element of aerobus::i64::val

5.8.3.4 pos_v

```
template<typename v >
constexpr bool aerobus::i64::pos_v = pos_t < v > ::value [static], [constexpr]
```

positivity (boolean value)

Template Parameters

v : an element of aerobus::i64::val

The documentation for this struct was generated from the following file:

· src/aerobus.h

5.9 aerobus::polynomial< Ring, variable_name >::horner_evaluation< valueRing, P >::inner< index, stop > Struct Template Reference

Static Public Member Functions

• static constexpr valueRing func (const valueRing &accum, const valueRing &x)

The documentation for this struct was generated from the following file:

· src/aerobus.h

5.10 aerobus::polynomial< Ring, variable_name >::horner_evaluation< valueRing, P >::inner< stop, stop > Struct Template Reference

Static Public Member Functions

• static constexpr valueRing func (const valueRing &accum, const valueRing &x)

The documentation for this struct was generated from the following file:

· src/aerobus.h

5.11 aerobus::is prime< n > Struct Template Reference

checks if n is prime

#include <aerobus.h>

Static Public Attributes

5.11.1 Detailed Description

template<int32_t n> struct aerobus::is_prime< n >

checks if n is prime

Template Parameters

```
n
```

The documentation for this struct was generated from the following file:

· src/aerobus.h

5.12 aerobus::polynomial< Ring, variable_name > Struct Template Reference

```
#include <aerobus.h>
```

Classes

```
    struct val
        values (seen as types) in polynomial ring
        struct val < coeffN >
```

specialization for constants

Public Types

```
• using zero = val< typename Ring::zero >
     constant zero
• using one = val< typename Ring::one >
     constant one

    using X = val< typename Ring::one, typename Ring::zero >

     generator
• template<typename P >
  using simplify_t = typename simplify< P >::type
     simplifies a polynomial (recursively deletes highest degree if zero, do nothing otherwise)
• template<typename v1 , typename v2 >
  using add_t = typename add< v1, v2 >::type
     adds two polynomials
• template<typename v1 , typename v2 >
  using sub_t = typename sub< v1, v2 >::type
     substraction of two polynomials

    template<typename v1 , typename v2 >

  using mul_t = typename mul < v1, v2 >::type
     multiplication of two polynomials
• template<typename v1 , typename v2 >
  using eq_t = typename eq_helper< v1, v2 >::type
     equality operator
• template<typename v1 , typename v2 >
  using lt_t = typename lt_helper< v1, v2 >::type
     strict less operator
• template<typename v1 , typename v2 >
  using gt_t = typename gt_helper< v1, v2 >::type
```

```
strict greater operator
• template<typename v1 , typename v2 >
  using div_t = typename div < v1, v2 >::q_type
     division operator
• template<typename v1 , typename v2 >
  using mod_t = typename div_helper< v1, v2, zero, v1 >::mod_type
     modulo operator
• template<typename coeff , size_t deg>
  using monomial t = typename monomial < coeff, deg >::type
     monomial : coeff X^{\wedge} deg
• template<typename v >
  using derive t = typename derive helper< v >::type
     derivation operator

    template<typename v >

  using pos_t = typename Ring::template pos_t < typename v::aN >
     checks for positivity (an > 0)
• template<typename v1 , typename v2 >
  using gcd_t = std::conditional_t < Ring::is_euclidean_domain, typename make_unit < gcd_t < polynomial <
  Ring, variable_name >, v1, v2 > >::type, void >
     greatest common divisor of two polynomials

    template<auto x>

  using inject_constant_t = val< typename Ring::template inject_constant_t < x > >

    template<typename v >

  using inject_ring_t = val< v >
```

Static Public Attributes

- static constexpr bool is_field = false
- static constexpr bool is_euclidean_domain = Ring::is_euclidean_domain
- template<typename v >
 static constexpr bool pos_v = pos_t<v>::value

5.12.1 Detailed Description

```
template<typename Ring, char variable_name = 'x'>
requires lsEuclideanDomain<Ring>
struct aerobus::polynomial< Ring, variable_name >
```

polynomial with coefficients in Ring Ring must be an integral domain

5.12.2 Member Typedef Documentation

5.12.2.1 add t

```
template<typename Ring , char variable_name = 'x'>
template<typename v1 , typename v2 >
using aerobus::polynomial< Ring, variable_name >::add_t = typename add<v1, v2>::type
adds two polynomials
```

Template Parameters

v1	
v2	

5.12.2.2 derive t

```
template<typename Ring , char variable_name = 'x'>
template<typename v >
using aerobus::polynomial< Ring, variable_name >::derive_t = typename derive_helper<v>::type
```

derivation operator

Template Parameters



5.12.2.3 div_t

```
template<typename Ring , char variable_name = 'x'>
template<typename v1 , typename v2 >
using aerobus::polynomial< Ring, variable_name >::div_t = typename div<v1, v2>::q_type
```

division operator

Template Parameters

v1	
v2	

5.12.2.4 eq t

```
template<typename Ring , char variable_name = 'x'>
template<typename v1 , typename v2 >
using aerobus::polynomial< Ring, variable_name >::eq_t = typename eq_helper<v1, v2>::type
```

equality operator

Template Parameters

v1	
v2	

5.12.2.5 gcd_t

template<typename Ring , char variable_name = 'x'>

greatest common divisor of two polynomials

Template Parameters

v1	
v2	

5.12.2.6 gt_t

```
template<typename Ring , char variable_name = 'x'>
template<typename v1 , typename v2 >
using aerobus::polynomial< Ring, variable_name >::gt_t = typename gt_helper<v1, v2>::type
```

strict greater operator

Template Parameters

v1	
v2	

5.12.2.7 lt_t

```
template<typename Ring , char variable_name = 'x'>
template<typename v1 , typename v2 >
using aerobus::polynomial< Ring, variable_name >::lt_t = typename lt_helper<v1, v2>::type
```

strict less operator

Template Parameters

v1	
v2	

5.12.2.8 mod_t

```
template<typename Ring , char variable_name = 'x'>
template<typename v1 , typename v2 >
using aerobus::polynomial< Ring, variable_name >::mod_t = typename div_helper<v1, v2, zero,
v1>::mod_type
```

modulo operator

Template Parameters

v1	
v2	

5.12.2.9 monomial t

```
template<typename Ring , char variable_name = 'x'>
template<typename coeff , size_t deg>
using aerobus::polynomial< Ring, variable_name >::monomial_t = typename monomial<coeff, deg>
::type
```

monomial : coeff X^deg

Template Parameters

coeff	
deg	

5.12.2.10 mul_t

```
template<typename Ring , char variable_name = 'x'>
template<typename v1 , typename v2 >
using aerobus::polynomial< Ring, variable_name >::mul_t = typename mul<v1, v2>::type
```

multiplication of two polynomials

Template Parameters

v1	
v2	

5.12.2.11 pos_t

checks for positivity (an > 0)

5.12.2.12 simplify_t

```
template<typename Ring , char variable_name = 'x'>
template<typename P >
using aerobus::polynomial< Ring, variable_name >::simplify_t = typename simplify<P>::type
```

simplifies a polynomial (recursively deletes highest degree if zero, do nothing otherwise)

Template Parameters



5.12.2.13 sub_t

```
template<typename Ring , char variable_name = 'x'>
template<typename v1 , typename v2 >
using aerobus::polynomial< Ring, variable_name >::sub_t = typename sub<v1, v2>::type
```

substraction of two polynomials

Template Parameters

v1	
v2	

The documentation for this struct was generated from the following file:

· src/aerobus.h

5.13 aerobus::type_list< Ts >::pop_front Struct Reference

removes types from head of the list

```
#include <aerobus.h>
```

Public Types

- using type = typename internal::pop_front_h< Ts... >::head
 type that was previously head of the list
- using **tail** = typename internal::pop_front_h< Ts... >::tail remaining types in parent list when front is removed

5.13.1 Detailed Description

```
template<typename... Ts> struct aerobus::type_list< Ts >::pop_front
```

removes types from head of the list

The documentation for this struct was generated from the following file:

• src/aerobus.h

5.14 aerobus::Quotient < Ring, X > Struct Template Reference

Classes

struct val

Public Types

```
• using zero = val< typename Ring::zero >
using one = val< typename Ring::one >
• template<typename v1, typename v2 >
  using add_t = val< typename Ring::template add_t< typename v1::type, typename v2::type >>
• template<typename v1 , typename v2 >
 using mul_t = val< typename Ring::template mul_t< typename v1::type, typename v2::type > >

    template<typename v1, typename v2 >

  using div_t = val< typename Ring::template div_t< typename v1::type, typename v2::type >>
• template<typename v1 , typename v2 >
  using mod_t = val< typename Ring::template mod_t< typename v1::type, typename v2::type >>

    template<typename v1 , typename v2 >

  using eq_t = typename Ring::template eq_t < typename v1::type, typename v2::type >

    template<typename v1 >

  using pos_t = std::true_type

    template<auto x>

  using inject_constant_t = val< typename Ring::template inject_constant_t < x > >

    template<typename v >

  using inject_ring_t = val< v >
```

Static Public Attributes

```
    template<typename v1, typename v2>
        static constexpr bool eq_v = Ring::template eq_t<typename v1::type, typename v2::type>::value
    template<typename v>
        static constexpr bool pos_v = pos_t<v>::value
    static constexpr bool is_euclidean_domain = true
```

The documentation for this struct was generated from the following file:

· src/aerobus.h

5.15 aerobus::type_list< Ts >::split< index > Struct Template Reference

```
splits list at index
```

```
#include <aerobus.h>
```

Public Types

- using head = typename inner::head
- using tail = typename inner::tail

5.15.1 Detailed Description

template<typename... Ts>
template<size_t index>
struct aerobus::type_list< Ts >::split< index >

splits list at index

Template Parameters

index	
-------	--

The documentation for this struct was generated from the following file:

· src/aerobus.h

5.16 aerobus::type_list< Ts > Struct Template Reference

Empty pure template struct to handle type list.

```
#include <aerobus.h>
```

Classes

```
· struct pop_front
```

removes types from head of the list

struct split

splits list at index

Public Types

```
template<typename T >
 using push_front = type_list< T, Ts... >
     Adds T to front of the list.
• template<size_t index>
  using at = internal::type_at_t< index, Ts... >
     returns type at index
template<typename T >
 using push_back = type_list< Ts..., T >
     pushes T at the tail of the list
• template<typename U>
  using concat = typename concat_h< U >::type
     concatenates two list into one
• template<typename T , size_t index>
 using insert = typename internal::insert_h< index, type_list< Ts... >, T >::type
     inserts type at index
• template<size_t index>
  using remove = typename internal::remove_h< index, type_list< Ts... > >::type
     removes type at index
```

Static Public Attributes

```
    static constexpr size_t length = sizeof...(Ts)
    length of list
```

5.16.1 Detailed Description

```
template<typename... Ts> struct aerobus::type_list< Ts >
```

Empty pure template struct to handle type list.

A list of types.

Template Parameters

Ts	

5.16.2 Member Typedef Documentation

5.16.2.1 at

```
template<typename... Ts>
template<size_t index>
using aerobus::type_list< Ts >::at = internal::type_at_t<index, Ts...>
```

returns type at index

Template Parameters

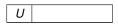


5.16.2.2 concat

```
template<typename... Ts>
template<typename U >
using aerobus::type_list< Ts >::concat = typename concat_h<U>::type
```

concatenates two list into one

Template Parameters



5.16.2.3 insert

```
template<typename... Ts>
template<typename T , size_t index>
using aerobus::type_list< Ts >::insert = typename internal::insert_h<index, type_list<Ts...>,
T>::type
```

inserts type at index

index	
T	

5.16.2.4 push_back

```
{\tt template}{<}{\tt typename}\dots \quad {\tt Ts}{>}
template<typename T >
using aerobus::type_list< Ts >::push_back = type_list<Ts..., T>
pushes T at the tail of the list
```

Template Parameters



5.16.2.5 push_front

```
template<typename... Ts>
{\tt template}{<}{\tt typename}\ {\tt T}\ >
using aerobus::type_list< Ts >::push_front = type_list<T, Ts...>
```

Adds T to front of the list.

Template Parameters



5.16.2.6 remove

```
template<typename... Ts>
template<size_t index>
using aerobus::type_list< Ts >::remove = typename internal::remove_h<index, type_list<Ts...>
>::type
```

removes type at index

Template Parameters



The documentation for this struct was generated from the following file:

· src/aerobus.h

aerobus::type_list<> Struct Reference

Public Types

 $\bullet \;\; {\sf template}{<} {\sf typename} \; {\sf T} >$ using **push_front** = type_list< T >

```
    template < typename T > using push_back = type_list < T >
    template < typename U > using concat = U
    template < typename T, size_t index > using insert = type_list < T >
```

Static Public Attributes

• static constexpr size_t length = 0

The documentation for this struct was generated from the following file:

· src/aerobus.h

5.18 aerobus::i32::val < x > Struct Template Reference

```
values in i32, again represented as types
```

```
#include <aerobus.h>
```

Public Types

```
using is_zero_t = std::bool_constant< x==0 >
is value zero
```

Static Public Member Functions

```
    template < typename valueType > static constexpr valueType get ()
        cast x into valueType
    static std::string to_string ()
        string representation of value
    template < typename valueRing > static constexpr valueRing eval (const valueRing &v)
        cast x into valueRing
```

Static Public Attributes

static constexpr int32_t v = x
 actual value stored in val type

5.18.1 Detailed Description

```
template<int32_t x>
struct aerobus::i32::val< x>
values in i32, again represented as types
```

32 Class Documentation

Template Parameters

```
x an actual integer
```

5.18.2 Member Function Documentation

5.18.2.1 eval()

cast x into valueRing

Template Parameters

5.18.2.2 get()

```
template<iint32_t x>
template<typename valueType >
static constexpr valueType aerobus::i32::val< x >::get ( ) [inline], [static], [constexpr]
```

cast x into valueType

Template Parameters

```
valueType | double for example
```

The documentation for this struct was generated from the following file:

· src/aerobus.h

5.19 aerobus::i64::val< x > Struct Template Reference

```
values in i64
```

```
#include <aerobus.h>
```

Public Types

```
using is_zero_t = std::bool_constant< x==0 >
is value zero
```

Static Public Member Functions

```
    template < typename valueType > static constexpr valueType get ()
        cast value in valueType
    static std::string to_string ()
        string representation
    template < typename valueRing > static constexpr valueRing eval (const valueRing &v)
        cast value in valueRing
```

Static Public Attributes

• static constexpr int64_t v = x

5.19.1 Detailed Description

```
template < int64_t x > struct aerobus::i64::val < x > values in i64

Template Parameters

x an actual integer
```

5.19.2 Member Function Documentation

5.19.2.1 eval()

Template Parameters

```
valueRing (double for example)
```

5.19.2.2 get()

```
template<iint64_t x>
template<typename valueType >
static constexpr valueType aerobus::i64::val< x >::get () [inline], [static], [constexpr]
```

34 Class Documentation

cast value in valueType

Template Parameters

```
valueType (double for example)
```

The documentation for this struct was generated from the following file:

· src/aerobus.h

5.20 aerobus::polynomial < Ring, variable_name >::val < coeffN, coeffs > Struct Template Reference

```
values (seen as types) in polynomial ring
```

Public Types

• using aN = coeffN

#include <aerobus.h>

heavy weight coefficient (non zero)

• using strip = val< coeffs... >

remove largest coefficient

• using is_zero_t = std::bool_constant<(degree==0) &&(aN::is_zero_t::value)>

true_type if polynomial is constant zero

template<size_t index>

using coeff_at_t = typename coeff_at< index >::type

type of coefficient at index

Static Public Member Functions

• static std::string to_string ()

get a string representation of polynomial

 $\bullet \ \ \mathsf{template} \mathord{<} \mathsf{typename} \ \mathsf{valueRing} >$

static constexpr valueRing eval (const valueRing &x)

evaluates polynomial seen as a function operating on ValueRing

Static Public Attributes

```
• static constexpr size_t degree = sizeof...(coeffs)
```

degree of the polynomial

static constexpr bool is_zero_v = is_zero_t::value

true if polynomial is constant zero

5.20.1 Detailed Description

```
template<typename Ring, char variable_name = 'x'>
template<typename coeffN, typename... coeffs>
struct aerobus::polynomial< Ring, variable_name >::val< coeffN, coeffs >
```

values (seen as types) in polynomial ring

36 Class Documentation

Template Parameters

coeffN	high degree coefficient
coeffs	lower degree coefficients

5.20.2 Member Typedef Documentation

5.20.2.1 coeff_at_t

```
template<typename Ring , char variable_name = 'x'>
template<typename coeffN , typename... coeffs>
template<size_t index>
using aerobus::polynomial< Ring, variable_name >::val< coeffN, coeffs >::coeff_at_t = typename
coeff_at<index>::type
```

type of coefficient at index

Template Parameters

index	
-------	--

5.20.3 Member Function Documentation

5.20.3.1 eval()

evaluates polynomial seen as a function operating on ValueRing

Template Parameters

valueRing	usually float or double
-----------	-------------------------

Parameters

x value

Returns

P(x)

5.20.3.2 to_string()

```
template<typename Ring , char variable_name = 'x'>
template<typename coeffN , typename... coeffs>
static std::string aerobus::polynomial< Ring, variable_name >::val< coeffN, coeffs >::to_
string ( ) [inline], [static]
```

get a string representation of polynomial

Returns

```
something like a_n X^n + ... + a_1 X + a_0
```

The documentation for this struct was generated from the following file:

src/aerobus.h

5.21 aerobus::Quotient < Ring, X >::val < V > Struct Template Reference

Public Types

using type = std::conditional_t< Ring::template pos_v< tmp >, tmp, typename Ring::template sub_t< typename Ring::zero, tmp > >

The documentation for this struct was generated from the following file:

· src/aerobus.h

5.22 aerobus::zpz::val< x > Struct Template Reference

Public Types

using is_zero_t = std::bool_constant< x% p==0 >

Static Public Member Functions

- template<typename valueType > static constexpr valueType get ()
- static std::string to_string ()
- template<typename valueRing >
 static constexpr valueRing eval (const valueRing &v)

Static Public Attributes

• static constexpr int32_t v = x % p

The documentation for this struct was generated from the following file:

src/aerobus.h

38 Class Documentation

5.23 aerobus::polynomial< Ring, variable_name >::val< coeffN > Struct Template Reference

specialization for constants

```
#include <aerobus.h>
```

Classes

- · struct coeff at
- struct coeff_at< index, std::enable_if_t<(index<0||index > 0)>>
- struct coeff at< index, std::enable if t<(index==0)>>

Public Types

```
• using aN = coeffN
```

- using strip = val< coeffN >
- using is_zero_t = std::bool_constant< aN::is_zero_t::value >
- template<size_t index>
 using coeff_at_t = typename coeff_at< index >::type

Static Public Member Functions

- static std::string to_string ()
- template<typename valueRing >
 static constexpr valueRing eval (const valueRing &x)

Static Public Attributes

- static constexpr size_t degree = 0
- static constexpr bool is_zero_v = is_zero_t::value

5.23.1 Detailed Description

```
template < typename Ring, char variable_name = 'x' > template < typename coeffN > struct aerobus::polynomial < Ring, variable_name >::val < coeffN > specialization for constants

Template Parameters

| CoeffN |
```

The documentation for this struct was generated from the following file:

• src/aerobus.h

5.24 aerobus::zpz Struct Template Reference

#include <aerobus.h>

Classes

struct val

Public Types

```
using inner_type = int32_t

    template<auto x>

  using inject_constant_t = val< static_cast< int32_t >(x)>
• using zero = val< 0 >
• using one = val< 1 >
• template<typename v1 , typename v2 >
  using add_t = typename add< v1, v2 >::type
     addition operator
• template<typename v1 , typename v2 >
  using sub_t = typename sub< v1, v2 >::type
     substraction operator
• template<typename v1 , typename v2 >
  using mul_t = typename mul < v1, v2 >::type
     multiplication operator
• template<typename v1 , typename v2 >
  using div_t = typename div < v1, v2 >::type
     division operator
• template<typename v1 , typename v2 >
  using mod_t = typename remainder < v1, v2 >::type
     modulo operator
• template<typename v1 , typename v2 >
  using gt_t = typename gt < v1, v2 >::type
     strictly greater operator (type)
• template<typename v1 , typename v2 >
  using It_t = typename It< v1, v2 >::type
     strictly smaller operator (type)
• template<typename v1 , typename v2 >
  using eq_t = typename eq< v1, v2 >::type
     equality operator (type)

    template<typename v1 , typename v2 >

  using gcd_t = gcd_t < i32, v1, v2 >
     greatest common divisor
• template<typename v1 >
  using pos_t = typename pos< v1 >::type
     positivity operator (type)
```

40 Class Documentation

Static Public Attributes

5.24.1 Detailed Description

```
template<int32_t p> struct aerobus::zpz
```

congruence classes of integers for a modulus if p is prime, zpz is a field, otherwise an integral domain with all related operations

The documentation for this struct was generated from the following file:

• src/aerobus.h

Chapter 6

File Documentation

```
00001 // -*- lsst-c++
00002 #ifndef __INC_AEROBUS__ // NOLINT
00003 #define __INC_AEROBUS__
00004
00005 #include <cstdint>
00006 #include <cstddef>
00007 #include <cstring>
00008 #include <type_traits>
00009 #include <utility>
00010 #include <algorithm>
00011 #include <functional>
00012 #include <string>
00013 #include <concepts> // NOLINT
00014 #include <array>
00015
00016
00017 #ifdef _MSC_VER
00018 #define ALIGNED(x) __declspec(align(x))
00019 #define INLINED __forceinline
00021 #define ALIGNED(x) __attribute__((aligned(x)))
00022 #define INLINED __attribute__((always_inline)) inline
00023 #endif
00024
00025 // aligned allocation
00026 namespace aerobus {
          template<typename T>
00034
          T* aligned_malloc(size_t count, size_t alignment) {
00035
              #ifdef _MSC_VER
00036
              return static_cast<T*>(_aligned_malloc(count * sizeof(T), alignment));
00037
              #else
00038
              return static_cast<T*>(aligned_alloc(alignment, count * sizeof(T)));
00039
00040
00041
00050
          template<typename T, size_t N>
00051
          constexpr bool contains (const std::array<T, N>& arr, const T& v) {
              for (const auto& vv : arr) {
00052
                  if (v == vv) {
00054
                       return true;
00055
00056
              }
00057
00058
              return false;
00059
00060
00061 } // namespace aerobus
00062
00063 // concepts
00064 namespace aerobus {
00066
          template <typename R>
00067
          concept IsRing = requires {
00068
              typename R::one;
00069
              typename R::zero;
00070
              typename R::template add_t<typename R::one, typename R::one>;
              typename R::template sub_t<typename R::one, typename R::one>;
00071
00072
              typename R::template mul_t<typename R::one, typename R::one>;
00073
```

```
00074
00076
          template <typename R>
00077
          concept IsEuclideanDomain = IsRing<R> && requires {
00078
              typename R::template div_t<typename R::one, typename R::one>;
00079
               typename R::template mod_t<typename R::one, typename R::one>;
               typename R::template gcd_t<typename R::one, typename R::one>;
typename R::template eq_t<typename R::one, typename R::one>;
00080
00082
               typename R::template pos_t<typename R::one>;
00083
00084
               R::template pos_v<typename R::one> == true;
              // typename R::template gt_t<typename R::one, typename R::zero>;
R::is_euclidean_domain == true;
00085
00086
00087
          };
00088
00090
          template<typename R>
          concept IsField = IsEuclideanDomain<R> && requires {
   R::is_field == true;
00091
00092
00093
00094 } // namespace aerobus
00095
00096 // utilities
00097 namespace aerobus {
00098
          namespace internal {
00099
              \label{template} \verb|template| < typename ...> typename TT, typename T>
00100
               struct is_instantiation_of : std::false_type { };
00101
00102
               template<template<typename...> typename TT, typename... Ts>
00103
               struct is_instantiation_of<TT, TT<Ts...» : std::true_type { };</pre>
00104
               template<template<typename...> typename TT, typename T>
00105
00106
              inline constexpr bool is_instantiation_of_v = is_instantiation_of<TT, T>::value;
00107
00108
               template <int64_t i, typename T, typename... Ts>
00109
               struct type_at {
                  static_assert(i < sizeof...(Ts) + 1, "index out of range");
using type = typename type_at<i - 1, Ts...>::type;
00110
00111
00112
              };
00113
00114
               template <typename T, typename... Ts> struct type_at<0, T, Ts...> {
00115
                  using type = T;
00116
               };
00117
00118
               template <size_t i, typename... Ts>
00119
               using type_at_t = typename type_at<i, Ts...>::type;
00120
00121
00122
               template<int32_t n, int32_t i, typename E = void>
00123
              struct _is_prime {};
00124
00125
               template<int32 t i>
00126
               struct _is_prime<1, i> {
00127
                  static constexpr bool value = false;
00128
00129
               template<int32 t i>
00130
               struct _is_prime<2, i> {
00131
                 static constexpr bool value = true;
00133
00134
00135
               template<int32_t i>
00136
               struct _is_prime<3, i> {
                  static constexpr bool value = true;
00137
00138
00139
00140
               template<int32_t i>
00141
               struct _is_prime<5, i> {
                  static constexpr bool value = true;
00142
00143
00144
00145
               template<int32_t i>
00146
               struct _is_prime<7, i> {
00147
                   static constexpr bool value = true;
00148
00149
               template<int32_t n, int32_t i> struct _is_prime<n, i, std::enable_if_t<(n != 2 && n % 2 == 0)>> {
00150
00151
00152
                  static constexpr bool value = false;
00153
00154
               00155
00156
                  static constexpr bool value = false;
00157
00158
00159
00160
               template<int32_t n, int32_t i>
               struct _is_prime<n, i, std::enable_if_t<(n >= 9 && i * i > n)» {
00161
00162
                   static constexpr bool value = true;
```

```
00163
              };
00164
00165
               template<int32_t n, int32_t i>
00166
               struct _is_prime<n, i, std::enable_if_t<(
00167
                  n % i == 0 &&
                   n >= 9 &&
00168
                  n % 3 != 0 &&
00169
00170
                   n % 2 != 0 &&
00171
                   i * i > n)» {
00172
                   static constexpr bool value = true;
00173
00174
              template<int32_t n, int32_t i>
struct _is_prime<n, i, std::enable_if_t<(</pre>
00175
00176
00177
                  n % (i+2) == 0 &&
00178
                   n >= 9 \& \&
                   n % 3 != 0 &&
00179
                   n % 2 != 0 &&
00180
00181
                   i * i <= n) » {
00182
                   static constexpr bool value = true;
00183
00184
00185
              template<int32_t n, int32_t i>
00186
              struct _is_prime<n, i, std::enable_if_t<( n % (i+2) != 0 &&
00187
                       n % i != 0 &&
00188
00189
                       n >= 9 &&
00190
                       n % 3 != 0 &&
                       n % 2 != 0 &&
00191
00192
                       (i * i \le n)) > {
00193
                   static constexpr bool value = _is_prime<n, i+6>::value;
00194
              };
00195
00196
          } // namespace internal
00197
00200
          template<int32_t n>
00201
          struct is prime {
00203
              static constexpr bool value = internal::_is_prime<n, 5>::value;
00204
00205
00206
          template<int32_t n>
00207
          static constexpr bool is_prime_v = is_prime<n>::value;
00208
00209
          namespace internal {
00210
              template <std::size_t... Is>
00211
              constexpr auto index_sequence_reverse(std::index_sequence<Is...> const&)
00212
                   -> decltype(std::index_sequence<sizeof...(Is) - 1U - Is...>{});
00213
00214
              template <std::size t N>
00215
              using make index sequence reverse
00216
                    = decltype(index_sequence_reverse(std::make_index_sequence<N>{}));
00217
00223
              template<typename Ring, typename E = void>
00224
              struct gcd;
00225
00226
               template<typename Ring>
              struct gcd<Ring, std::enable_if_t<Ring::is_euclidean_domain» {</pre>
00228
                   template<typename A, typename B, typename E = void>
00229
                   struct gcd_helper {};
00230
                   //B = 0.A > 0
00231
                   template<typename A, typename B>
struct gcd_helper<A, B, std::enable_if_t</pre>
00232
00233
00234
                       ((B::is_zero_t::value) &&
00235
                            (Ring::template gt_t<A, typename Ring::zero>::value))» {
00236
                       using type = A;
00237
                   };
00238
00239
                   // B = 0, A < 0
                   template<typename A, typename B>
00240
00241
                   struct gcd_helper<A, B, std::enable_if_t<
00242
                       ((B::is_zero_t::value) &&
00243
                           !(Ring::template gt_t<A, typename Ring::zero>::value))» {
00244
                       using type = typename Ring::template sub_t<typename Ring::zero, A>;
00245
                   };
00246
00247
                   // B != 0
                   template<typename A, typename B>
00248
                   struct gcd_helper<A, B, std::enable_if_t<
00249
00250
                       (!B::is_zero_t::value)
00251
                       » {
                   private: // NOLINT
00252
00253
                       // A / B
00254
                       using k = typename Ring::template div_t<A, B>;
00255
                       // A - (A/B) *B = A % B
00256
                       using m = typename Ring::template sub_t<A, typename Ring::template mul_t<k, B»;
00257
```

```
public:
00259
                      using type = typename gcd_helper<B, m>::type;
00260
                   };
00261
00262
                  template<typename A, typename B>
00263
                  using type = typename gcd_helper<A, B>::type;
00265
          } // namespace internal
00266
00269
          template<typename T, typename A, typename B>
          using gcd_t = typename internal::gcd<T>::template type<A, B>;
00270
00271 } // namespace aerobus
00272
00273 // quotient ring by the principal ideal generated by X
00274 namespace aerobus {
00275
          template<typename Ring, typename X>
00276
          requires IsRing<Ring>
00277
          struct Ouotient {
              template <typename V>
              struct val {
 private: // NOLINT
00279
00280
00281
                  using tmp = typename Ring::template mod_t<V, X>;
00282
00283
                public:
00284
                  using type = std::conditional_t<
00285
                     Ring::template pos_v<tmp>,
00286
00287
                       typename Ring::template sub_t<typename Ring::zero, tmp>
00288
00289
              };
00290
00291
              using zero = val<typename Ring::zero>;
00292
              using one = val<typename Ring::one>;
00293
00294
               template<typename v1, typename v2>
              using add_t = val<typename Ring::template add_t<typename v1::type, typename v2::type>>;
00295
00296
               template<typename v1, typename v2>
00297
              using mul_t = val<typename Ring::template mul_t<typename v1::type, typename v2::type>>;
00298
               template<typename v1, typename v2>
00299
               using div_t = val<typename Ring::template div_t<typename v1::type, typename v2::type>>;
00300
               template<typename v1, typename v2>
              using mod_t = val<typename Ring::template mod_t<typename v1::type, typename v2::type>>;
00301
00302
              template<typename v1, typename v2>
              using eq_t = typename Ring::template eq_t<typename v1::type, typename v2::type>;
00303
               template<typename v1, typename v2>
00304
00305
               static constexpr bool eq_v = Ring::template eq_t<typename v1::type, typename v2::type>::value;
00306
              template<typename v1>
00307
              using pos_t = std::true_type;
00308
00309
              template<tvpename v>
00310
              static constexpr bool pos_v = pos_t<v>::value;
00311
00312
              static constexpr bool is_euclidean_domain = true;
00313
00314
              template<auto x>
00315
              using inject constant t = val<typename Ring::template inject constant t<x>>;
00316
00317
               template<typename v>
00318
              using inject_ring_t = val<v>;
00319
          } ;
00320 } // namespace aerobus
00321
00322 // type_list
00323 namespace aerobus {
00325
          template <typename... Ts>
00326
          struct type_list;
00327
00328
          namespace internal {
              template <typename T, typename... Us>
00329
00330
              struct pop_front_h {
                 using tail = type_list<Us...>;
using head = T;
00331
00332
00333
              };
00334
00335
              template <size_t index, typename L1, typename L2>
00336
               struct split_h {
00337
00338
                  static_assert(index <= L2::length, "index ouf of bounds");</pre>
                  using a = typename L2::pop_front::type;
using b = typename L2::pop_front::tail;
00339
00340
                  using c = typename L1::template push_back<a>;
00341
00342
00343
               public:
                  using head = typename split_h<index - 1, c, b>::head; using tail = typename split_h<index - 1, c, b>::tail;
00344
00345
00346
              };
00347
```

```
template <typename L1, typename L2>
00349
               struct split_h<0, L1, L2> {
00350
                   using head = L1;
00351
                   using tail = L2;
00352
00353
               template <size_t index, typename L, typename T>
00355
               struct insert_h {
00356
                  static_assert(index <= L::length, "index ouf of bounds");</pre>
00357
                   using s = typename L::template split<index>;
                   using left = typename s::head;
using right = typename s::tail;
00358
00359
                   using ll = typename left::template push_back<T>;
00360
00361
                   using type = typename ll::template concat<right>;
00362
               };
00363
00364
               template <size_t index, typename L>
00365
               struct remove_h {
00366
                   using s = typename L::template split<index>;
00367
                   using left = typename s::head;
00368
                    using right = typename s::tail;
00369
                   using rr = typename right::pop_front::tail;
                   using type = typename left::template concat<rr>;
00370
00371
          };
} // namespace internal
00372
00373
00374
00377
          template <typename... Ts>
00378
          struct type_list {
00379
           private:
00380
               template <typename T>
00381
               struct concat_h;
00382
00383
               template <typename... Us>
00384
               struct concat_h<type_list<Us...» {</pre>
00385
                   using type = type_list<Ts..., Us...>;
00386
               };
00387
00388
00390
              static constexpr size_t length = sizeof...(Ts);
00391
00394
               \texttt{template} \; \texttt{<typename} \; \; \texttt{T>}
00395
               using push_front = type_list<T, Ts...>;
00396
00399
               template <size_t index>
00400
               using at = internal::type_at_t<index, Ts...>;
00401
               struct pop_front {
   using type = typename internal::pop_front_h<Ts...>::head;
   using tail = typename internal::pop_front_h<Ts...>::tail;
00403
00405
00407
00408
               };
00409
00412
               template <typename T>
00413
               using push_back = type_list<Ts..., T>;
00414
00417
               template <typename U>
00418
               using concat = typename concat_h<U>::type;
00419
00422
               template <size_t index>
00423
               struct split {
00424
               private:
00425
                   using inner = internal::split_h<index, type_list<>, type_list<Ts...»;</pre>
00426
00427
00428
                   using head = typename inner::head;
00429
                   using tail = typename inner::tail;
00430
00431
00435
               template <typename T, size_t index>
00436
               using insert = typename internal::insert_h<index, type_list<Ts...>, T>::type;
00437
00440
               template <size_t index>
00441
               using remove = typename internal::remove_h<index, type_list<Ts...»::type;</pre>
00442
          };
00443
00444
           template <>
00445
          struct type_list<> {
00446
               static constexpr size_t length = 0;
00447
00448
               template <typename T>
               using push_front = type_list<T>;
00449
00450
00451
               template <typename T>
00452
               using push_back = type_list<T>;
00453
00454
               template <typename U> \,
00455
               using concat = U:
```

```
00456
00457
              // TODO(jewave): assert index == 0
             template <typename T, size_t index>
using insert = type_list<T>;
00458
00459
00460
          };
00461 } // namespace aerobus
00462
00463 // i32
00464 namespace aerobus {
00466
          struct i32 {
00467
              using inner_type = int32_t;
              template<int32_t x>
00470
00471
              struct val {
                  static constexpr int32_t v = x;
00473
00474
00477
                  template<typename valueType>
                  static constexpr valueType get() { return static_cast<valueType>(x); }
00478
00479
00481
                  using is_zero_t = std::bool_constant<x == 0>;
00482
00484
                  static std::string to_string() {
00485
                      return std::to_string(x);
                  }
00486
00487
00490
                  template<typename valueRing>
00491
                  static constexpr valueRing eval(const valueRing& v) {
00492
                      return static_cast<valueRing>(x);
00493
00494
              } ;
00495
00497
              using zero = val<0>:
00499
              using one = val<1>;
00501
              static constexpr bool is_field = false;
00503
              static constexpr bool is_euclidean_domain = true;
00507
              template<auto x>
              using inject_constant_t = val<static_cast<int32_t>(x)>;
00508
00509
00510
              template<typename v>
00511
              using inject_ring_t = v;
00512
           private:
00513
              template<typename v1, typename v2>
00514
00515
              struct add {
00516
                  using type = val<v1::v + v2::v>;
00517
00518
00519
              template<typename v1, typename v2>
              struct sub {
00520
                  using type = val<v1::v - v2::v>;
00521
00522
00523
00524
              template<typename v1, typename v2>
00525
              struct mul {
00526
                 using type = val<v1::v* v2::v>;
00527
00528
00529
              template<typename v1, typename v2>
00530
00531
                  using type = val<v1::v / v2::v>;
00532
00533
              template<typename v1, typename v2>
00534
00535
              struct remainder {
00536
                 using type = val<v1::v % v2::v>;
00537
00538
00539
              template<typename v1, typename v2>
00540
              struct at {
                  using type = std::conditional_t<(v1::v > v2::v), std::true_type, std::false_type>;
00541
00542
00543
00544
              template<typename v1, typename v2>
00545
              struct lt {
                  using type = std::conditional_t<(v1::v < v2::v), std::true_type, std::false_type>;
00546
00547
00548
00549
              template<typename v1, typename v2>
00550
                  using type = std::conditional_t<(v1::v == v2::v), std::true_type, std::false_type>;
00551
00552
00553
00554
              template<typename v1>
00555
              struct pos {
00556
                  using type = std::bool_constant<(v1::v > 0)>;
00557
00558
00559
           public:
```

```
template<typename v1, typename v2>
00562
              using add_t = typename add<v1, v2>::type;
00563
00565
              template<typename v1, typename v2>
              using sub_t = typename sub<v1, v2>::type;
00566
00567
00569
              template<typename v1, typename v2>
00570
              using mul_t = typename mul<v1, v2>::type;
00571
00573
              template<typename v1, typename v2>
00574
              using div_t = typename div<v1, v2>::type;
00575
00577
              template<typename v1, typename v2>
00578
              using mod_t = typename remainder<v1, v2>::type;
00579
00581
              template<typename v1, typename v2>
00582
              using gt_t = typename gt<v1, v2>::type;
00583
              template<typename v1, typename v2>
00586
              using lt_t = typename lt<v1, v2>::type;
00587
00589
              template<typename v1, typename v2> ^{\circ}
00590
              using eq_t = typename eq<v1, v2>::type;
00591
00595
              template<typename v1, typename v2>
00596
              static constexpr bool eq_v = eq_t<v1, v2>::value;
00597
00599
              template<typename v1, typename v2>
00600
              using gcd_t = gcd_t < i32, v1, v2>;
00601
00603
              template < typename v >
00604
              using pos_t = typename pos<v>::type;
00605
00608
              {\tt template}{<}{\tt typename}\ {\tt v}{>}
00609
              static constexpr bool pos_v = pos_t<v>::value;
00610
          };
00611 } // namespace aerobus
00612
00613 // i64
00614 namespace aerobus {
00616
          struct i64 {
00617
              using inner_type = int64_t;
              template<int64_t x>
00620
00621
              struct val {
00622
                  static constexpr int64_t v = x;
00623
00626
                  template<typename valueType>
                  static constexpr valueType get() { return static_cast<valueType>(x); }
00627
00628
00630
                  using is zero t = std::bool constant<x == 0>;
00631
00633
                  static std::string to_string() {
00634
                      return std::to_string(x);
00635
00636
00639
                  template<typename valueRing>
                  static constexpr valueRing eval(const valueRing& v) {
00641
                      return static_cast<valueRing>(x);
00642
00643
              };
00644
00648
              template<auto x>
00649
              using inject_constant_t = val<static_cast<int64_t>(x)>;
00650
00651
              template<typename v>
00652
              using inject_ring_t = v;
00653
00655
              using zero = val<0>:
              using one = val<1>;
00657
              static constexpr bool is_field = false;
00661
              static constexpr bool is_euclidean_domain = true;
00662
           private:
00663
              template<typename v1, typename v2>
00664
00665
              struct add {
                  using type = val<v1::v + v2::v>;
00666
00667
00668
00669
              template<typename v1, typename v2>
00670
              struct sub {
00671
                  using type = val<v1::v - v2::v>;
00672
00673
00674
              template<typename v1, typename v2>
00675
              struct mul {
                  using type = val<v1::v* v2::v>;
00676
00677
              };
```

```
00679
              template<typename v1, typename v2>
              struct div {
00680
                 using type = val<v1::v / v2::v>;
00681
00682
00683
00684
              template<typename v1, typename v2>
00685
              struct remainder {
00686
                 using type = val<v1::v% v2::v>;
00687
00688
00689
              template<typename v1, typename v2>
00690
              struct qt {
00691
                  using type = std::conditional_t<(v1::v > v2::v), std::true_type, std::false_type>;
00692
00693
00694
              template<typename v1, typename v2>
00695
              struct lt {
00696
                  using type = std::conditional_t<(v1::v < v2::v), std::true_type, std::false_type>;
00697
00698
00699
              template<typename v1, typename v2>
00700
              struct eq {
00701
                  using type = std::conditional_t<(v1::v == v2::v), std::true_type, std::false_type>;
00702
00703
              template<typename v>
00704
00705
              struct pos {
00706
                 using type = std::bool_constant<(v::v > 0)>;
00707
00708
00709
           public:
00713
              template<typename v1, typename v2>
00714
              using add_t = typename add<v1, v2>::type;
00715
00719
              template<typename v1, typename v2> ^{\circ}
00720
              using sub_t = typename sub<v1, v2>::type;
00721
00725
              template<typename v1, typename v2>
00726
              using mul_t = typename mul<v1, v2>::type;
00727
00731
              template<typename v1, typename v2>
00732
              using div t = typename div<v1, v2>::type;
00733
00737
              template<typename v1, typename v2>
00738
              using mod_t = typename remainder<v1, v2>::type;
00739
00743
              template<typename v1, typename v2>
00744
              using gt_t = typename gt<v1, v2>::type;
00745
00749
              template<typename v1, typename v2>
00750
              static constexpr bool gt_v = gt_t<v1, v2>::value;
00751
00755
              template<typename v1, typename v2>
00756
              using lt_t = typename lt<v1, v2>::type;
00757
00761
              template<typename v1, typename v2>
00762
              static constexpr bool lt_v = lt_t<v1, v2>::value;
00763
00767
              template<typename v1, typename v2>
00768
              using eq_t = typename eq<v1, v2>::type;
00769
00773
              template<typename v1, typename v2>
00774
              static constexpr bool eq_v = eq_t<v1, v2>::value;
00775
00779
              template<typename v1, typename v2>
00780
              using gcd_t = gcd_t < i64, v1, v2>;
00781
00784
              template<typename v>
00785
              using pos_t = typename pos<v>::type;
00786
00789
              template < typename v >
00790
              static constexpr bool pos_v = pos_t<v>::value;
00791
          };
00792 }
        // namespace aerobus
00793
00794 // z/pz
00795 namespace aerobus {
00800
          template<int32_t p>
00801
          struct zpz {
             using inner_type = int32_t;
00802
00803
              template<int32_t x>
00804
              struct val {
00805
                  static constexpr int32_t v = x % p;
00806
00807
                  template<typename valueType>
                  static constexpr valueType get() { return static_cast<valueType>(x % p); }
00808
```

```
00809
00810
                  using is_zero_t = std::bool_constant<x% p == 0>;
00811
                  static std::string to_string() {
00812
                     return std::to_string(x % p);
00813
00814
                  template<typename valueRing>
00816
                  static constexpr valueRing eval(const valueRing& v) {
00817
                     return static_cast<valueRing>(x % p);
00818
00819
              };
00820
00821
              template<auto x>
              using inject_constant_t = val<static_cast<int32_t>(x)>;
00822
00823
00824
              using zero = val<0>;
              using one = val<1>:
00825
00826
              static constexpr bool is field = is prime::value;
00827
              static constexpr bool is_euclidean_domain = true;
00828
00829
00830
              template<typename v1, typename v2>
00831
              struct add {
                  using type = val<(v1::v + v2::v) % p>;
00832
00833
00834
              template<typename v1, typename v2>
00835
00836
              struct sub {
                 using type = val<(v1::v - v2::v) % p>;
00837
00838
00839
00840
              template<typename v1, typename v2>
00841
00842
                  using type = val<(v1::v* v2::v) % p>;
00843
00844
00845
              template<typename v1, typename v2>
              struct div {
00847
                 using type = val<(v1::v% p) / (v2::v % p)>;
00848
00849
00850
              template<typename v1, typename v2>
00851
              struct remainder {
00852
                  using type = val<(v1::v% v2::v) % p>;
00853
00854
00855
              template<typename v1, typename v2>
00856
              struct gt {
                  using type = std::conditional_t<(v1::v% p > v2::v% p), std::true_type, std::false_type>;
00857
00858
00859
00860
              template<typename v1, typename v2>
00861
              struct lt {
00862
                 using type = std::conditional_t<(v1::v% p < v2::v% p), std::true_type, std::false_type>;
00863
00864
00865
              template<typename v1, typename v2>
00866
              struct eq {
00867
                  using type = std::conditional_t<(v1::v% p == v2::v % p), std::true_type, std::false_type>;
00868
00869
00870
              template<typename v1>
00871
              struct pos {
00872
                 using type = std::bool_constant<(v1::v > 0)>;
00873
              };
00874
00875
           public:
00877
              template<typename v1, typename v2>
00878
              using add_t = typename add<v1, v2>::type;
00879
00881
              template<typename v1, typename v2>
00882
              using sub_t = typename sub<v1, v2>::type;
00883
00885
              template<typename v1, typename v2>
00886
              using mul_t = typename mul<v1, v2>::type;
00887
00889
              template<typename v1, typename v2>
00890
              using div_t = typename div<v1, v2>::type;
00891
00893
              template<typename v1, typename v2>
00894
              using mod_t = typename remainder<v1, v2>::type;
00895
00897
              template<typename v1, typename v2>
00898
              using gt_t = typename gt<v1, v2>::type;
00899
              template<typename v1, typename v2>
static constexpr bool gt_v = gt_t<v1, v2>::value;
00901
00902
```

```
template<typename v1, typename v2>
using lt_t = typename lt<v1, v2>::type;
00905
00906
00907
00909
              template<typename v1, typename v2>
static constexpr bool lt_v = lt_t<v1, v2>::value;
00910
00911
00913
               template<typename v1, typename v2>
00914
              using eq_t = typename eq<v1, v2>::type;
00915
              template<typename v1, typename v2>
static constexpr bool eq_v = eq_t<v1, v2>::value;
00917
00918
00919
00921
               template<typename v1, typename v2>
00922
              using gcd_t = gcd_t<i32, v1, v2>;
00923
00925
              template<typename v1>
00926
              using pos_t = typename pos<v1>::type;
00929
              template<typename v>
00930
              static constexpr bool pos_v = pos_t<v>::value;
00931
          } ;
00932 } // namespace aerobus
00933
00934 // polynomial
00935 namespace aerobus {
00936
          // coeffN x^N + ...
00941
          template<typename Ring, char variable_name = 'x'>
00942
          requires IsEuclideanDomain<Ring>
00943
          struct polynomial {
00944
              static constexpr bool is_field = false;
00945
              static constexpr bool is_euclidean_domain = Ring::is_euclidean_domain;
00946
              template<typename coeffN, typename... coeffs>
00950
               struct val {
00951
00953
                  static constexpr size_t degree = sizeof...(coeffs);
00955
                   using aN = coeffN;
                  using strip = val<coeffs...>;
00959
                  using is_zero_t = std::bool_constant<(degree == 0) && (aN::is_zero_t::value)>;
00961
                  static constexpr bool is_zero_v = is_zero_t::value;
00962
00963
               private:
                  template<size_t index, typename E = void>
00964
00965
                  struct coeff_at {};
00966
00967
                  template<size_t index>
00968
                  struct coeff_at<index, std::enable_if_t<(index >= 0 && index <= sizeof...(coeffs))» {</pre>
00969
                       using type = internal::type_at_t<sizeof...(coeffs) - index, coeffN, coeffs...>;
00970
00971
00972
                  template<size_t index>
00973
                  struct coeff_at<index, std::enable_if_t<(index < 0 || index > sizeof...(coeffs))» {
00974
                       using type = typename Ring::zero;
00975
00976
00977
                public:
00980
                  template<size_t index>
00981
                  using coeff_at_t = typename coeff_at<index>::type;
00982
00985
                   static std::string to_string() {
                       return string_helper<coeffN, coeffs...>::func();
00986
00987
                   }
00988
00993
                  template<typename valueRing>
00994
                   static constexpr valueRing eval(const valueRing& x) {
00995
                       return horner_evaluation<valueRing, val>
00996
                               ::template inner<0, degree + 1>
00997
                                ::func(static_cast<valueRing>(0), x);
00998
00999
              };
01000
01003
              template<typename coeffN>
01004
               struct val<coeffN> {
01005
                  static constexpr size_t degree = 0;
01006
                   using aN = coeffN;
01007
                   using strip = val<coeffN>;
01008
                  using is_zero_t = std::bool_constant<aN::is_zero_t::value>;
01009
                  static constexpr bool is_zero_v = is_zero_t::value;
01010
01011
01012
                  template<size t index, typename E = void>
01013
                  struct coeff_at {};
01014
01015
                   template<size_t index>
01016
                   struct coeff_at<index, std::enable_if_t<(index == 0)» {</pre>
01017
                       using type = aN;
01018
                   };
```

```
01019
01020
                  template<size_t index>
01021
                  struct coeff_at<index, std::enable_if_t<(index < 0 || index > 0)» {
01022
                      using type = typename Ring::zero;
01023
01024
01025
                  template<size_t index>
01026
                  using coeff_at_t = typename coeff_at<index>::type;
01027
01028
                  static std::string to_string() {
01029
                      return string_helper<coeffN>::func();
01030
01031
01032
                  template<typename valueRing>
01033
                  static constexpr valueRing eval(const valueRing& x) {
01034
                      return static_cast<valueRing>(aN::template get<valueRing>());
01035
                  }
01036
              };
01037
01039
              using zero = val<typename Ring::zero>;
01041
              using one = val<typename Ring::one>;
01043
              using X = val<typename Ring::one, typename Ring::zero>;
01044
01045
01046
              template<typename P, typename E = void>
01047
              struct simplify;
01048
01049
              template <typename P1, typename P2, typename I>
01050
              struct add_low;
01051
01052
              template<typename P1, typename P2>
01053
              struct add {
01054
                 using type = typename simplify<typename add_low<
01055
                  P1,
01056
                  P2,
                  internal::make_index_sequence_reverse<</pre>
01057
01058
                  std::max(P1::degree, P2::degree) + 1
                  »::type>::type;
01060
01061
01062
              template <typename P1, typename P2, typename I>
01063
              struct sub low;
01064
01065
              template <typename P1, typename P2, typename I>
01066
              struct mul_low;
01067
01068
              template<typename v1, typename v2>
01069
              struct mul {
01070
                      using type = typename mul_low<
01071
01072
                           v2
01073
                           internal::make_index_sequence_reverse<</pre>
01074
                          v1::degree + v2::degree + 1
01075
                          »::type;
01076
              };
01077
01078
              template<typename coeff, size_t deg>
01079
              struct monomial:
01080
01081
              template<typename v, typename E = void>
01082
              struct derive_helper {};
01083
01084
              template<typename v>
01085
              struct derive_helper<v, std::enable_if_t<v::degree == 0» {</pre>
01086
                  using type = zero;
01087
01088
01089
              template<tvpename v>
01090
              struct derive_helper<v, std::enable_if_t<v::degree != 0» {</pre>
                  using type = typename add<
01092
                      typename derive_helper<typename simplify<typename v::strip>::type>::type,
                       typename monomial<
01093
01094
                          typename Ring::template mul_t<</pre>
01095
                               typename v::aN,
01096
                               typename Ring::template inject_constant_t<(v::degree)>
01097
01098
                          v::degree - 1
01099
                      >::type
01100
                  >::type;
01101
              }:
01102
01103
              template<typename v1, typename v2, typename E = void>
01104
              struct eq_helper {};
01105
01106
              template<typename v1, typename v2> \,
              struct eq_helper<v1, v2, std::enable_if_t<v1::degree != v2::degree» {</pre>
01107
                  using type = std::false_type;
01108
```

```
01109
               };
01110
01111
01112
               template<typename v1, typename v2>
               struct eq_helper<v1, v2, std::enable_if_t<
   v1::degree == v2::degree &&</pre>
01113
01114
                   (v1::degree != 0 || v2::degree != 0) &&
01115
01116
                   std::is_same<
01117
                   typename Ring::template eq_t<typename v1::aN, typename v2::aN>,
01118
                   std::false_type
01119
                   >::value
01120
              > {
01121
01122
                   using type = std::false_type;
01123
               };
01124
               template<typename v1, typename v2>
struct eq_helper<v1, v2, std::enable_if_t<
    v1::degree == v2::degree &&</pre>
01125
01126
01127
01128
                   (v1::degree != 0 || v2::degree != 0) &&
01129
                   std::is_same<
01130
                   typename Ring::template eq_t<typename v1::aN, typename v2::aN>,
01131
                   std::true_type
01132
                   >::value
               » {
01133
01134
                   using type = typename eq_helper<typename v1::strip, typename v2::strip>::type;
01135
               } ;
01136
01137
               template<typename v1, typename v2>
01138
               struct eq_helper<v1, v2, std::enable_if_t<
    v1::degree == v2::degree &&</pre>
01139
01140
                   (v1::degree == 0)
01141
01142
                   using type = typename Ring::template eq_t<typename v1::aN, typename v2::aN>;
01143
01144
01145
               template<typename v1, typename v2, typename E = void>
01146
               struct lt_helper {};
01147
01148
               template<typename v1, typename v2>
01149
               struct lt_helper<v1, v2, std::enable_if_t<(v1::degree < v2::degree)» {</pre>
                   using type = std::true_type;
01150
01151
01152
01153
               template<typename v1, typename v2>
01154
               struct lt_helper<v1, v2, std::enable_if_t<(v1::degree == v2::degree)» {</pre>
01155
                  using type = typename Ring::template lt_t<typename v1::aN, typename v2::aN>;
01156
               };
01157
01158
               template<typename v1, typename v2> \,
               struct lt_helper<v1, v2, std::enable_if_t<(v1::degree > v2::degree)» {
01159
01160
                   using type = std::false_type;
01161
01162
               template<typename v1, typename v2, typename E = void>
01163
01164
               struct at helper {};
01165
01166
               template<typename v1, typename v2>
01167
               struct gt_helper<v1, v2, std::enable_if_t<(v1::degree > v2::degree)» {
01168
                   using type = std::true_type;
01169
01170
01171
               template<typename v1, typename v2>
01172
               struct gt_helper<v1, v2, std::enable_if_t<(v1::degree == v2::degree)» {</pre>
01173
                   using type = std::false_type;
01174
01175
01176
               template<typename v1, typename v2>
               struct gt_helper<v1, v2, std::enable_if_t<(v1::degree < v2::degree)» {
01177
                  using type = std::false_type;
01178
01179
01180
               \ensuremath{//} when high power is zero : strip
01181
01182
               template<tvpename P>
               struct simplify<P, std::enable_if_t<
01183
                  std::is_same<
01184
01185
                   typename Ring::zero,
01186
                   typename P::aN
01187
                   >::value && (P::degree > 0)
01188
               » {
01189
                   using type = typename simplify<typename P::strip>::type;
01190
               } ;
01191
01192
               // otherwise : do nothing
01193
               template<typename P>
01194
               struct simplify<P, std::enable_if_t<
01195
                   !std::is same<
```

```
typename Ring::zero,
01197
                 typename P::aN
01198
                 >::value && (P::degree > 0)
01199
01200
                 using type = P;
01201
             };
01202
01203
             // do not simplify constants
01204
             template<typename P>
01205
             struct simplify<P, std::enable_if_t<P::degree == 0» {</pre>
01206
                 using type = P;
01207
01208
01209
             // addition at
01210
             template<typename P1, typename P2, size_t index>
01211
             struct add_at {
01212
                 using type =
                     typename Ring::template add_t<</pre>
01213
01214
                         typename P1::template coeff_at_t<index>,
01215
                         typename P2::template coeff_at_t<index>>;
01216
01217
01218
             template<typename P1, typename P2, size_t index>
01219
             using add_at_t = typename add_at<P1, P2, index>::type;
01220
01221
             template<typename P1, typename P2, std::size_t... I>
01222
             struct add_low<P1, P2, std::index_sequence<I...» {</pre>
01223
                 using type = val<add_at_t<P1, P2, I>...>;
01224
01225
             // substraction at
01226
01227
             template<typename P1, typename P2, size_t index>
01228
             struct sub_at {
01229
                 using type =
01230
                     typename Ring::template sub_t<</pre>
                         typename P1::template coeff_at_t<index>,
01231
                         typename P2::template coeff_at_t<index>>;
01232
             };
01234
01235
             template<typename P1, typename P2, size_t index>
01236
             using sub_at_t = typename sub_at<P1, P2, index>::type;
01237
01238
             template<typename P1, typename P2, std::size_t... I>
             struct sub_low<P1, P2, std::index_sequence<I...» {
01239
01240
                 using type = val<sub_at_t<P1, P2, I>...>;
01241
01242
01243
             template<typename P1, typename P2>
01244
             struct sub {
01245
                using type = typename simplify<typename sub_low<
01246
                 P1,
01247
01248
                 internal::make_index_sequence_reverse<
01249
                 std::max(P1::degree, P2::degree) + 1
01250
                 »::type>::type;
01251
             };
01252
01253
             // multiplication at
01254
             template<typename v1, typename v2, size_t k, size_t index, size_t stop>
01255
             struct mul_at_loop_helper {
01256
                 using type = typename Ring::template add_t<
                     typename Ring::template mul_t<</pre>
01257
01258
                     typename v1::template coeff_at_t<index>,
01259
                     typename v2::template coeff_at_t<k - index>
01260
01261
                     typename mul_at_loop_helper<v1, v2, k, index + 1, stop>::type
01262
01263
             };
01264
01265
             template<typename v1, typename v2, size_t k, size_t stop>
01266
             struct mul_at_loop_helper<v1, v2, k, stop, stop> {
01267
                 using type = typename Ring::template mul_t<</pre>
01268
                     typename v1::template coeff_at_t<stop>,
01269
                     typename v2::template coeff_at_t<0>>;
01270
             };
01271
01272
             template <typename v1, typename v2, size_t k, typename E = void>
01273
             struct mul_at {};
01274
01275
             01276
                 using type = typename Ring::zero;
01278
01279
             01280
01281
01282
                 using type = typename mul_at_loop_helper<v1, v2, k, 0, k>::type;
```

```
01283
01284
01285
              template<typename P1, typename P2, size_t index>
01286
              using mul_at_t = typename mul_at<P1, P2, index>::type;
01287
              template<typename P1, typename P2, std::size_t... I>
struct mul_low<P1, P2, std::index_sequence<I...» {</pre>
01288
01289
01290
                  using type = val<mul_at_t<P1, P2, I>...>;
01291
01292
              // division helper
01293
              template< typename A, typename B, typename Q, typename R, typename E = void > 0
01294
01295
              struct div_helper {};
01296
01297
               template<typename A, typename B, typename Q, typename R>
              struct div_helper<A, B, Q, R, std::enable_if_t<
    (R::degree < B::degree) ||</pre>
01298
01299
                   (R::degree == 0 && std::is_same<typename R::aN, typename Ring::zero>::value)» {
01300
01301
                  using q_type = Q;
01302
                  using mod_type = R;
01303
                  using gcd_type = B;
01304
01305
01306
              template<typename A, typename B, typename Q, typename R>
struct div_helper<A, B, Q, R, std::enable_if_t</pre>
01307
                (R::degree >= B::degree) &&
01308
01309
                   !(R::degree == 0 && std::is_same<typename R::aN, typename Ring::zero>::value)» {
01310
               private: // NOLINT
01311
                  using rN = typename R::aN;
                   using bN = typename B::aN;
01312
                  using pT = typename monomial<typename Ring::template div_t<rN, bN>, R::degree -
01313
     B::degree>::type;
01314
                 using rr = typename sub<R, typename mul<pT, B>::type>::type;
                  using qq = typename add<Q, pT>::type;
01315
01316
               public:
01317
01318
                  using q_type = typename div_helper<A, B, qq, rr>::q_type;
                  using mod_type = typename div_helper<A, B, qq, rr>::mod_type;
01319
                  using gcd_type = rr;
01320
01321
01322
01323
              template<typename A, typename B>
01324
              struct div {
01325
                  static_assert (Ring::is_euclidean_domain, "cannot divide in that type of Ring");
                   using q_type = typename div_helper<A, B, zero, A>::q_type;
01326
01327
                   using m_type = typename div_helper<A, B, zero, A>::mod_type;
01328
              };
01329
01330
              template<tvpename P>
01331
              struct make unit {
01332
                  using type = typename div<P, val<typename P::aN>>::q_type;
01333
01334
01335
              template<typename coeff, size_t deg>
01336
              struct monomial {
01337
                  using type = typename mul<X, typename monomial<coeff, deg - 1>::type>::type;
01338
01339
01340
              template<typename coeff>
01341
              struct monomial < coeff, 0 > {
01342
                  using type = val<coeff>;
01343
              };
01344
01346
              template<typename valueRing, typename P>
01347
              struct horner_evaluation {
01348
                  template<size_t index, size_t stop>
01349
                   struct inner {
                       static constexpr valueRing func(const valueRing& accum, const valueRing& x) {
01350
01351
                           constexpr valueRing coeff =
01352
                               static_cast<valueRing>(P::template coeff_at_t<P::degree - index>::template
      get<valueRing>());
01353
                           return horner_evaluation<valueRing, P>::template inner<index + 1, stop>::func(x *
      accum + coeff, x);
01354
01355
                  };
01356
01357
                   template<size_t stop>
01358
                   struct inner<stop, stop> {
01359
                       static constexpr valueRing func(const valueRing& accum, const valueRing& x) {
01360
                           return accum:
01361
01362
                  };
01363
01364
01365
              template<typename coeff, typename... coeffs>
01366
              struct string helper {
                  static std::string func() {
01367
```

```
std::string tail = string_helper<coeffs...>::func();
                      std::string result = "";
01369
01370
                      if (Ring::template eq_t<coeff, typename Ring::zero>::value) {
                      return tail;
} else if (Ring::template eq_t<coeff, typename Ring::one>::value) {
01371
01372
01373
                          if (sizeof...(coeffs) == 1) {
01374
                              result += std::string(1, variable_name);
01375
                          } else {
01376
                             result += std::string(1, variable_name) + "^" +
     std::to_string(sizeof...(coeffs));
01377
                          }
01378
                      } else {
01379
                          if (sizeof...(coeffs) == 1) {
01380
                              result += coeff::to_string() + " " + std::string(1, variable_name);
01381
                          } else {
                             01382
01383
                                      + "^" + std::to_string(sizeof...(coeffs));
01384
01385
                          }
01386
01387
                      if (!tail.empty()) {
    result += " + " + tail;
01388
01389
01390
01391
01392
                      return result;
01393
                  }
01394
             };
01395
01396
              template<typename coeff>
01397
              struct string helper<coeff> {
01398
                  static std::string func() {
01399
                     if (!std::is_same<coeff, typename Ring::zero>::value) {
01400
                          return coeff::to_string();
01401
                      } else {
                          return "";
01402
01403
01404
                  }
01405
              };
01406
01407
           public:
01410
              template<typename P>
01411
              using simplify t = typename simplify<P>::type;
01412
              template<typename v1, typename v2>
01416
01417
              using add_t = typename add<v1, v2>::type;
01418
01422
              template<typename v1, typename v2>
01423
              using sub_t = typename sub<v1, v2>::type;
01424
01428
              template<typename v1, typename v2>
01429
              using mul_t = typename mul<v1, v2>::type;
01430
01434
              template<typename v1, typename v2>
01435
              using eq_t = typename eq_helper<v1, v2>::type;
01436
              template<typename v1, typename v2>
01441
              using lt_t = typename lt_helper<v1, v2>::type;
01442
01446
              template<typename v1, typename v2>
01447
              using gt_t = typename gt_helper<v1, v2>::type;
01448
01452
              template<typename v1, typename v2>
              using div_t = typename div<v1, v2>::q_type;
01453
01454
01458
              template<typename v1, typename v2>
01459
              using mod_t = typename div_helper<v1, v2, zero, v1>::mod_type;
01460
01464
              template<typename coeff, size_t deg>
01465
              using monomial_t = typename monomial<coeff, deg>::type;
01466
01469
              template < typename v >
01470
              using derive_t = typename derive_helper<v>::type;
01471
01474
              template<typename v>
01475
              using pos_t = typename Ring::template pos_t<typename v::aN>;
01476
01477
              template<typename v>
01478
              static constexpr bool pos_v = pos_t<v>::value;
01479
01483
              template<typename v1, typename v2>
01484
              using gcd_t = std::conditional_t<
01485
                  Ring::is_euclidean_domain,
01486
                  typename make_unit<gcd_t<polynomial<Ring, variable_name>, v1, v2»::type,
01487
                  void>;
01488
01492
              template<auto x>
```

```
using inject_constant_t = val<typename Ring::template inject_constant_t<x>>;
01494
01498
              template<typename v>
01499
              using inject_ring_t = val<v>;
01500
          };
01501 } // namespace aerobus
01502
01503 // fraction field
01504 namespace aerobus {
01505
          namespace internal {
              template<typename Ring, typename E = void>
01506
01507
              requires IsEuclideanDomain<Ring>
01508
              struct _FractionField {};
01509
01510
              template<typename Ring>
01511
              requires IsEuclideanDomain<Ring>
              struct _FractionField<Ring, std::enable_if_t<Ring::is_euclidean_domain> {
01512
                  static constexpr bool is_field = true;
static constexpr bool is_euclidean_domain = true;
01514
01515
01516
01517
01518
                  template<typename val1, typename val2, typename E = void>
01519
                  struct to_string_helper {};
01520
01521
                  template<typename val1, typename val2>
                  struct to_string_helper <val1, val2,
01522
01523
                       std::enable_if_t<
01524
                      Ring::template eq_t<
01525
                      val2, typename Ring::one
01526
                      >::value
01527
01528
                  > {
01529
                      static std::string func() {
01530
                          return vall::to_string();
01531
                  };
01532
01533
01534
                  template<typename val1, typename val2>
01535
                  struct to_string_helper<val1, val2,
01536
                     std::enable_if_t<
01537
                      !Ring::template eq_t<
01538
                      val2,
                      typename Ring::one
01539
01540
                      >::value
01541
01542
                  > {
01543
                      static std::string func() {
                          return "(" + val1::to_string() + ") / (" + val2::to_string() + ")";
01544
01545
01546
                  };
01547
01548
               public:
01552
                  template<typename val1, typename val2>
01553
                  struct val {
01555
                      using x = val1;
01557
                      using y = val2;
01559
                      using is_zero_t = typename val1::is_zero_t;
01561
                      static constexpr bool is_zero_v = val1::is_zero_t::value;
01562
01564
                      using ring_type = Ring;
                      using field_type = _FractionField<Ring>;
01565
01566
01569
                       static constexpr bool is_integer = std::is_same_v<val2, typename Ring::one>;
01570
01574
                      template<typename valueType>
01575
                      static constexpr valueType get() { return static_cast<valueType>(x::v) /
     static_cast<valueType>(y::v); }
01576
01579
                      static std::string to_string() {
                          return to_string_helper<val1, val2>::func();
01581
01582
01587
                      template<typename valueRing>
01588
                      static constexpr valueRing eval(const valueRing& v) {
                          return x::eval(v) / y::eval(v);
01589
01590
01591
01592
01594
                  using zero = val<typename Ring::zero, typename Ring::one>;
01596
                  using one = val<typename Ring::one, typename Ring::one>;
01597
01600
                  template<typename v>
01601
                  using inject_t = val<v, typename Ring::one>;
01602
01605
                  template<auto x>
                  using inject_constant_t = val<typename Ring::template inject_constant_t<x>, typename
01606
      Ring::one>;
```

```
01610
                  template<typename v>
01611
                  using inject_ring_t = val<typename Ring::template inject_ring_t<v>, typename Ring::one>;
01612
01614
                  using ring_type = Ring;
01615
01616
               private:
01617
                  template<typename v, typename E = void>
01618
                  struct simplify {};
01619
01620
                  // x = 0
                  template<typename v>
01621
01622
                  struct simplify<v, std::enable_if_t<v::x::is_zero_t::value» {
                      using type = typename _FractionField<Ring>::zero;
01623
01624
01625
                  // x != 0
01626
01627
                  template<typename v>
01628
                  struct simplify<v, std::enable_if_t<!v::x::is_zero_t::value» {</pre>
01629
01630
                      using _gcd = typename Ring::template gcd_t<typename v::x, typename v::y>;
01631
                      using newx = typename Ring::template div_t<typename v::x, _gcd>;
01632
                      using newy = typename Ring::template div_t<typename v::y, _gcd>;
01633
01634
                      using posx = std::conditional_t<
01635
                                           !Ring::template pos_v<newy>,
                                           typename Ring::template sub_t<typename Ring::zero, newx>,
01636
01637
                                           newx>;
01638
                      using posy = std::conditional_t<
01639
                                           !Ring::template pos_v<newy>,
01640
                                           typename Ring::template sub_t<typename Ring::zero, newy>,
01641
01642
                   public:
01643
                      using type = typename _FractionField<Ring>::template val<posx, posy>;
01644
01645
01646
               public:
01649
                  template<typename v>
01650
                  using simplify_t = typename simplify<v>::type;
01651
01652
01653
                  template<typename v1, typename v2>
01654
                  struct add {
01655
                   private:
01656
                      using a = typename Ring::template mul_t<typename v1::x, typename v2::y>;
01657
                      using b = typename Ring::template mul_t<typename v1::y, typename v2::x>;
                      using dividend = typename Ring::template add_t<a, b>;
using diviser = typename Ring::template mul_t<typename v1::y, typename v2::y>;
01658
01659
                      using g = typename Ring::template gcd_t<dividend, diviser>;
01660
01661
01662
                   public:
                      using type = typename _FractionField<Ring>::template simplify_t<val<dividend,
     diviser»;
01664
01665
01666
                  template<typename v>
01667
                  struct pos {
01668
                      using type = std::conditional t<
01669
                           (Ring::template pos_v<typename v::x> && Ring::template pos_v<typename v::y>) ||
01670
                           (!Ring::template pos_v<typename v::x> && !Ring::template pos_v<typename v::y>),
01671
                          std::true type,
01672
                          std::false_type>;
01673
                  };
01674
01675
                  template<typename v1, typename v2>
                  struct sub {
01676
                   private:
01677
01678
                      using a = typename Ring::template mul_t<typename v1::x, typename v2::y>;
                      using b = typename Ring::template mul_t<typename v1::y, typename v2::x>;
01679
                      using dividend = typename Ring::template sub_t<a, b>;
01681
                      using diviser = typename Ring::template mul_t<typename v1::y, typename v2::y>;
01682
                      using g = typename Ring::template gcd_t<dividend, diviser>;
01683
                   public:
01684
                      using type = typename FractionField<Ring>::template simplify t<val<dividend,
01685
01686
01687
01688
                  template<typename v1, typename v2>
01689
                  struct mul {
01690
                   private:
01691
                      using a = typename Ring::template mul_t<typename v1::x, typename v2::x>;
01692
                      using b = typename Ring::template mul_t<typename v1::y, typename v2::y>;
01693
                   public:
01694
01695
                      using type = typename _FractionField<Ring>::template simplify_t<val<a, b>;
01696
```

```
01698
                   template<typename v1, typename v2, typename E = void>
01699
                   struct div {};
01700
01701
                   template<typename v1, typename v2>
struct div<v1, v2, std::enable_if_t<!std::is_same<v2, typename</pre>
01702
      _FractionField<Ring>::zero>::value» {
01703
01704
                      using a = typename Ring::template mul_t<typename v1::x, typename v2::y>;
01705
                       using b = typename Ring::template mul_t<typename v1::y, typename v2::x>;
01706
                    public:
01707
01708
                       using type = typename _FractionField<Ring>::template simplify_t<val<a, b>;
01709
01710
01711
                   template<typename v1, typename v2> \,
01712
                   struct div<v1, v2, std::enable if t<
01713
                       std::is_same<zero, v1>::value && std::is_same<v2, zero>::value» {
                       using type = one;
01715
                   };
01716
01717
                   template<typename v1, typename v2>
01718
                   struct eq {
                       using type = std::conditional_t<
01719
01720
                                std::is_same<typename simplify_t<vl>::x, typename simplify_t<v2>::x>::value &&
01721
                                std::is_same<typename simplify_t<v1>::y, typename simplify_t<v2>::y>::value,
                            std::true_type,
01722
01723
                            std::false_type>;
01724
                   };
01725
01726
                   template<typename TL, typename E = void>
01727
                   struct vadd {};
01728
01729
                   template<typename TL>
01730
                   struct vadd<TL, std::enable_if_t<(TL::length > 1)  {
                       using head = typename TL::pop_front::type;
using tail = typename TL::pop_front::tail;
01731
01732
01733
                        using type = typename add<head, typename vadd<tail>::type>::type;
01734
01735
01736
                   template<typename TL>
                   struct vadd<TL, std::enable_if_t<(TL::length == 1)» {
    using type = typename TL::template at<0>;
01737
01738
01739
01740
01741
                   template<typename... vals>
01742
                   struct vmul {};
01743
01744
                   template<typename v1, typename... vals>
                   struct vmul<v1, vals...> {
01745
                       using type = typename mul<v1, typename vmul<vals...>::type>::type;
01746
01747
01748
01749
                   template<typename v1>
01750
                   struct vmul<v1> {
01751
                       using type = v1;
01752
01753
01754
01755
                   template<typename v1, typename v2, typename E = void>
01756
                   struct qt;
01757
01758
                   template<typename v1, typename v2>
01759
                   struct gt<v1, v2, std::enable_if_t<
01760
                        (eq<v1, v2>::type::value)
01761
01762
                       using type = std::false_type;
01763
                   };
01764
01765
                   template<typename v1, typename v2>
01766
                   struct gt<v1, v2, std::enable_if_t<
01767
                        (!eq<v1, v2>::type::value) &&
01768
                        (!pos<v1>::type::value) && (!pos<v2>::type::value)
01769
01770
                       using type = typename gt<
01771
                           typename sub<zero, v1>::type, typename sub<zero, v2>::type
01772
                        >::type;
01773
01774
01775
                   template<typename v1, typename v2>
01776
                   struct gt<v1, v2, std::enable_if_t<
(!eq<v1, v2>::type::value) &&
01777
01778
                        (pos<v1>::type::value) && (!pos<v2>::type::value)
01779
01780
                       using type = std::true_type;
01781
                   };
01782
```

```
template<typename v1, typename v2>
                   struct gt<v1, v2, std::enable_if_t<
(!eq<v1, v2>::type::value) &&
01784
01785
                       (!pos<v1>::type::value) && (pos<v2>::type::value)
01786
01787
01788
                       using type = std::false type;
01789
                   };
01790
01791
                   template<typename v1, typename v2>
                   struct gt<v1, v2, std::enable_if_t<
(!eq<v1, v2>::type::value) &&
01792
01793
01794
                       (pos<v1>::type::value) && (pos<v2>::type::value)
01795
01796
                       using type = typename Ring::template gt_t<
01797
                            typename Ring::template mul_t<v1::x, v2::y>,
01798
                           typename Ring::template mul_t<v2::y, v2::x>
01799
01800
                   };
01801
01802
                public:
01804
                   template<typename v1, typename v2>
01805
                   using add_t = typename add<v1, v2>::type;
01807
                   template<typename v1, typename v2>
01808
                   using mod t = zero;
01812
                   template<typename v1, typename v2>
01813
                   using gcd_t = v1;
                   template<typename... vs>
01816
01817
                   using vadd_t = typename vadd<vs...>::type;
01820
                   template<typename... vs>
                   using vmul_t = typename vmul<vs...>::type;
01821
01823
                   template<typename v1, typename v2>
01824
                   using sub_t = typename sub<v1, v2>::type;
01826
                   template<typename v1, typename v2>
01827
                   using mul_t = typename mul<v1, v2>::type;
01829
                   template<typename v1, typename v2>
01830
                   using div_t = typename div<v1, v2>::type;
01832
                   template<typename v1, typename v2>
                   using eq_t = typename eq<v1, v2>::type;
01835
                   template<typename v1, typename v2>
01836
                   static constexpr bool eq_v = eq<v1, v2>::type::value;
01838
                   template<typename v1, typename v2>
                   using gt_t = typename gt<v1, v2>::type;
01839
                   template<typename v1, typename v2>
static constexpr bool gt_v = gt<v1, v2>::type::value;
01841
01842
01844
                   template<typename v1>
01845
                   using pos_t = typename pos<v1>::type;
01847
                   template<typename v>
01848
                   static constexpr bool pos_v = pos_t<v>::value;
01849
               };
01850
               template<typename Ring, typename E = void>
01852
               requires IsEuclideanDomain<Ring>
01853
               struct FractionFieldImpl {};
01854
               // fraction field of a field is the field itself
01855
               template<typename Field>
01856
               requires IsEuclideanDomain<Field>
01858
               struct FractionFieldImpl<Field, std::enable_if_t<Field::is_field» {</pre>
01859
                   using type = Field;
01860
                   template < typename v >
01861
                   using inject_t = v;
01862
               };
01863
               // fraction field of a ring is the actual fraction field
01864
01865
               template<typename Ring>
01866
               requires IsEuclideanDomain<Ring>
               struct FractionFieldImpl<Ring, std::enable_if_t<!Ring::is_fieldw {
    using type = _FractionField<Ring>;
01867
01868
01869
               };
          } // namespace internal
01870
01871
01872
          template<typename Ring>
01873
          requires IsEuclideanDomain<Ring>
          using FractionField = typename internal::FractionFieldImpl<Ring>::type;
01874
01875 }
         // namespace aerobus
01876
01877 // short names for common types
01878 namespace aerobus {
01880
          using q32 = FractionField<i32>;
          using fpq32 = FractionField<polynomial<q32»;
01882
          using q64 = FractionField<i64>;
01884
          using pi64 = polynomial<i64>;
01886
          using pq64 = polynomial<q64>;
01888
          using fpq64 = FractionField<polynomial<q64»;
01890
01895
          template<typename Ring, typename v1, typename v2>
          using makefraction_t = typename FractionField<Ring>::template val<v1, v2>;
01896
01897
```

```
template<typename Ring, typename v1, typename v2>
          using addfractions_t = typename FractionField<Ring>::template add_t<v1, v2>;
01903
01908
          template<typename Ring, typename v1, typename v2>
         using mulfractions_t = typename FractionField<Ring>::template mul_t<v1, v2>;
01909
01910 }
        // namespace aerobus
01911
01912 // taylor series and common integers (factorial, bernouilli...) appearing in taylor coefficients
01913 namespace aerobus {
01914
       namespace internal {
01915
             template<typename T, size_t x, typename E = void>
              struct factorial {};
01916
01917
01918
              template<typename T, size_t x>
01919
              struct factorial<T, x, std::enable_if_t<(x > 0)» {
01920
              private:
01921
                  template<typename, size_t, typename>
01922
                  friend struct factorial:
              public:
01923
01924
                 using type = typename T::template mul_t<typename T::template val<x>, typename factorial<T,
     x - 1>::type>;
01925
                  static constexpr typename T::inner_type value = type::template get<typename
     T::inner_type>();
01926
            };
01927
01928
             template<typename T>
01929
             struct factorial<T, 0> {
01930
              public:
01931
                  using type = typename T::one;
01932
                  static constexpr typename T::inner_type value = type::template get<typename
     T::inner_type>();
01933
              };
01934
          } // namespace internal
01935
01939
          template<typename T, size_t i>
01940
          using factorial_t = typename internal::factorial<T, i>::type;
01941
01945
          template<typename T, size_t i>
          inline constexpr typename T::inner_type factorial_v = internal::factorial<T, i>::value;
01946
01947
01948
          namespace internal {
01949
              template<typename T, size_t k, size_t n, typename E = void>
01950
              struct combination_helper {};
01951
01952
              template<typename T, size_t k, size_t n>
01953
              struct combination_helper<T, k, n, std::enable_if_t<(n >= 0 && k <= (n / 2) && k > 0)» {
01954
                  using type = typename FractionField<T>::template mul_t<</pre>
01955
                      typename combination_helper<T, k - 1, n - 1>::type,
01956
                      makefraction_t<T, typename T::template val<n>, typename T::template val<k>>;
01957
              };
01958
01959
              template<typename T, size_t k, size_t n>
01960
              struct combination_helper<T, k, n, std::enable_if_t<(n >= 0 && k > (n / 2) && k > 0)» {
01961
                  using type = typename combination_helper<T, n - k, n>::type;
01962
01963
01964
              template<typename T, size t n>
              struct combination_helper<T, 0, n> {
01965
                  using type = typename FractionField<T>::one;
01966
01967
01968
              template<typename T, size_t k, size_t n>
01969
01970
              struct combination {
01971
                  using type = typename internal::combination_helper<T, k, n>::type::x;
01972
                  static constexpr typename T::inner_type value
01973
                              internal::combination_helper<T, k, n>::type::template get<typename</pre>
     T::inner_type>();
01974
            };
// namespace internal
01975
01976
01979
          template<typename T, size_t k, size_t n>
01980
          using combination_t = typename internal::combination<T, k, n>::type;
01981
          template<typename T, size_t k, size_t n>
inline constexpr typename T::inner_type combination_v = internal::combination<T, k, n>::value;
01986
01987
01988
01989
          namespace internal {
01990
              template<typename T, size_t m>
01991
              struct bernouilli;
01992
01993
              template<typename T, typename accum, size t k, size t m>
01994
              struct bernouilli helper {
                  using type = typename bernouilli_helper<
01995
01996
01997
                      addfractions_t<T,
                          accum,
01998
                          mulfractions t<T,
01999
02000
                              makefraction t<T.
```

```
02001
                                   combination_t<T, k, m + 1>,
02002
                                   typename T::one>,
02003
                               typename bernouilli<T, k>::type
02004
02005
                      >,
k + 1,
02006
02007
                      m>::type;
02008
02009
              template<typename T, typename accum, size_t m>
struct bernouilli_helper<T, accum, m, m> {
02010
02011
02012
                  using type = accum;
02013
02014
02015
02016
02017
              template<typename T, size_t m>
02018
              struct bernouilli {
02019
                  using type = typename FractionField<T>::template mul_t<</pre>
02020
                      typename internal::bernouilli_helper<T, typename FractionField<T>::zero, 0, m>::type,
02021
                      makefraction t<T,
02022
                      typename T::template val<static_cast<typename T::inner_type>(-1)>,
02023
                      typename T::template val<static_cast<typename T::inner_type>(m + 1)>
02024
02025
                  >;
02026
                  template<typename floatType>
02027
02028
                  static constexpr floatType value = type::template get<floatType>();
02029
              };
02030
02031
              template<tvpename T>
02032
              struct bernouilli<T, 0> {
02033
                  using type = typename FractionField<T>::one;
02034
02035
                  template<typename floatType>
                  static constexpr floatType value = type::template get<floatType>();
02036
02037
              };
02038
          } // namespace internal
02039
02043
          template<typename T, size_t n>
02044
          using bernouilli_t = typename internal::bernouilli<T, n>::type;
02045
02050
          template<typename FloatType, typename T, size t n >
02051
          inline constexpr FloatType bernouilli_v = internal::bernouilli<T, n>::template value<FloatType>;
02052
02053
          namespace internal {
02054
             template<typename T, int k, typename E = void>
02055
              struct alternate { };
02056
02057
              template<tvpename T, int k>
02058
              struct alternate<T, k, std::enable_if_t<k % 2 == 0» {
02059
                  using type = typename T::one;
02060
                  static constexpr typename T::inner_type value = type::template get<typename
     T::inner_type>();
02061
              };
02062
02063
              template<typename T, int k>
02064
              struct alternate<T, k, std::enable_if_t<k % 2 != 0» {
02065
                 using type = typename T::template sub_t<typename T::zero, typename T::one>;
02066
                  static constexpr typename T::inner_type value = type::template get<typename
     T::inner_type>();
02067
              };
02068
          } // namespace internal
02069
02072
          template<typename T, int k>
02073
          using alternate_t = typename internal::alternate<T, k>::type;
02074
02077
          template<typename T, size_t k>
02078
          inline constexpr typename T::inner_type alternate_v = internal::alternate<T, k>::value;
02079
02081
          namespace internal {
02082
              template<typename T, auto p, auto n>
02083
              struct pow {
                  using type = typename T::template mul_t<typename T::template val<p>, typename pow<T, p, n
02084
      - 1>::type>;
02085
              };
02086
02087
              template<typename T, auto p>
02088
              struct pow<T, p, 0> { using type = typename T::one; };
02089
02090
02095
          template<typename T, auto p, auto n>
02096
          using pow_t = typename internal::pow<T, p, n>::type;
02097
02098
          namespace internal {
              template<typename, template<typename, size_t> typename, class>
02099
02100
              struct make_taylor_impl;
```

```
template<typename T, template<typename, size_t> typename coeff_at, size_t... Is>
02102
02103
               struct make_taylor_impl<T, coeff_at, std::integer_sequence<size_t, Is...» {</pre>
                using type = typename polynomial<FractionField<To::template val<typename coeff_at<T,
02104
      Is>::type...>;
02105
             };
02106
02107
02112
           template<typename T, template<typename, size_t index> typename coeff_at, size_t deg>
02113
          using taylor = typename internal::make_taylor_impl<</pre>
02114
02115
               coeff at.
02116
               internal::make_index_sequence_reverse<deg + 1>::type;
02117
02118
          namespace internal {
02119
               template<typename T, size_t i>
02120
               struct exp_coeff {
                  using type = makefraction_t<T, typename T::one, factorial_t<T, i»;</pre>
02121
02123
02124
               template<typename T, size_t i, typename E = void>
02125
               struct sin_coeff_helper {};
02126
               template<typename T, size_t i>
struct sin_coeff_helper<T, i, std::enable_if_t<(i & 1) == 0» {</pre>
02127
02128
                 using type = typename FractionField<T>::zero;
02129
02130
02131
02132
               template<typename T, size_t i>
               struct sin_coeff_helper<T, i, std::enable_if_t<(i & 1) == 1» {</pre>
02133
02134
                   using type = makefraction t<T, alternate t<T, i / 2>, factorial t<T, i»:
02135
02136
02137
               template<typename T, size_t i>
02138
               struct sin_coeff {
                   using type = typename sin_coeff_helper<T, i>::type;
02139
02140
               };
02142
               template<typename T, size_t i, typename E = void>
02143
               struct sh_coeff_helper {};
02144
02145
               template<typename T, size_t i>
               struct sh_coeff_helper<T, i, std::enable_if_t<(i & 1) == 0» {
    using type = typename FractionField<T>::zero;
02146
02147
02148
02149
02150
               template<typename T, size_t i>
               struct sh_coeff_helper<T, i, std::enable_if_t<(i & 1) == 1» {
02151
                   using type = makefraction_t<T, typename T::one, factorial_t<T, i»;</pre>
02152
02153
02154
02155
               template<typename T, size_t i>
02156
               struct sh_coeff {
02157
                  using type = typename sh_coeff_helper<T, i>::type;
02158
02159
02160
               template<typename T, size_t i, typename E = void>
02161
               struct cos_coeff_helper {};
02162
02163
               template<typename T, size_t i>
               struct cos_coeff_helper<T, i, std::enable_if_t<(i & 1) == 1» {
    using type = typename FractionField<T>::zero;
02164
02165
02166
02167
               template<typename T, size_t i>
02168
02169
               struct cos_coeff_helper<T, i, std::enable_if_t<(i & 1) == 0\times {
                   using type = makefraction_t<T, alternate_t<T, i / 2>, factorial_t<T, i»;</pre>
02170
02171
02172
02173
               template<typename T, size_t i>
02174
               struct cos_coeff {
02175
                   using type = typename cos_coeff_helper<T, i>::type;
02176
02177
02178
               template<typename T, size_t i, typename E = void>
               struct cosh_coeff_helper {};
02179
02180
               template<typename T, size_t i>
02181
               struct cosh_coeff_helper<T, i, std::enable_if_t<(i & 1) == 1» {
    using type = typename FractionField<T>::zero;
02182
02183
02184
02185
02186
               template<typename T, size_t i>
               struct cosh_coeff_helper<T, i, std::enable_if_t<(i & 1) == 0» {
02187
02188
                   using type = makefraction_t<T, typename T::one, factorial_t<T, i»;</pre>
02189
02190
```

```
template<typename T, size_t i>
02192
              struct cosh coeff {
02193
                  using type = typename cosh_coeff_helper<T, i>::type;
02194
02195
02196
              template<tvpename T, size t i>
              struct geom_coeff { using type = typename FractionField<T>::one; };
02197
02198
02199
02200
              template<typename T, size_t i, typename E = void>
02201
              struct atan_coeff_helper;
02202
02203
              template<typename T, size t i>
02204
              struct atan_coeff_helper<T, i, std::enable_if_t<(i & 1) == 1» {
02205
                  using type = makefraction_t<T, alternate_t<T, i / 2>, typename T::template val<i>»;
02206
02207
02208
              template<typename T, size t i>
              struct atan_coeff_helper<T, i, std::enable_if_t<(i & 1) == 0» {</pre>
02210
                 using type = typename FractionField<T>::zero;
02211
02212
02213
              template<typename T, size_t i>
              struct atan_coeff { using type = typename atan_coeff_helper<T, i>::type; };
02214
02215
02216
              template<typename T, size_t i, typename E = void>
02217
              struct asin_coeff_helper;
02218
02219
              template<typename T, size_t i>
              struct asin_coeff_helper<T, i, std::enable_if_t<(i & 1) == 1» {</pre>
02220
02221
                  using type = makefraction t<T.
02222
                       factorial_t<T, i - 1>,
02223
                       typename T::template mul_t<
02224
                           typename T::template val<i>,
                           T::template mul_t<
    pow_t<T, 4, i / 2>,
    pow<T, factorial<T, i / 2>::value, 2
02225
02226
02227
02229
02230
02231
              };
02232
              template<typename T, size_t i>
02233
02234
              struct asin_coeff_helper<T, i, std::enable_if_t<(i & 1) == 0» {</pre>
02235
                  using type = typename FractionField<T>::zero;
02236
02237
02238
              template<typename T, size_t i>
02239
              struct asin coeff {
02240
                 using type = typename asin_coeff_helper<T, i>::type;
02241
02242
02243
              template<typename T, size_t i>
02244
              struct lnp1_coeff {
                  using type = makefraction_t<T,
02245
02246
                      alternate_t<T, i + 1>,
02247
                       typename T::template val<i>;;
02248
              } ;
02249
02250
              template<typename T>
02251
              struct lnpl_coeff<T, 0> { using type = typename FractionField<T>::zero; };
02252
02253
              template<typename T, size_t i, typename E = void>
02254
              struct asinh_coeff_helper;
02255
02256
              template<typename T, size_t i>
              struct asinh_coeff_helper<T, i, std::enable_if_t<(i & 1) == 1» {</pre>
02257
02258
                  using type = makefraction_t<T,
                       typename T::template mul_t<
02259
02260
                          alternate_t<T, i / 2>,
02261
                           factorial_t<T, i - 1>
02262
02263
                       typename T::template mul_t<
02264
                           T::template mul_t<
02265
                               typename T::template val<i>,
02266
                               pow_t<T, (factorial<T, i / 2>::value), 2>
02267
02268
                           pow_t<T, 4, i / 2>
02269
02270
                  >:
02271
              };
02272
02273
              template<typename T, size_t i>
02274
              struct asinh_coeff_helper<T, i, std::enable_if_t<(i & 1) == 0» {
02275
                  using type = typename FractionField<T>::zero;
02276
02277
```

```
template<typename T, size_t i>
              struct asinh_coeff {
02279
02280
                  using type = typename asinh_coeff_helper<T, i>::type;
02281
02282
02283
              template<typename T, size_t i, typename E = void>
              struct atanh_coeff_helper;
02284
02285
02286
              template<typename T, size_t i>
02287
              struct atanh_coeff_helper<T, i, std::enable_if_t<(i & 1) == 1» {</pre>
02288
                  // 1/i
                  using type = typename FractionField<T>:: template val<</pre>
02289
02290
                       typename T::one,
02291
                       typename T::template val<static_cast<typename T::inner_type>(i)»;
02292
              };
02293
02294
              template<typename T, size_t i>
              struct atanh_coeff_helper<T, i, std::enable_if_t<(i & 1) == 0» {</pre>
02295
                  using type = typename FractionField<T>::zero;
02297
              };
02298
02299
              template<typename T, size_t i>
02300
              struct atanh_coeff {
                  using type = typename asinh_coeff_helper<T, i>::type;
02301
02302
02303
02304
              template<typename T, size_t i, typename E = void>
02305
              struct tan_coeff_helper;
02306
02307
              template<typename T, size_t i>
02308
              struct tan_coeff_helper<T, i, std::enable_if_t<(i % 2) == 0» {</pre>
02309
                  using type = typename FractionField<T>::zero;
02310
02311
02312
              template<typename T, size_t i>
              struct tan_coeff_helper<T, i, std::enable_if_t<(i % 2) != 0» {</pre>
02313
02314
              private:
02315
                  using _4p = typename FractionField<T>::template inject_t<pow_t<T, 4, (i + 1) / 2»; // 4^{((i+1)/2)} - 1
02316
02317
02318
                  using _4pm1 = typename FractionField<T>::template sub_t<_4p, typename</pre>
     FractionField<T>::one>;
02319
                  // (-1) ^ ((i-1) /2)
02320
                  using altp = typename FractionField<T>::template inject_t<alternate_t<T, (i - 1) / 2»;
                  using dividend = typename FractionField<T>::template mul_t<
02321
02322
                       altp,
02323
                      FractionField<T>::template mul_t<
02324
                       _4p,
                      FractionField<T>::template mul_t<
02325
02326
                       4pm1.
                       bernouilli_t<T, (i + 1)>
02327
02328
02329
02330
02331
              public:
                  using type = typename FractionField<T>::template div_t<dividend,</pre>
02332
02333
                      typename FractionField<T>::template inject_t<factorial_t<T, i + 1>>;
02334
              };
02335
02336
              template<typename T, size_t i>
02337
              struct tan_coeff {
02338
                  using type = typename tan_coeff_helper<T, i>::type;
02339
02340
02341
              template<typename T, size_t i, typename E = void>
02342
              struct tanh_coeff_helper;
02343
              template<typename T, size_t i>
02344
              struct tanh_coeff_helper<T, i, std::enable_if_t<(i % 2) == 0» {
02345
                  using type = typename FractionField<T>::zero;
02346
02347
02348
02349
              template<typename T, size_t i>
              struct tanh_coeff_helper<T, i, std::enable_if_t<(i % 2) != 0» {</pre>
02350
02351
              private:
02352
                  using _4p = typename FractionField<T>::template inject_t<pow_t<T, 4, (i + 1) / 2»;
                  using _4pm1 = typename FractionField<T>::template sub_t<_4p, typename
02353
     FractionField<T>::one>;
02354
                  using dividend =
02355
                      typename FractionField<T>::template mul t<
02356
                       4p,
02357
                      typename FractionField<T>::template mul_t<</pre>
02358
02359
                      bernouilli_t<T, (i + 1)>
02360
02361
                      >::type;
02362
              public:
```

```
02363
                              using type = typename FractionField<T>::template div_t<dividend,
02364
                                     FractionField<T>::template inject_t<factorial_t<T, i + 1>>;
02365
                       };
02366
02367
                        template<typename T, size_t i>
02368
                        struct tanh coeff {
02369
                              using type = typename tanh_coeff_helper<T, i>::type;
02370
02371
                 } // namespace internal
02372
                template<typename T, size_t deg>
using exp = taylor<T, internal::exp_coeff, deg>;
02376
02377
02378
02382
                 template<typename T, size_t deg>
02383
                 using expm1 = typename polynomial<FractionField<T>::template sub_t<</pre>
02384
                        exp<T, deg>
02385
                        typename polynomial<FractionField<T>::one>;
02386
02390
                 template<typename T, size_t deg>
02391
                 using lnp1 = taylor<T, internal::lnp1_coeff, deg>;
02392
02396
                 template<typename T, size_t deg>
02397
                 using atan = taylor<T, internal::atan_coeff, deg>;
02398
02402
                template<typename T, size_t deg>
using sin = taylor<T, internal::sin_coeff, deg>;
02403
02404
02408
                 template<typename T, size_t deg>
02409
                 using sinh = taylor<T, internal::sh_coeff, deg>;
02410
02414
                 template<typename T, size_t deg>
02415
                 using cosh = taylor<T, internal::cosh_coeff, deg>;
02416
02420
                 template<typename T, size_t deg>
02421
                 using cos = taylor<T, internal::cos_coeff, deg>;
02422
                template<typename T, size_t deg>
using geometric_sum = taylor<T, internal::geom_coeff, deg>;
02426
02427
02428
02432
                 template<typename T, size_t deg>
02433
                 using asin = taylor<T, internal::asin_coeff, deg>;
02434
                template<typename T, size_t deg>
using asinh = taylor<T, internal::asinh_coeff, deg>;
02438
02439
02440
                 template<typename T, size_t deg>
02444
02445
                 using atanh = taylor<T, internal::atanh_coeff, deg>;
02446
02450
                template<typename T, size_t deg>
using tan = taylor<T, internal::tan_coeff, deg>;
02451
02452
02456
                 template<typename T, size_t deg>
02457
                 using tanh = taylor<T, internal::tanh_coeff, deg>;
02458 } // namespace aerobus
02459
02460 // continued fractions
02461 namespace aerobus {
02464
                 template<int64 t... values>
02465
                 struct ContinuedFraction {};
02466
02469
                 template<int.64 t a0>
02470
                 struct ContinuedFraction<a0> {
02471
                       using type = typename q64::template inject_constant_t<a0>;
02472
                        static constexpr double val = type::template get<double>();
02473
02474
02478
                 template<int64_t a0, int64_t... rest>
                 struct ContinuedFraction<a0, rest...> {
02479
02480
                       using type = q64::template add_t<
02481
                                      typename q64::template inject_constant_t<a0>,
02482
                                      typename q64::template div_t<
02483
                                             typename q64::one,
02484
                                            typename ContinuedFraction<rest...>::type
02485
                                      »;
02486
                        static constexpr double val = type::template get<double>();
02487
                 };
02488
02493
                using PI_fraction =
          ContinuedFraction<3, 7, 15, 1, 292, 1, 1, 1, 2, 1, 3, 1, 14, 2, 1, 1, 2, 2, 2, 2, 1>;
02496
                using E fraction =
          ContinuedFraction<2, 1, 2, 1, 1, 4, 1, 1, 6, 1, 1, 8, 1, 1, 10, 1, 1, 12, 1, 1, 14, 1, 1>;
                using SQRT2_fraction =
          02500
                using SQRT3_fraction =
          ContinuedFraction<1, 1, 2, 1, 2, 1, 2, 1, 2, 1, 2, 1, 2, 1, 2, 1, 2, 1, 2, 1, 2, 1, 2, 1, 2, 1, 2, 1, 2, 1, 2, 1, 2, 1, 2, 1, 2, 1, 2, 1, 2, 1, 2, 1, 2, 1, 2, 1, 2, 1, 2, 1, 2, 1, 2, 1, 2, 1, 2, 1, 2, 1, 2, 1, 2, 1, 2, 1, 2, 1, 2, 1, 2, 1, 2, 1, 2, 1, 2, 1, 2, 1, 2, 1, 2, 1, 2, 1, 2, 1, 2, 1, 2, 1, 2, 1, 2, 1, 2, 1, 2, 1, 2, 1, 2, 1, 2, 1, 2, 1, 2, 1, 2, 1, 2, 1, 2, 1, 2, 1, 2, 1, 2, 1, 2, 1, 2, 1, 2, 1, 2, 1, 2, 1, 2, 1, 2, 1, 2, 1, 2, 1, 2, 1, 2, 1, 2, 1, 2, 1, 2, 1, 2, 1, 2, 1, 2, 1, 2, 1, 2, 1, 2, 1, 2, 1, 2, 1, 2, 1, 2, 1, 2, 1, 2, 1, 2, 1, 2, 1, 2, 1, 2, 1, 2, 1, 2, 1, 2, 1, 2, 1, 2, 1, 2, 1, 2, 1, 2, 1, 2, 1, 2, 1, 2, 1, 2, 1, 2, 1, 2, 1, 2, 1, 2, 1, 2, 1, 2, 1, 2, 1, 2, 1, 2, 1, 2, 1, 2, 1, 2, 1, 2, 1, 2, 1, 2, 1, 2, 1, 2, 1, 2, 1, 2, 1, 2, 1, 2, 1, 2, 1, 2, 1, 2, 1, 2, 1, 2, 1, 2, 1, 2, 1, 2, 1, 2, 1, 2, 1, 2, 1, 2, 1, 2, 1, 2, 1, 2, 1, 2, 1, 2, 1, 2, 1, 2, 1, 2, 1, 2, 1, 2, 1, 2, 1, 2, 1, 2, 1, 2, 1, 2, 1, 2, 1, 2, 1, 2, 1, 2, 1, 2, 1, 2, 1, 2, 1, 2, 1, 2, 1, 2, 1, 2, 1, 2, 1, 2, 1, 2, 1, 2, 1, 2, 1, 2, 1, 2, 1, 2, 1, 2, 1, 2, 1, 2, 1, 2, 1, 2, 1, 2, 1, 2, 1, 2, 1, 2, 1, 2, 1, 2, 1, 2, 1, 2, 1, 2, 1, 2, 1, 2, 1, 2, 1, 2, 1, 2, 1, 2, 1, 2, 1, 2, 1, 2, 1, 2, 1, 2, 1, 2, 1, 2, 1, 2, 1, 2, 1, 2, 1, 2, 1, 2, 1, 2, 1, 2, 1, 2, 1, 2, 1, 2, 1, 2, 1, 2, 1, 2, 1, 2, 1, 2, 1, 2, 1, 2, 1, 2, 1, 2, 1, 2, 1, 2, 1, 2, 1, 2, 1, 2, 1, 2, 1, 2, 1, 2, 1, 2, 1, 2, 1, 2, 1, 2, 1, 2, 1, 2, 1, 2, 1, 2, 1, 2, 1, 2, 1, 2, 1, 2, 1, 2, 1, 2, 1, 2, 1, 2, 1, 2, 1, 2, 1, 2, 1, 2, 1, 2, 1, 2, 1, 2, 1, 2, 1, 2, 1, 2, 1, 2, 1, 2, 1, 2, 1, 2, 1, 2, 1, 2, 1, 2, 1, 2, 1, 2, 1, 2, 1, 2, 1, 2, 1, 2, 1, 2, 1, 2, 1, 2, 1, 2, 1, 2, 1, 2, 1, 2, 1, 2, 1, 2, 1, 2, 1, 2, 1, 2, 1, 2, 1, 2, 1, 2, 1, 2, 1, 2, 1, 2, 1, 2, 1, 2, 1, 2, 1, 2, 1, 2, 1, 2, 1, 2, 1, 2, 1, 2, 1, 2, 1, 2, 1, 2, 1, 2, 1, 2, 1, 2, 1, 2, 1, 2, 1, 2, 1, 2, 1, 2, 1, 2, 1, 2, 1, 2, 1, 2, 1, 2, 1, 2, 1, 2, 1, 2, 1, 2, 1, 2, 1, 2, 1, 2, 1, 2, 1, 2, 1, 2, 1, 2, 1, 2, 1, 2, 1, 2, 1, 2, 1, 2, 1, 2, 1, 2, 1, 2, 1, 2, 1, 2, 1, 2, 1, 2, 1, 2, 1, 2, 1, 2, 1, 2, 1, 2, 1, 2, 1, 2, 1, 2, 1, 
          // NOLINT
02501 } // namespace aerobus
```

```
02502
02503 // known polynomials
02504 namespace aerobus {
02505 // CChebyshev
02506
          namespace internal {
               template<int kind, int deg>
02507
02508
               struct chebyshev_helper {
02509
                   using type = typename pi64::template sub_t<
02510
                       typename pi64::template mul_t<
02511
                        typename pi64::template mul_t<
02512
                        pi64::inject_constant_t<2>,
02513
                       typename pi64::X
02514
02515
                        typename chebyshev_helper<kind, deg - 1>::type
02516
02517
                        typename chebyshev_helper<kind, deg - 2>::type
02518
02519
              };
02520
02521
               template<>
02522
               struct chebyshev_helper<1, 0> {
02523
                   using type = typename pi64::one;
02524
               };
02525
02526
               template<>
               struct chebyshev_helper<1, 1> {
02527
02528
                   using type = typename pi64::X;
02529
02530
02531
               template<>
02532
               struct chebyshev_helper<2, 0> {
02533
                   using type = typename pi64::one;
02534
02535
02536
               template<>
               struct chebyshev_helper<2, 1> {
02537
                   using type = typename pi64::template mul_t
typename pi64::inject_constant_t<2>,
02538
02540
                       typename pi64::X>;
02541
02542
           } // namespace internal
02543
          // Laguerre
02544
02545
          namespace internal {
02546
               template<size_t deg>
02547
               struct laguerre_helper {
02548
               private:
                   // Lk = (1 / k) * ((2 * k - 1 - x) * 1km1 - (k - 2) Lkm2)
02549
                   using lnm2 = typename laguerre_helper<deg - 2>::type;
02550
02551
                   using lnm1 = typename laguerre_helper<deg - 1>::type;
02552
                   // -x + 2k-1
02553
                   using p = typename pq64::template val<
02554
                       typename q64::template inject_constant_t<-1>,
02555
                        typename q64::template inject_constant_t<2 * deg - 1»;
02556
                   // 1/n
                   using factor = typename pq64::template inject_ring_t<
q64::val<typename i64::one, typename i64::template inject_constant_t<deg>>;
02557
02558
02559
02560
                public:
02561
                   using type = typename pq64::template mul_t <</pre>
02562
                        factor.
                        typename pq64::template sub_t<
02563
                            typename pq64::template mul_t<
02564
02565
                                p,
02566
                                1 nm1
02567
02568
                            typename pq64::template mul_t<
                                typename pq64::template inject_constant_t<deg-1>,
02569
02570
                                lnm2
02571
02572
02573
                   >;
02574
02575
               };
02576
02577
02578
               struct laguerre_helper<0> {
02579
                   using type = typename pq64::one;
02580
               };
02581
02582
               template<>
02583
               struct laguerre_helper<1> {
02584
                   using type = typename pq64::template sub_t<typename pq64::one, typename pq64::X>;
02585
02586
           } // namespace internal
02587
02588
          namespace known polynomials {
```

```
enum hermite_kind {
02591
                  probabilist,
02592
                  physicist
02593
              };
02594
         }
02595
02596
         namespace internal {
02597
              template<size_t deg, known_polynomials::hermite_kind kind>
02598
              struct hermite_helper {};
02599
02600
              template<size_t deg>
02601
              struct hermite_helper<deg, known_polynomials::hermite_kind::probabilist> {
02602
              private:
                  using hnm1 = typename hermite_helper<deg - 1,
02603
      known_polynomials::hermite_kind::probabilist>::type;
02604
                  using hnm2 = typename hermite_helper<deg - 2,
      known_polynomials::hermite_kind::probabilist>::type;
02605
               public:
02606
02607
                  using type = typename pi64::template sub_t<
02608
                      typename pi64::template mul_t<typename pi64::X, hnm1>,
02609
                      typename pi64::template mul_t<
02610
                          typename pi64::template inject_constant_t<deg - 1>,
02611
                          hnm2
02612
02613
                  >;
02614
              } ;
02615
02616
              template<size_t deg>
02617
              struct hermite_helper<deq, known_polynomials::hermite_kind::physicist> {
02618
              private:
02619
                  using hnm1 = typename hermite_helper<deg - 1,
      known_polynomials::hermite_kind::physicist>::type;
02620
                  using hnm2 = typename hermite_helper<deg - 2,</pre>
      known_polynomials::hermite_kind::physicist>::type;
02621
02622
               public:
02623
                  using type = typename pi64::template sub_t<
02624
                       // 2X Hn-1
02625
                      typename pi64::template mul_t<
02626
                          typename pi64::val<typename i64::template inject_constant_t<2>,
                          typename i64::zero>, hnm1>,
02627
02628
02629
                      typename pi64::template mul_t<
02630
                          typename pi64::template inject_constant_t<2*(deg - 1)>,
02631
                          hnm2
02632
02633
                  >;
02634
             };
02635
02636
              template<>
02637
              struct hermite_helper<0, known_polynomials::hermite_kind::probabilist> {
02638
                  using type = typename pi64::one;
02639
02640
02641
              template<>
02642
              struct hermite_helper<1, known_polynomials::hermite_kind::probabilist> {
02643
                  using type = typename pi64::X;
02644
02645
02646
              template<>
02647
              struct hermite_helper<0, known_polynomials::hermite_kind::physicist> {
02648
                 using type = typename pi64::one;
02649
02650
02651
              template<>
02652
              struct hermite_helper<1, known_polynomials::hermite_kind::physicist> {
02653
                  // 2X
02654
                  using type = typename pi64::template val<typename i64::template inject constant t<2>.
     typename i64::zero>;
02655
              } ;
02656
          } // namespace internal
02657
02658
          namespace known_polynomials {
02661
              template <size_t deg>
02662
              using chebyshev_T = typename internal::chebyshev_helper<1, deg>::type;
02663
02666
              template <size_t deg>
02667
              using chebyshev_U = typename internal::chebyshev_helper<2, deg>::type;
02668
02671
              template <size t deg>
02672
              using laguerre = typename internal::laguerre_helper<deg>::type;
02673
02676
              template <size_t deg>
02677
              using hermite_prob = typename internal::hermite_helper<deg, hermite_kind::probabilist>::type;
02678
02681
              template <size t deg>
```

```
using hermite_phys = typename internal::hermite_helper<deg, hermite_kind::physicist>::type;
                                               // namespace known_polynomials
 02684 } // namespace aerobus
 02685
 02686
 02687 #ifdef AEROBUS_CONWAY_IMPORTS
 02688 template<int p, int n>
 02689 struct ConwayPolynomial;
 02690
 02691 #define ZPZV ZPZ::template val
 02692 #define POLYV aerobus::polynomial<ZPZ>::template val
02693 template<> struct ConwayPolynomial<2, 1> { using ZPZ = aerobus::zpz<2>; using type = POLYV<ZPZV<1>,
                       ZPZV<1»; }; // NOLINT</pre>
 02694 template<> struct ConwayPolynomial<2, 2> { using ZPZ = aerobus::zpz<2>; using type = POLYV<ZPZV<1>,
                       ZPZV<1>, ZPZV<1»; }; // NOLINT</pre>
02695 template<> struct ConwayPolynomial<2, 3> { using ZPZ = aerobus::zpz<2>; using type = POLYV<ZPZV<1>,
                      ZPZV<0>, ZPZV<1>, ZPZV<1»; }; // NOLINT</pre>
02696 template<> struct ConwayPolynomial<2, 4> { using ZPZ = aerobus::zpz<2>; using type = POLYV<ZPZV<1>,
                       ZPZV<0>, ZPZV<0>, ZPZV<1>, ZPZV<1»; }; // NOLINT</pre>
02697 template<> struct ConwayPolynomial<2, 5> { using ZPZ = aerobus::zpz<2>; using type = POLYV<ZPZV<1>,
                       ZPZV<0>, ZPZV<0>, ZPZV<1>, ZPZV<0>, ZPZV<1»; }; // NOLINT</pre>
 02698 template<> struct ConwayPolynomial<2, 6> { using ZPZ = aerobus::zpz<2>; using type = POLYV<ZPZV<1>,
                       ZPZV<0>, ZPZV<1>, ZPZV<1>, ZPZV<0>, ZPZV<1>, ZPZV<1»; }; // NOLINT</pre>
02699 template<> struct ConwayPolynomial<2, 7> { using ZPZ = aerobus::zpz<2>; using type = POLYV<ZPZV<1>, ZPZV<0>, ZPZV<0>, ZPZV<0>, ZPZV<0>, ZPZV<1>, ZPZV<1>, ZPZV<1»; }; // NOLINT
 02700 template<> struct ConwayPolynomial<2, 8> { using ZPZ = aerobus::zpz<2>; using type = POLYV<ZPZV<1>,
                       ZPZV<0>, ZPZV<0>, ZPZV<0>, ZPZV<1>, ZPZV<1>, ZPZV<1>, ZPZV<0>, ZPZV<1»; };</pre>
02701 template<> struct ConwayPolynomial<2, 9> { using ZPZ = aerobus::zpz<2>; using type = POLYV<ZPZV<1>,
ZPZV<0>, ZPZV<0>, ZPZV<0>, ZPZV<0>, ZPZV<1>, ZPZV<0>, ZPZV<0>, ZPZV<0>, ZPZV<0>, ZPZV<0>, ZPZV<1»; }; // NOLINT
02702 template<> struct ConwayPolynomial<2, 10> { using ZPZ = aerobus::zpz<2>; using type = POLYV<ZPZV<1>,
                       ZPZV<0>, ZPZV<0>, ZPZV<0>, ZPZV<1>, ZPZV<1>, ZPZV<1>, ZPZV<1>, ZPZV<1>, ZPZV<1>, ZPZV<1>, ZPZV<1</pre>
                       NOLINT
02703 template<> struct ConwayPolynomial<2, 11> { using ZPZ = aerobus::zpz<2>; using type = POLYV<ZPZV<1>,
                       ZPZV<0>, ZPZV<0>, ZPZV<0>, ZPZV<0>, ZPZV<0>, ZPZV<0>, ZPZV<0>, ZPZV<0>, ZPZV<0>, ZPZV<1>, ZPZV<1
, ZPZV<1
                        // NOLINT
02704 template<> struct ConwayPolynomial<2, 12> { using ZPZ = aerobus::zpz<2>; using type = POLYV<ZPZV<1>,
                       ZPZV<0>, ZPZV<0>, ZPZV<0>, ZPZV<0>, ZPZV<1>, ZPZV<1>, ZPZV<1>, ZPZV<0>, ZPZV<0>, ZPZV<0>, ZPZV<0>,
                       ZPZV<1»; }; // NOLINT</pre>
 02705 template<> struct ConwayPolynomial<2, 13> { using ZPZ = aerobus::zpz<2>; using type = POLYV<ZPZV<1>,
                       ZPZV<0>, ZPZV<0>, ZPZV<0>, ZPZV<0>, ZPZV<0>, ZPZV<0>, ZPZV<0>, ZPZV<0>, ZPZV<0>, ZPZV<1>, ZPZV<1
, ZPZV<1 , ZPZV<1 , ZPZV<1 , ZPZV<1 , ZPZV<1 , ZPZV<1 , ZPZV<1 , ZPZV<1 , ZPZV<1 , ZPZV<1 , ZPZV<1 , ZPZV<1 , ZPZV<1 , ZPZV<1 , ZPZV<1 , ZPZV<1 , ZPZV<1 , ZPZV<1 , ZPZV<1 , ZPZV<1 , ZPZV<1 , ZPZV<1 , ZPZV<1 , ZPZV<1 , ZPZV<1 , ZPZV<1 , ZPZV<1 , ZPZV<1 , ZPZV<1 , ZPZV<1 , ZPZV<1 , ZPZV<1 , ZPZV<1 , ZPZV<1 , ZPZV<1 , ZPZV<1 , ZPZV<1 , ZPZV<1 , ZPZV<1 , ZPZV<1 , ZPZV<1 , ZPZV<1 , ZPZV<1 , ZPZV<1 , ZPZV<1 , ZPZV<1 , ZPZV<1 , ZPZV<1 , ZPZV<1 , ZPZV<1 , ZPZV<1 , ZPZV<1 , ZPZV<1 , ZPZV<1 , ZPZV<1 , ZPZV<1 , ZPZV<1 , ZPZV<1 , ZPZV<1 , ZPZV<1 , ZPZV<1 , ZPZV<1 , ZPZV<1 , ZPZV<1 , ZPZV<1 , ZPZV<1 , ZPZV<1 , ZPZV<1 , ZPZV<1 , ZPZV<1 , ZPZV<1 
                       ZPZV<1>, ZPZV<1»; }; // NOLINT</pre>
02706 template<> struct ConwayPolynomial<2, 14> { using ZPZ = aerobus::zpz<2>; using type = POLYV<ZPZV<1>,
                       ZPZV<0>, ZPZV<0>, ZPZV<0>, ZPZV<0>, ZPZV<1>, ZPZV<10 , ZPZV<1 , ZP
                       ZPZV<0>, ZPZV<0>, ZPZV<1»; }; // NOLINT</pre>
 02707 template<> struct ConwayPolynomial<2, 15> { using ZPZ = aerobus::zpz<2>; using type = POLYV<ZPZV<1>,
                       ZPZV<0>, ZPZV<1>, ZPZV<1 , ZPZ
                       ZPZV<0>, ZPZV<1>, ZPZV<0>, ZPZV<1»; }; // NOLINT
02708 template<> struct ConwayPolynomial<2, 16> { using ZPZ = aerobus::zpz<2>; using type = POLYV<ZPZV<1>, ZPZV<0>, ZPZV<1>, ZPZV<1
02709 template<> struct ConwayPolynomial<2, 17> { using ZPZ = aerobus::zpz<2>; using type = POLYV<ZPZV<1>,
                       ZPZV<0>, ZPZ
                       ZPZV<0>, ZPZV<0>, ZPZV<1>, ZPZV<0>, ZPZV<0>, ZPZV<1»; }; // NOLINT</pre>
02710 template<> struct ConwayPolynomial<2, 18> { using ZPZ = aerobus::zpz<2>; using type = POLYV<ZPZV<1>,
                       ZPZV<0>, ZPZ
                       ZPZV<0>, ZPZV<0>, ZPZV<0>, ZPZV<0>, ZPZV<0>, ZPZV<1>, ZPZV<1»; }; // NOLINT</pre>
02711 template<> struct ConwayPolynomial<2, 19> { using ZPZ = aerobus::zpz<2>; using type = POLYV<ZPZV<1>,
                       ZPZV<0>, ZPZV<0>
                       ZPZV<0>, ZPZV<0>, ZPZV<1>, ZPZV<0>, ZPZV<0>, ZPZV<1>, ZPZV<1>, ZPZV<1>, ZPZV<1»; }; // NOLINT</pre>
02712 template<> struct ConwayPolynomial<2, 20> { using ZPZ = aerobus::zpz<2>; using type = POLYV<ZPZV<1>, ZPZV<0>, ZPZV<1>, ZPZV<1
02713 template<> struct ConwayPolynomial<3, 1> { using ZPZ = aerobus::zpz<3>; using type = POLYV<ZPZV<1>,
                       ZPZV<1»; }; // NOLINT</pre>
 02714 template<> struct ConwayPolynomial<3, 2> { using ZPZ = aerobus::zpz<3>; using type = POLYV<ZPZV<1>,
                       ZPZV<2>, ZPZV<2>; }; // NOLINT
 02715 template<> struct ConwayPolynomial<3, 3> { using ZPZ = aerobus::zpz<3>; using type = POLYV<ZPZV<1>,
                       ZPZV<0>, ZPZV<2>, ZPZV<1»; }; // NOLINT
 02716 template<> struct ConwayPolynomial<3, 4> { using ZPZ = aerobus::zpz<3>; using type = POLYV<ZPZV<1>,
                       ZPZV<2>, ZPZV<0>, ZPZV<0>, ZPZV<2»; };</pre>
                                                                                                                                                                         // NOLINT
 02717 template<> struct ConwayPolynomial<3, 5> { using ZPZ = aerobus::zpz<3>; using type = POLYV<ZPZV<1>,
                       ZPZV<0>, ZPZV<0>, ZPZV<0>, ZPZV<2>, ZPZV<1>*; }; // NOLINT
 02718 template<> struct ConwayPolynomial<3, 6> { using ZPZ = aerobus::zpz<3>; using type = POLYV<ZPZV<1>,
ZPZV<0>, ZPZV<2>, ZPZV<0>, ZPZV<1>, ZPZV<2>, ZPZV<2>, ZPZV<2>; }; // NOLINT

02719 template<> struct ConwayPolynomial<3, 7> { using ZPZ = aerobus::zpz<3>; using type = POLYV<ZPZV<1>,
                       ZPZV<0>, ZPZV<0>, ZPZV<0>, ZPZV<0>, ZPZV<2>, ZPZV<0>, ZPZV<1»; }; // NOLINT</pre>
 02720 template<> struct ConwayPolynomial<3, 8> { using ZPZ = aerobus::zpz<3>; using type = POLYV<ZPZV<1>,
                       ZPZV<0>, ZPZV<0>, ZPZV<2>, ZPZV<1>, ZPZV<0>, ZPZV<2>, ZPZV<2>, ZPZV<2>; }; // NOLINT
02721 template<> struct ConwayPolynomial<3, 9> { using ZPZ = aerobus::zpz<3>; using type = POLYV<ZPZV<1>, ZPZV<0>, ZPZV<0>, ZPZV<0>, ZPZV<0>, ZPZV<0>, ZPZV<2>, ZPZV<2>, ZPZV<2>, ZPZV<1>; }; // NOLINT
 02722 template<> struct ConwayPolynomial<3, 10> { using ZPZ = aerobus::zpz<3>; using type = POLYV<ZPZV<1>,
                       ZPZV<0>, ZPZV<0>, ZPZV<0>, ZPZV<2>, ZPZV<2>, ZPZV<2>, ZPZV<0>, ZPZV<0>, ZPZV<1>, ZPZV<2»; };</pre>
 02723 template<> struct ConwayPolynomial<3, 11> { using ZPZ = aerobus::zpz<3>; using type = POLYV<ZPZV<1>,
                      ZPZV<0>, ZPZV<0>, ZPZV<0>, ZPZV<0>, ZPZV<0>, ZPZV<0>, ZPZV<0>, ZPZV<0>, ZPZV<0>, ZPZV<2>, ZPZV<2>, ZPZV<2>, ZPZV<1»; };</pre>
                       // NOLINT
 02724 template<> struct ConwayPolynomial<3, 12> { using ZPZ = aerobus::zpz<3>; using type = POLYV<ZPZV<1>,
```

```
ZPZV<0>, ZPZV<0>, ZPZV<0>, ZPZV<0>, ZPZV<0>, ZPZV<1>, ZPZV<1>, ZPZV<1>, ZPZV<1>, ZPZV<0>,
                             ZPZV<2»; }; // NOLINT</pre>
02725 template<> struct ConwayPolynomial<3, 13> { using ZPZ = aerobus::zpz<3>; using type = POLYV<ZPZV<1>,
                             ZPZV<0>, ZPZ
                             ZPZV<2>. ZPZV<1»: }: // NOLINT
02726 template<> struct ConwayPolynomial<3, 14> { using ZPZ = aerobus::zpz<3>; using type = POLYV<ZPZV<1>,
                             ZPZV<0>, ZPZV<0>, ZPZV<0>, ZPZV<0>, ZPZV<2>, ZPZV<1>, ZPZV<1>, ZPZV<2>, ZPZV<2>, ZPZV<2>, ZPZV<3</pre>
                             ZPZV<1>, ZPZV<0>, ZPZV<2»; }; // NOLINT</pre>
02727 template<> struct ConwayPolynomial<3, 15> { using ZPZ = aerobus::zpz<3>; using type = POLYV<ZPZV<1>,
                              \texttt{ZPZV} < \texttt{0} >, \ \texttt{Z
                             ZPZV<0>, ZPZV<2>, ZPZV<1>, ZPZV<1»; }; // NOLINT
02728 template<> struct ConwayPolynomial<3, 16> { using ZPZ = aerobus::zpz<3>; using type = POLYV<ZPZV<1>,
                             ZPZV<0>, ZPZV<0>, ZPZV<0>, ZPZV<0>, ZPZV<0>, ZPZV<0>, ZPZV<0>, ZPZV<0>, ZPZV<0>, ZPZV<2>, ZPZV<2>
                             ZPZV<2>, ZPZV<2>, ZPZV<2>, ZPZV<1>, ZPZV<2»; }; // NOLINT</pre>
 02729 template<> struct ConwayPolynomial<3, 17> { using ZPZ = aerobus::zpz<3>; using type = POLYV<ZPZV<1>,
                            ZPZV<0>, ZPZV<0 , ZPZV<0 
02730 template<> struct ConwayPolynomial<3, 18> { using ZPZ = aerobus::zpz<3>; using type = POLYV<ZPZV<1>,
                             ZPZV<0>, ZPZV<2>, ZPZV<2>, ZPZV<2>, ZPZV<2>, ZPZV<3>, ZPZV<3 , ZPZ
                             ZPZV<2>, ZPZV<1>, ZPZV<2>, ZPZV<0>, ZPZV<2>, ZPZV<0>, ZPZV<0>, ZPZV<2»; }; // NOLINT</pre>
 02731 template<> struct ConwayPolynomial<3, 19> { using ZPZ = aerobus::zpz<3>; using type = POLYV<ZPZV<1>,
                             ZPZV<0>, ZPZ
                             02732 template<> struct ConwayPolynomial<3, 20> { using ZPZ = aerobus::zpz<3>; using type = POLYV<ZPZV<1>, ZPZV<0>, ZPZV<0>, ZPZV<0>, ZPZV<0>, ZPZV<0>, ZPZV<0>, ZPZV<1>, ZPZV<1>, ZPZV<1>, ZPZV<1>, ZPZV<1>, ZPZV<1>, ZPZV<2>, ZPZV<2>, ZPZV<2>, ZPZV<2>, ZPZV<2>, ZPZV<2>; }; // NOLINT
 02733 template<> struct ConwayPolynomial<5, 1> { using ZPZ = aerobus::zpz<5>; using type = POLYV<ZPZV<1>,
                             ZPZV<3»; }; // NOLINT</pre>
 02734 template<> struct ConwayPolynomial<5, 2> { using ZPZ = aerobus::zpz<5>; using type = POLYV<ZPZV<1>,
                             ZPZV<4>, ZPZV<2»; }; // NOLINT</pre>
 02735 template<> struct ConwayPolynomial<5, 3> { using ZPZ = aerobus::zpz<5>; using type = POLYV<ZPZV<1>,
                             ZPZV<0>, ZPZV<3>, ZPZV<3»; }; // NOLINT</pre>
 02736 template<> struct ConwayPolynomial<5, 4> { using ZPZ = aerobus::zpz<5>; using type = POLYV<ZPZV<1>,
                             ZPZV<0>, ZPZV<4>, ZPZV<4>, ZPZV<2»; }; // NOLINT
 02737 template<> struct ConwayPolynomial<5, 5> { using ZPZ = aerobus::zpz<5>; using type = POLYV<ZPZV<1>,
                            ZPZV<0>, ZPZV<0>, ZPZV<0>, ZPZV<4>, ZPZV<3»; }; // NOLINT</pre>
02738 template<> struct ConwayPolynomial<5, 6> { using ZPZ = aerobus::zpz<5>; using type = POLYV<ZPZV<1>,
                            ZPZV<0>, ZPZV<1>, ZPZV<4>, ZPZV<1>, ZPZV<0>, ZPZV<2»; }; // NOLINT</pre>
 02739 template<> struct ConwayPolynomial<5, 7> { using ZPZ = aerobus::zpz<5>; using type = POLYV<ZPZV<1>,
                             ZPZV<0>, ZPZV<0>, ZPZV<0>, ZPZV<0>, ZPZV<3>, ZPZV<3»; }; // NOLINT</pre>
 02740 template<> struct ConwayPolynomial<5, 8> { using ZPZ = aerobus::zpz<5>; using type = POLYV<ZPZV<1>,
ZPZV<0>, ZPZV<0>, ZPZV<0>, ZPZV<1>, ZPZV<1>, ZPZV<0>, ZPZV<3>, ZPZV<4>, ZPZV<4>; }; // NOLINT

02741 template<> struct ConwayPolynomial<5, 9> { using ZPZ = aerobus::zpz<5>; using type = POLYV<ZPZV<1>, ZPZV<0>, ZPZV<0 , 
 02742 template<> struct ConwayPolynomial<5, 10> { using ZPZ = aerobus::zpz<5>; using type = POLYV<ZPZV<1>,
                             ZPZV<0>, ZPZV<0>, ZPZV<0>, ZPZV<0>, ZPZV<3>, ZPZV<3>, ZPZV<2>, ZPZV<4>, ZPZV<1>, ZPZV<2»; };</pre>
                            NOLINT
02743 template<> struct ConwayPolynomial<5, 11> { using ZPZ = aerobus::zpz<5>; using type = POLYV<ZPZV<1>, ZPZV<0>, ZPZV<0>, ZPZV<0>, ZPZV<0>, ZPZV<0>, ZPZV<0>, ZPZV<0>, ZPZV<3>, ZPZV<3 , Z
                             // NOLINT
02744 template<> struct ConwayPolynomial<5, 12> { using ZPZ = aerobus::zpz<5>; using type = POLYV<ZPZV<1>,
                             ZPZV<0>, ZPZV<0>, ZPZV<0>, ZPZV<0>, ZPZV<1>, ZPZV<1>, ZPZV<0>, ZPZV<4>, ZPZV<4>, ZPZV<4>, ZPZV<3>, ZPZV<2>,
                             ZPZV<2»; }; // NOLINT</pre>
02745 template<> struct ConwayPolynomial<5, 13> { using ZPZ = aerobus::zpz<5>; using type = POLYV<ZPZV<1>,
                             ZPZV<0>, ZPZV<0>
                             ZPZV<3>, ZPZV<3»; }; // NOLINT</pre>
02746 template<> struct ConwayPolynomial<5, 14> { using ZPZ = aerobus::zpz<5>; using type = POLYV<ZPZV<1>,
                             ZPZV<0>, ZPZV<0>, ZPZV<0>, ZPZV<0>, ZPZV<0>, ZPZV<1>, ZPZV<4>, ZPZV<4>, ZPZV<4>, ZPZV<2>, ZPZV<3>,
                             ZPZV<0>, ZPZV<1>, ZPZV<2»; }; // NOLINT</pre>
02747 template<> struct ConwayPolynomial<5, 15> { using ZPZ = aerobus::zpz<5>; using type = POLYV<ZPZV<1>,
                            ZPZV<0>, ZPZ
02748 template<> struct ConwayPolynomial<5, 16> { using ZPZ = aerobus::zpz<5>; using type = POLYV<ZPZV<1>,
                             ZPZV<0>, ZPZV<0>, ZPZV<0>, ZPZV<0>, ZPZV<0>, ZPZV<0>, ZPZV<0>, ZPZV<0>, ZPZV<0>, ZPZV<1>, ZPZV<4>, ZPZV<4>, ZPZV<4>,
                             ZPZV<2>, ZPZV<4>, ZPZV<4>, ZPZV<1>, ZPZV<2»; }; // NOLINT</pre>
02749 template<> struct ConwayPolynomial<5, 17> { using ZPZ = aerobus::zpz<5>; using type = POLYV<ZPZV<1>,
                             ZPZV<0>, ZPZV<0>
ZPZV<0>, ZPZV<0>, ZPZV<0>, ZPZV<3>, ZPZV<3>, ZPZV<2>, ZPZV<2»; }; // NOLINT

02750 template<> struct ConwayPolynomial<5, 18> { using ZPZ = aerobus::zpz<5>; using type = POLYV<ZPZV<1>,
                             ZPZV<0>, ZPZV<0>, ZPZV<0>, ZPZV<0>, ZPZV<0>, ZPZV<1>, ZPZV<1>, ZPZV<1>, ZPZV<1>, ZPZV<2>, ZPZV<0>,
                             ZPZV<2>, ZPZV<1>, ZPZV<0>, ZPZV<2>, ZPZV<2>, ZPZV<0>, ZPZV<2»; }; // NOLINT</pre>
 02751 template<> struct ConwayPolynomial<5, 19> { using ZPZ = aerobus::zpz<5>; using type = POLYV<ZPZV<1>,
                            ZPZV<0>, ZPZV<0>
02752 template<> struct ConwayPolynomial<5, 20> { using ZPZ = aerobus::zpz<5>; using type = POLYV<ZPZV<1>,
                             ZPZV<0>, ZPZV<0>, ZPZV<0>, ZPZV<0>, ZPZV<0>, ZPZV<0>, ZPZV<0>, ZPZV<0>, ZPZV<3>, ZPZV<3>, ZPZV<4>, ZPZV<4>, ZPZV<3>,
ZPZV<2>, ZPZV<3>, ZPZV<3>, ZPZV<3>, ZPZV<0>, ZPZV<4>, ZPZV<4>, ZPZV<4>, ZPZV<2>; 3; // NOLINT 02753 template<> struct ConwayPolynomial<7, 1> { using ZPZ = aerobus::zpz<7>; using type = POLYV<ZPZV<1>,
                            ZPZV<4»; }; // NOLINT
02754 template<> struct ConwayPolynomial<7, 2> { using ZPZ = aerobus::zpz<7>; using type = POLYV<ZPZV<1>,
                             ZPZV<6>, ZPZV<3»; }; // NOLINT
02755 template<> struct ConwayPolynomial<7, 3> { using ZPZ = aerobus::zpz<7>; using type = POLYV<ZPZV<1>,
                             ZPZV<6>, ZPZV<0>, ZPZV<4»; }; // NOLINT
 02756 template<> struct ConwayPolynomial<7, 4> { using ZPZ = aerobus::zpz<7>; using type = POLYV<ZPZV<1>,
```

```
02758 template<> struct ConwayPolynomial<7, 6> { using ZPZ = aerobus::zpz<7>; using type = POLYV<ZPZV<1>,
                                    ZPZV<0>, ZPZV<1>, ZPZV<5>, ZPZV<4>, ZPZV<6>, ZPZV<3»; }; // NOLINT</pre>
  02759 template<> struct ConwayPolynomial<7, 7> { using ZPZ = aerobus::zpz<7>; using type = POLYV<ZPZV<1>,
 ZPZV<0>, ZPZV<0>, ZPZV<0>, ZPZV<0>, ZPZV<0>, ZPZV<0>, ZPZV<6>, ZPZV<4»; }; // NOLINT
02760 template<> struct ConwayPolynomial<7, 8> { using ZPZ = aerobus::zpz<7>; using type = POLYV<ZPZV<1>,
                                   ZPZV<0>, ZPZV<0>, ZPZV<0>, ZPZV<0>, ZPZV<4>, ZPZV<6>, ZPZV<2>, ZPZV<3»; }; // NOLINT
  02761 template<> struct ConwayPolynomial<7, 9> { using ZPZ = aerobus::zpz<7>; using type = POLYV<ZPZV<1>,
                                    ZPZV<0>, ZPZV<0>, ZPZV<0>, ZPZV<0>, ZPZV<6>, ZPZV<1>, ZPZV<0>, ZPZV<6>, ZPZV<6>, ZPZV<6>, ZPZV<6</pre>
  02762 template<> struct ConwayPolynomial<7, 10> { using ZPZ = aerobus::zpz<7>; using type = POLYV<ZPZV<1>
                                   ZPZV<0>, ZPZV<0>, ZPZV<0>, ZPZV<1>, ZPZV<1>, ZPZV<4>, ZPZV<1>, ZPZV<2>, ZPZV<3>, ZPZV<3»; }; //</pre>
                                  NOLINT
02763 template<> struct ConwayPolynomial<7, 11> { using ZPZ = aerobus::zpz<7>; using type = POLYV<ZPZV<1>, ZPZV<0>, ZPZV<0>, ZPZV<0>, ZPZV<0>, ZPZV<0>, ZPZV<0>, ZPZV<0>, ZPZV<0>, ZPZV<1>, ZPZV<4»; };
                                     // NOLINT
  02764 template<> struct ConwayPolynomial<7, 12> { using ZPZ = aerobus::zpz<7>; using type = POLYV<ZPZV<1>,
                                    \texttt{ZPZV} < \texttt{0} >, \ \texttt{ZPZV} < \texttt{0} >, \ \texttt{ZPZV} < \texttt{2} >, \ \texttt{ZPZV} < \texttt{2} >, \ \texttt{ZPZV} < \texttt{3} >, \ \texttt{ZPZV} < \texttt{2} >, \ \texttt{ZPZV} < \texttt{4} >, \ \texttt{ZPZV} < \texttt{0} >, \ \texttt{ZPZV} < \texttt{5} >, \ \texttt{ZPZV} < \texttt{6} >, \ \texttt{2PZV} < \texttt{6} >, \ \texttt{2
                                   ZPZV<3»; }; // NOLINT</pre>
02765 template<> struct ConwayPolynomial<7, 13> { using ZPZ = aerobus::zpz<7>; using type = POLYV<ZPZV<1>, ZPZV<0>, ZPZV
                                    ZPZV<0>, ZPZV<4»; }; // NOLINT</pre>
  02766 template<> struct ConwayPolynomial<7, 14> { using ZPZ = aerobus::zpz<7>; using type = POLYV<ZPZV<1>,
                                   ZPZV<0>, ZPZV<0>, ZPZV<0>, ZPZV<0>, ZPZV<0>, ZPZV<0>, ZPZV<5>, ZPZV<6>, ZPZV<6>, ZPZV<6>, ZPZV<6>,
                                   ZPZV<3>, ZPZV<6>, ZPZV<3»; }; // NOLINT</pre>
02767 template<> struct ConwayPolynomial<7, 15> { using ZPZ = aerobus::zpz<7>; using type = POLYV<ZPZV<1>, ZPZV<0>, ZPZV<0>, ZPZV<0>, ZPZV<0>, ZPZV<0>, ZPZV<0>, ZPZV<0>, ZPZV<6>, ZPZV<6 ) , ZPZV | Z
                                   ZPZV<4>, ZPZV<1>, ZPZV<2>, ZPZV<4»; }; // NOLINT</pre>
  02768 template<> struct ConwayPolynomial<7,
                                                                                                                                                                                                                                                       16> { using ZPZ = aerobus::zpz<7>; using type = POLYV<ZPZV<1>,
                                    \texttt{ZPZV} < \texttt{0} >, \ \texttt{ZPZV} < \texttt{4} >, \ \texttt{ZPZV} < \texttt{5} >, \ \texttt{ZPZV} < \texttt{3} >, \ \texttt{ZPZV} < \texttt{4} >, \ \texttt{ZPZV} < \texttt{5} >, \ \texttt{ZPZV} < \texttt{4} >, \ \texttt{ZPZV} < \texttt{5} >, \ \texttt{ZPZV} < \texttt{4} >, \ \texttt{ZPZV} < \texttt{5} >, \ \texttt{2PZV} < \texttt{5} >, \ \texttt{2
ZPZV<1>, ZPZV<6>, ZPZV<2>, ZPZV<4>, ZPZV<3»; }; // NOLINT
02769 template<> struct ConwayPolynomial<7, 17> { using ZPZ = aerobus::zpz<7>; using type = POLYV<ZPZV<1>,
                                  ZPZV<0>, ZPZV<0 , ZPZ
  02770 template<> struct ConwayPolynomial<7, 18> { using ZPZ = aerobus::zpz<7>; using type = POLYV<ZPZV<1>,
                                    \texttt{ZPZV} < \texttt{0} >, \ \texttt{ZPZV} < \texttt{1} >, \ \texttt{ZPZV} < \texttt{2} >, \ \texttt{ZPZV} < \texttt{4} >, \ \texttt{ZPZV} < \texttt{5} >, \ \texttt{ZPZV} < \texttt{5} >, \ \texttt{2PZV} < \texttt{5} >, \ \texttt{2
ZPZV<1>, ZPZV<3>, ZPZV<0>, ZPZV<0>, ZPZV<6>, ZPZV<2>, ZPZV<2>, ZPZV<3>; }; // NOLINT

02771 template<> struct ConwayPolynomial<7, 19> { using ZPZ = aerobus::zpz<7>; using type = POLYV<ZPZV<1>, ZPZV<0>, ZPZV<0 , ZPZV<0 ,
  02772 template<> struct ConwayPolynomial<7, 20> { using ZPZ = aerobus::zpz<7>; using type = POLYV<ZPZV<1>,
ZPZV<0>, ZPZV<0>, ZPZV<0>, ZPZV<0>, ZPZV<0>, ZPZV<0>, ZPZV<0>, ZPZV<0>, ZPZV<0>, ZPZV<6>, ZPZV<6>, ZPZV<6>, ZPZV<6>, ZPZV<5>, ZPZV<5>, ZPZV<2>, ZPZV<3>, ZPZV<3>, ZPZV<1>, ZPZV<3>, ZPZV<1>, ZPZV<1
, ZPZV
, ZPZV
, ZPZV
, ZPZV
, ZP
                                  ZPZV<9»; }; // NOLINT
  02774 template<> struct ConwayPolynomial<11, 2> { using ZPZ = aerobus::zpz<11>; using type = POLYV<ZPZV<1>,
                                  ZPZV<7>, ZPZV<2»; }; // NOLINT</pre>
  02775 template<> struct ConwayPolynomial<11, 3> { using ZPZ = aerobus::zpz<11>; using type = POLYV<ZPZV<1>,
                                   ZPZV<0>, ZPZV<2>, ZPZV<9»; }; // NOLINT</pre>
  02776 template<> struct ConwayPolynomial<11, 4> { using ZPZ = aerobus::zpz<11>; using type = POLYV<ZPZV<1>,
                                  ZPZV<0>, ZPZV<8>, ZPZV<10>, ZPZV<2»; };</pre>
                                                                                                                                                                                                                                                                       // NOLINT
  02777 template<> struct ConwayPolynomial<11, 5> { using ZPZ = aerobus::zpz<11>; using type = POLYV<ZPZV<1>,
                                   ZPZV<0>, ZPZV<0>, ZPZV<10>, ZPZV<0>, ZPZV<9»; }; // NOLINT</pre>
  02778 template<> struct ConwayPolynomial<11, 6> { using ZPZ = aerobus::zpz<11>; using type = POLYV<ZPZV<1>,
                                   ZPZV<0>, ZPZV<3>, ZPZV<4>, ZPZV<6>, ZPZV<7>, ZPZV<2»; }; // NOLINT
  02779 template<> struct ConwayPolynomial<11, 7> { using ZPZ = aerobus::zpz<11>; using type = POLYV<ZPZV<1>,
                                  ZPZV<0>, ZPZV<0>, ZPZV<0>, ZPZV<0>, ZPZV<4>, ZPZV<9»; }; // NOLINT</pre>
  02780 template<> struct ConwayPolynomial<11, 8> { using ZPZ = aerobus::zpz<11>; using type = POLYV<ZPZV<1>,
                                   ZPZV<0>, ZPZV<0>, ZPZV<0>, ZPZV<7>, ZPZV<7
  02781 template<> struct ConwayPolynomial<11, 9> { using ZPZ = aerobus::zpz<11>, using type = POLYV<ZPZV<1>,
                                   02782 template<> struct ConwayPolynomial<11, 10> { using ZPZ = aerobus::zpz<11>; using type = POLYV<ZPZV<1>,
                                   ZPZV<0>, ZPZV<0>, ZPZV<0>, ZPZV<0>, ZPZV<7>, ZPZV<8>, ZPZV<8>, ZPZV<6>, ZPZV<6>, ZPZV<6>, ZPZV<2»; }; //</pre>
                                  NOLINT
 02783 template<> struct ConwayPolynomial<11, 11> { using ZPZ = aerobus::zpz<11>; using type = POLYV<ZPZV<1>,
                                   ZPZV<0>, ZPZV<10>, ZP
                                     // NOLINT
 02784 template<> struct ConwayPolynomial<11, 12> { using ZPZ = aerobus::zpz<11>; using type = POLYV<ZPZV<1>,
                                   ZPZV<0>, ZPZV<0>, ZPZV<0>, ZPZV<1>, ZPZV<4>, ZPZV<4>, ZPZV<5>, ZPZV<5>, ZPZV<5>, ZPZV<5>, ZPZV<5</pre>
                                   ZPZV<2»: }: // NOLINT
 02785 template<> struct ConwayPolynomial<11, 13> { using ZPZ = aerobus::zpz<11>; using type = POLYV<ZPZV<1>,
                                    ZPZV<0>, ZPZV<0</pre>, ZPZV<0>, ZP
                                    ZPZV<7>, ZPZV<9»; }; // NOLINT</pre>
 02786 template<> struct ConwayPolynomial<11, 14> { using ZPZ = aerobus::zpz<11>; using type = POLYV<ZPZV<1>,
                                   ZPZV<0>, ZPZV<0>, ZPZV<0>, ZPZV<0>, ZPZV<0>, ZPZV<0>, ZPZV<2>, ZPZV<5, ZPZV<5, ZPZV<4>, ZPZV<4>, ZPZV<5, ZPZV<
                                   ZPZV<6>, ZPZV<10>, ZPZV<2»; }; // NOLINT</pre>
 02787 template<> struct ConwayPolynomial<11, 15> { using ZPZ = aerobus::zpz<11>; using type = POLYV<ZPZV<1>,
                                    ZPZV<0>, ZPZV<10>, ZPZV<7>, ZPZV<7>, ZPZV<0>,
                                    ZPZV<5>, ZPZV<0>, ZPZV<0>, ZPZV<9»; }; // NOLINT</pre>
 02788 template<> struct ConwayPolynomial<11, 16> { using ZPZ = aerobus::zpz<11>, using type = POLYV<ZPZV<1>,
                                  ZPZV<0>, ZPZV<0>, ZPZV<0>, ZPZV<0>, ZPZV<0>, ZPZV<0>, ZPZV<0>, ZPZV<0>, ZPZV<1>, ZPZV<1>, ZPZV<1>, ZPZV<1>, ZPZV<1>, ZPZV<1>, ZPZV<1>, ZPZV<3>,
 02789 template<> struct ConwayPolynomial<11, 17> { using ZPZ = aerobus::zpz<11>; using type = POLYV<ZPZV<1>,
                                    ZPZV<0>, ZPZV<0>
 ZPZV<0>, ZPZV<0>, ZPZV<0>, ZPZV<0>, ZPZV<4>, ZPZV<9>; }; // NOLINT

02790 template<> struct ConwayPolynomial<11, 18> { using ZPZ = aerobus::zpz<11>; using type = POLYV<ZPZV<1>,
 ZPZV<0>, ZPZV<0>, ZPZV<0>, ZPZV<0>, ZPZV<0>, ZPZV<3>, ZPZV<3>, ZPZV<8>, ZPZV<10>, ZPZV<8>, ZPZV<3>, ZPZV<3>, ZPZV<3>, ZPZV<3>, ZPZV<3>, ZPZV<3>, ZPZV<3>, ZPZV<3>, ZPZV<3>, ZPZV<2>, ZPZV<2>, ZPZV<2>, ZPZV<2>; }; // NOLINT

02791 template<>> struct ConwayPolynomial<11, 19> { using ZPZ = aerobus::zpz<11>; using type = POLYV<ZPZV<1>,
```

```
ZPZV<0>, ZPZV<0>
                         ZPZV<0>, ZPZV<0>, ZPZV<0>, ZPZV<0>, ZPZV<0>, ZPZV<8>, ZPZV<2>, ZPZV<2»; }; // NOLINT</pre>
02792 template<> struct ConwayPolynomial<11, 20> { using ZPZ = aerobus::zpz<11>; using type = POLYV<ZPZV<1>,
                        ZPZV<0>, ZPZV<0>, ZPZV<0>, ZPZV<0>, ZPZV<0>, ZPZV<0>, ZPZV<0>, ZPZV<0>, ZPZV<0>, ZPZV<1>, ZPZV<1>, ZPZV<1>, ZPZV<1>, ZPZV<9>, ZPZV<1>, ZPZV<5>, ZPZV<5 >, ZPZV<5 
02793 template>> struct ConwayPolynomial<13, 1> { using ZPZ = aerobus::zpz<13>; using type = POLYV<ZPZV<1>,
                         ZPZV<11»; }; // NOLINT</pre>
 02794 template<> struct ConwayPolynomial<13, 2> { using ZPZ = aerobus::zpz<13>; using type = POLYV<ZPZV<1>,
                         ZPZV<12>, ZPZV<2»; }; // NOLINT</pre>
 02795 template<> struct ConwayPolynomial<13, 3> { using ZPZ = aerobus::zpz<13>; using type = POLYV<ZPZV<1>,
                         ZPZV<0>, ZPZV<2>, ZPZV<11»; }; // NOLINT</pre>
 02796 template<> struct ConwayPolynomial<13, 4> { using ZPZ = aerobus::zpz<13>; using type = POLYV<ZPZV<1>,
                                                                                                                                                                                           // NOLINT
                         ZPZV<0>, ZPZV<3>, ZPZV<12>, ZPZV<2»; };</pre>
 02797 template<> struct ConwayPolynomial<13, 5> { using ZPZ = aerobus::zpz<13>; using type = POLYV<ZPZV<1>,
                         ZPZV<0>, ZPZV<0>, ZPZV<0>, ZPZV<4>, ZPZV<11»; }; // NOLINT</pre>
 02798 template<> struct ConwayPolynomial<13, 6> { using ZPZ = aerobus::zpz<13>; using type = POLYV<ZPZV<1>,
                        ZPZV<0>, ZPZV<0>, ZPZV<10>, ZPZV<11>, ZPZV<11>, ZPZV<2»; }; // NOLINT</pre>
02799 template<> struct ConwayPolynomial<13, 7> { using ZPZ = aerobus::zpz<13>; using type = POLYV<ZPZV<1>,
                         ZPZV<0>, ZPZV<0>, ZPZV<0>, ZPZV<0>, ZPZV<3>, ZPZV<11»; }; // NOLINT</pre>
02800 template<> struct ConwayPolynomial<13, 8> { using ZPZ = aerobus::zpz<13>; using type = POLYV<ZPZV<1>,
ZPZV<0>, ZPZV<0>, ZPZV<0>, ZPZV<8>, ZPZV<12>, ZPZV<2>, ZPZV<2>, ZPZV<2»; }; // NOLINT
02801 template<> struct ConwayPolynomial<13, 9> { using ZPZ = aerobus::zpz<13>; using type = POLYV<ZPZV<1>,
NOLINT
 02803 template<> struct ConwayPolynomial<13, 11> { using ZPZ = aerobus::zpz<13>; using type = POLYV<ZPZV<1>,
                         ZPZV<0>, ZPZV<3>, ZPZV<3</pre>
                          // NOLINT
02804 template<> struct ConwayPolynomial<13, 12> { using ZPZ = aerobus::zpz<13>; using type = POLYV<ZPZV<1>,
                         ZPZV<0>, ZPZV<0>, ZPZV<0>, ZPZV<1>, ZPZV<5>, ZPZV<8>, ZPZV<11>, ZPZV<3>, ZPZV<1>, ZPZV<1>, ZPZV<4>,
                         ZPZV<2»; }; // NOLINT</pre>
 02805 template<> struct ConwayPolynomial<13, 13> { using ZPZ = aerobus::zpz<13>; using type = POLYV<ZPZV<1>,
                        ZPZV<0>, ZPZV<12>, ZPZV<11»; }; // NOLINT</pre>
02806 template<> struct ConwayPolynomial<13, 14> { using ZPZ = aerobus::zpz<13>; using type = POLYV<ZPZV<1>,
                        ZPZV<0>, ZPZV<0>, ZPZV<0>, ZPZV<0>, ZPZV<0>, ZPZV<0>, ZPZV<0>, ZPZV<0>, ZPZV<7>, ZPZV<10>, ZPZV<
 02807 template<> struct ConwayPolynomial<13, 15> { using ZPZ = aerobus::zpz<13>; using type = POLYV<ZPZV<1>,
ZPZV<0>, ZPZV<0>, ZPZV<0>, ZPZV<0>, ZPZV<0>, ZPZV<0>, ZPZV<0>, ZPZV<0>, ZPZV<0>, ZPZV<2>, ZPZV<12>, ZPZV<12>, ZPZV<11>, ZPZV<11>, ZPZV<11>, ZPZV<11»; }; // NOLINT

02808 template<> struct ConwayPolynomial<13, 16> { using ZPZ = aerobus::zpz<13>; using type = POLYV<ZPZV<1>,
                        ZPZV<0>, ZPZV<0>, ZPZV<0>, ZPZV<0>, ZPZV<0>, ZPZV<0>, ZPZV<0>, ZPZV<0>, ZPZV<2>, ZPZV<2>, ZPZV<2>, ZPZV<2>, ZPZV<3>, ZPZV<12>, ZPZV<12>, ZPZV<5>, ZPZV<5-, Z
 02809 template<> struct ConwayPolynomial<13, 17> { using ZPZ = aerobus::zpz<13>; using type = POLYV<ZPZV<1>,
                         ZPZV<0>, ZPZ
ZPZV<0>, ZPZV<0>, ZPZV<0>, ZPZV<10>, ZPZV<6>, ZPZV<11»; }; // NOLINT
02810 template<> struct ConwayPolynomial<13, 18> { using ZPZ = aerobus::zpz<13>; using type = POLYV<ZPZV<1>,
                        ZPZV<0>, ZPZV<0>, ZPZV<0>, ZPZV<0>, ZPZV<0>, ZPZV<0>, ZPZV<0>, ZPZV<1>, ZPZV<1>, ZPZV<1>, ZPZV<1>, ZPZV<2>, ZPZV<2>, ZPZV<2>, ZPZV<3>, ZPZV<3>, ZPZV<3>, ZPZV<3>, ZPZV<5>, ZPZV<5>, ZPZV<5>, ZPZV<5>, ZPZV<6>, ZPZV<6>, ZPZV<6>, ZPZV<6>, ZPZV<6>, ZPZV<6 , ZPZ
02811 template<> struct ConwayPolynomial<13, 19> { using ZPZ = aerobus::zpz<13>; using type = POLYV<ZPZV<1>,
                         ZPZV<0>, ZPZV<0>
02812 template<> struct ConwayPolynomial<13, 20> { using ZPZ = aerobus::zpz<13>; using type = POLYV<ZPZV<1>, ZPZV<0>, ZP
                         ZPZV<7>, ZPZV<8>, ZPZV<7>, ZPZV<4>, ZPZV<0>, ZPZV<4>, ZPZV<8>, ZPZV<11>, ZPZV<2»; }; // NOLINT</pre>
 02813 template<> struct ConwayPolynomial<17, 1> { using ZPZ = aerobus::zpz<17>; using type = POLYV<ZPZV<1>,
                         ZPZV<14»; }; // NOLINT
 02814 template<> struct ConwayPolynomial<17, 2> { using ZPZ = aerobus::zpz<17>; using type = POLYV<ZPZV<1>,
ZPZV<16>, ZPZV<3»; }; // NOLINT
02815 template<> struct ConwayPolynomial<17, 3> { using ZPZ = aerobus::zpz<17>; using type = POLYV<ZPZV<1>,
                        ZPZV<0>, ZPZV<1>, ZPZV<14»; }; // NOLINT</pre>
 02816 template<> struct ConwayPolynomial<17, 4> { using ZPZ = aerobus::zpz<17>; using type = POLYV<ZPZV<1>,
                                                                                                                                                                                         // NOLINT
                         ZPZV<0>, ZPZV<7>, ZPZV<10>, ZPZV<3»; };</pre>
 02817 template<> struct ConwayPolynomial<17, 5> { using ZPZ = aerobus::zpz<17>; using type = POLYV<ZPZV<1>,
                        {\tt ZPZV<0>}, {\tt ZPZV<0>}, {\tt ZPZV<0>}, {\tt ZPZV<1>}, {\tt ZPZV<14*}; }; // NOLINT
 02818 template<> struct ConwayPolynomial<17, 6> { using ZPZ = aerobus::zpz<17>; using type = POLYV<ZPZV<1>,
                         ZPZV<0>, ZPZV<2>, ZPZV<0>, ZPZV<10>, ZPZV<3>, ZPZV<3»; }; // NOLINT</pre>
 02819 template<> struct ConwayPolynomial<17, 7> { using ZPZ = aerobus::zpz<17>; using type = POLYV<ZPZV<1>,
                         ZPZV<0>, ZPZV<0>, ZPZV<0>, ZPZV<0>, ZPZV<0>, ZPZV<12>, ZPZV<14»; }; // NOLINT</pre>
 02820 template<> struct ConwayPolynomial<17, 8> { using ZPZ = aerobus::zpz<17>; using type = POLYV<ZPZV<1>,
                         ZPZV<0>, ZPZV<0>, ZPZV<0>, ZPZV<11>, ZPZV<12>, ZPZV<0>, ZPZV<6>, ZPZV<3»; }; // NOLINT</pre>
02821 template<> struct ConwayPolynomial<17, 9> { using ZPZ = aerobus::zpz<17>; using type = POLYV<ZPZV<1>, ZPZV<0>, ZPZV<0>, ZPZV<0>, ZPZV<0>, ZPZV<0>, ZPZV<0>, ZPZV<0>, ZPZV<0>, ZPZV<0>, ZPZV<14»; }; // NOLINT
02822 template<> struct ConwayPolynomial<17, 10> { using ZPZ = aerobus::zpz<17>; using type = POLYV<ZPZV<1>,
                         ZPZV<0>, ZPZV<0>, ZPZV<0>, ZPZV<0>, ZPZV<13>, ZPZV<6>, ZPZV<5>, ZPZV<9>, ZPZV<12>, ZPZV<3»; }; //</pre>
02823 template<> struct ConwayPolynomial<17, 11> { using ZPZ = aerobus::zpz<17>; using type = POLYV<ZPZV<1>,
                         ZPZV<0>, ZPZV<5>, ZPZV<5>, ZPZV<14»; };</pre>
                          // NOLINT
02824 template<> struct ConwayPolynomial<17, 12> { using ZPZ = aerobus::zpz<17>; using type = POLYV<ZPZV<1>,
                         ZPZV<0>, ZPZV<0>, ZPZV<0>, ZPZV<1>, ZPZV<4>, ZPZV<14>, ZPZV<14>, ZPZV<14>, ZPZV<13>, ZPZV<6>, ZPZV<614>, ZPZV<14>, ZPZV<9>,
                         ZPZV<3»; }; // NOLINT</pre>
 02825 template<> struct ConwayPolynomial<17, 13> { using ZPZ = aerobus::zpz<17>; using type = POLYV<ZPZV<1>,
                        ZPZV<0>, ZPZV<0>
 02826 template<> struct ConwayPolynomial<17, 14> { using ZPZ = aerobus::zpz<17>; using type = POLYV<ZPZV<1>,
```

```
ZPZV<0>, ZPZV<0>, ZPZV<0>, ZPZV<0>, ZPZV<0>, ZPZV<1>, ZPZV<11>, ZPZV<1>, ZPZV<8>, ZPZV<16>, ZPZV<13>,
                   ZPZV<9>, ZPZV<3>, ZPZV<3»; }; // NOLINT</pre>
02827 template<> struct ConwayPolynomial<17, 15> { using ZPZ = aerobus::zpz<17>; using type = POLYV<ZPZV<1>,
ZPZV<0>, ZPZV<0>, ZPZV<0>, ZPZV<0>, ZPZV<0>, ZPZV<0>, ZPZV<0>, ZPZV<0>, ZPZV<0>, ZPZV<4>, ZPZV<4>, ZPZV<4>, ZPZV<46, ZPZV<44, ZPZV<16>, ZPZV<66>, ZPZV<44, ZPZV<14>, ZPZV<14»; }; // NOLINT</pre>
02828 template<> struct ConwayPolynomial<17, 16> { using ZPZ = aerobus::zpz<17>; using type = POLYV<ZPZV<1>,
                   ZPZV<0>, ZPZV<0>, ZPZV<0>, ZPZV<0>, ZPZV<0>, ZPZV<0>, ZPZV<0>, ZPZV<0>, ZPZV<1>, ZPZV<13>, ZPZV<13>, ZPZV<5>, ZPZV<2>,
                   ZPZV<12>, ZPZV<13>, ZPZV<12>, ZPZV<1>, ZPZV<3»; };</pre>
                                                                                                                                                                                  // NOLINT
02829 template<> struct ConwayPolynomial<17, 17> { using ZPZ = aerobus::zpz<17>; using type = POLYV<ZPZV<1>,
ZPZV<0>, ZPZV<0>
                   ZPZV<0>, ZPZV<0>, ZPZV<0>, ZPZV<0>, ZPZV<0>, ZPZV<0>, ZPZV<1>, ZPZV<0>, ZPZV<16>, ZPZV<7>, ZPZV<1>,
ZPZV<07, ZPZV<97, ZPZV<11>, ZPZV<13>, ZPZV<13>, ZPZV<9>, ZPZV<9>, ZPZV<9>, ZPZV<11>, ZPZV<13>, Z
ZPZV<0>, ZPZV<0>
                   ZPZV<0>, ZPZV<0>, ZPZV<0>, ZPZV<0>, ZPZV<0>, ZPZV<0>, ZPZV<0>, ZPZV<0>, ZPZV<1>, ZPZV<5>, ZPZV<5>, ZPZV<16>, ZPZV<14>,
                   ZPZV<13>, ZPZV<3>, ZPZV<14>, ZPZV<9>, ZPZV<1>, ZPZV<13>, ZPZV<2>, ZPZV<5>, ZPZV<5>, ZPZV<5>, ZPZV<5</pre>
 02833 template<> struct ConwayPolynomial<19, 1> { using ZPZ = aerobus::zpz<19>; using type = POLYV<ZPZV<1>,
                   ZPZV<17»; }; // NOLINT</pre>
 02834 template<> struct ConwayPolynomial<19, 2> { using ZPZ = aerobus::zpz<19>; using type = POLYV<ZPZV<1>,
                  ZPZV<18>, ZPZV<2»; }; // NOLINT</pre>
 02835 template<> struct ConwayPolynomial<19, 3> { using ZPZ = aerobus::zpz<19>; using type = POLYV<ZPZV<1>,
                   ZPZV<0>, ZPZV<4>, ZPZV<17»; }; // NOLINT</pre>
 02836 template<> struct ConwayPolynomial<19, 4> { using ZPZ = aerobus::zpz<19>; using type = POLYV<ZPZV<1>,
                   ZPZV<0>, ZPZV<2>, ZPZV<11>, ZPZV<2»; };</pre>
                                                                                                                                               // NOLINT
 02837 template<> struct ConwayPolynomial<19, 5> { using ZPZ = aerobus::zpz<19>; using type = POLYV<ZPZV<1>,
02839 template<> struct ConwayPolynomial<19, 7> { using ZPZ = aerobus::zpz<19>; using type = POLYV<ZPZV<1>,
                   ZPZV<0>, ZPZV<0>, ZPZV<0>, ZPZV<0>, ZPZV<6>, ZPZV<6>, ZPZV<17»; }; // NOLINT</pre>
 02840 template<> struct ConwayPolynomial<19, 8> { using ZPZ = aerobus::zpz<19>; using type = POLYV<ZPZV<1>,
                  ZPZV<0>, ZPZV<0>, ZPZV<0>, ZPZV<1>, ZPZV<12>, ZPZV<10>, ZPZV<3>, ZPZV<2»; }; // NOLINT
02841 template<> struct ConwayPolynomial<19, 9> { using ZPZ = aerobus::zpz<19>; using type = POLYV<ZPZV<1>, ZPZV<0>, ZPZV<0>, ZPZV<0>, ZPZV<0>, ZPZV<0>, ZPZV<0>, ZPZV<1>, ZPZV<14>, ZPZV<16>, ZPZV<17»; }; // NOLINT
 02842 template<> struct ConwayPolynomial<19, 10> { using ZPZ = aerobus::zpz<19>; using type = POLYV<ZPZV<1>,
                   ZPZV<0>, ZPZV<0>, ZPZV<0>, ZPZV<0>, ZPZV<18>, ZPZV<13>, ZPZV<17>, ZPZV<3>, ZPZV<4>, ZPZV<4>, ZPZV<4>, ZPZV<4</pre>
                   NOLTNT
02843 template<> struct ConwayPolynomial<19, 11> { using ZPZ = aerobus::zpz<19>; using type = POLYV<ZPZV<1>,
                  ZPZV<0>, ZPZV<0>, ZPZV<0>, ZPZV<0>, ZPZV<0>, ZPZV<0>, ZPZV<0>, ZPZV<0>, ZPZV<0>, ZPZV<17»; };</pre>
                    // NOLINT
 02844 template<> struct ConwayPolynomial<19, 12> { using ZPZ = aerobus::zpz<19>; using type = POLYV<ZPZV<1>,
                   ZPZV<0>, ZPZV<0>, ZPZV<0>, ZPZV<0>, ZPZV<1>, ZPZV<2>, ZPZV<18>, ZPZV<2>, ZPZV<2>, ZPZV<9>, ZPZV<16>, ZPZV<16>, ZPZV<10</pre>
                   ZPZV<2»; }; // NOLINT</pre>
02845 template<> struct ConwayPolynomial<19, 13> { using ZPZ = aerobus::zpz<19>; using type = POLYV<ZPZV<1>, ZPZV<0>, ZPZV<1>, ZPZV<1>, ZPZV<1>, ZPZV<1>, ZPZV<1>, ZPZV<1>, ZPZV<10>, ZPZV<10>, ZPZV<0>, 
02846 template<> struct ConwayPolynomial<19, 14> { using ZPZ = aerobus::zpz<19>; using type = POLYV<ZPZV<1>,
                   ZPZV<0>, ZPZV<0>, ZPZV<0>, ZPZV<0>, ZPZV<0>, ZPZV<0>, ZPZV<0>, ZPZV<11>, ZPZV<11>, ZPZV<11>, ZPZV<1>, ZPZV<5>,
                   ZPZV<16>, ZPZV<7>, ZPZV<2»; }; // NOLINT</pre>
02847 template<> struct ConwayPolynomial<19, 15> { using ZPZ = aerobus::zpz<19>; using type = POLYV<ZPZV<1>,
ZPZV<0>, ZPZV<0>, ZPZV<0>, ZPZV<0>, ZPZV<0>, ZPZV<0>, ZPZV<0>, ZPZV<0>, ZPZV<1>, ZPZV<1>, ZPZV<11>, ZPZV<11>, ZPZV<13>, ZPZV<15>, ZPZV<14>, ZPZV<14>, ZPZV<17»; }; // NOLINT

02848 template<> struct ConwayPolynomial<19, 16> { using ZPZ = aerobus::zpz<19>; using type = POLYV<ZPZV<1>,
                   ZPZV<0>, ZPZV<12>, ZPZV<13>, ZPZV<0>, ZPZV<14>, ZPZV<15>, ZPZV<14>, ZPZV<14>, ZPZV<2»; }; // NOLINT
02849 template<> struct ConwayPolynomial<19, 17> { using ZPZ = aerobus::zpz<19>; using type = POLYV<ZPZV<1>,
                  ZPZV<0>, ZPZV<0 , ZPZV<0 
02850 template<> struct ConwayPolynomial<19, 18> { using ZPZ = aerobus::zpz<19>; using type = POLYV<ZPZV<1>,
                   ZPZV<0>, ZPZV<0>, ZPZV<0>, ZPZV<0>, ZPZV<0>, ZPZV<0>, ZPZV<0>, ZPZV<10>, ZPZV<9>, ZPZV<7>, ZPZV<7>, ZPZV<17>, ZPZV<5>,
                    \texttt{ZPZV} < 0>, \ \texttt{ZPZV} < 16>, \ \texttt{ZPZV} < 5>, \ \texttt{ZPZV} < 7>, \ \texttt{ZPZV} < 3>, \ \texttt{ZPZV} < 14>, \ \texttt{ZPZV} < 2»; \ \}; \ \ // \ \texttt{NOLINT} 
02851 template<> struct ConwayPolynomial<19, 19> { using ZPZ = aerobus::zpz<19>; using type = POLYV<ZPZV<1>,
                  ZPZV<0>, ZPZV<0>
02852 template<> struct ConwayPolynomial<19, 20> { using ZPZ = aerobus::zpz<19>; using type = POLYV<ZPZV<1>,
                   ZPZV<0>, ZPZV<16>, ZPZV<13>, ZPZV<0>,
                   ZPZV<4>, ZPZV<7>, ZPZV<8>, ZPZV<6>, ZPZV<0>, ZPZV<3>, ZPZV<6>, ZPZV<11>, ZPZV<2»; }; // NOLINT</pre>
 02853 template<> struct ConwayPolynomial<23, 1> { using ZPZ = aerobus::zpz<23>; using type = POLYV<ZPZV<1>,
                   ZPZV<18»; }; // NOLINT</pre>
 02854 template<> struct ConwayPolynomial<23, 2> { using ZPZ = aerobus::zpz<23>; using type = POLYV<ZPZV<1>,
                   ZPZV<21>, ZPZV<5»; }; // NOLINT</pre>
 02855 template<> struct ConwayPolynomial<23, 3> { using ZPZ = aerobus::zpz<23>; using type = POLYV<ZPZV<1>,
                   ZPZV<0>, ZPZV<2>, ZPZV<18»; }; // NOLINT</pre>
02856 template<> struct ConwayPolynomial<23, 4> { using ZPZ = aerobus::zpz<23>; using type = POLYV<ZPZV<1>, ZPZV<0>, ZPZV<3>, ZPZV<19>, ZPZV<5»; }; // NOLINT
02857 template<> struct ConwayPolynomial<23, 5> { using ZPZ = aerobus::zpz<23>; using type = POLYV<ZPZV<1>,
                   ZPZV<0>, ZPZV<0>, ZPZV<0>, ZPZV<3>, ZPZV<18»; }; // NOLINT</pre>
02858 template<> struct ConwayPolynomial<23, 6> { using ZPZ = aerobus::zpz<23>; using type = POLYV<ZPZV<1>,
                   ZPZV<0>, ZPZV<1>, ZPZV<9>, ZPZV<9>, ZPZV<1>, ZPZV<5»; }; // NOLINT</pre>
 02859 template<> struct ConwayPolynomial<23, 7> { using ZPZ = aerobus::zpz<23>; using type = POLYV<ZPZV<1>,
```

```
02861 template<> struct ConwayPolynomial<23, 9> { using ZPZ = aerobus::zpz<23>; using type = POLYV<ZPZV<1>,
ZPZV<0>, ZPZV<0>, ZPZV<0>, ZPZV<0>, ZPZV<0>, ZPZV<0>, ZPZV<3>, ZPZV<3, ZPZV<3, ZPZV<3>, ZPZV<3, ZPZV<3, ZPZV<3, ZPZV<3, ZPZV<3, Z
                               ZPZV<0>, ZPZV<0>, ZPZV<0>, ZPZV<0>, ZPZV<1>, ZPZV<15>, ZPZV<15>, ZPZV<6>, ZPZV<1>, ZPZV<5»; }; //</pre>
                               NOLINT
02863 template<> struct ConwayPolynomial<23, 11> { using ZPZ = aerobus::zpz<23>; using type = POLYV<ZPZV<1>,
                               ZPZV<0>, ZPZV<0>, ZPZV<0>, ZPZV<0>, ZPZV<0>, ZPZV<0>, ZPZV<0>, ZPZV<0>, ZPZV<0>, ZPZV<2>, ZPZV<2>, ZPZV<18»;</pre>
                                 }; // NOLINT
 02864 template<> struct ConwayPolynomial<23, 12> { using ZPZ = aerobus::zpz<23>; using type = POLYV<ZPZV<1>,
                                \texttt{ZPZV} < \texttt{0} >, \ \texttt{ZPZV} < \texttt{21} >, \ \texttt{ZPZV} < \texttt{21} >, \ \texttt{ZPZV} < \texttt{15} >, \ \texttt{ZPZV} < \texttt{14} >, \ \texttt{ZPZV} < \texttt{12} >, \ \texttt{ZPZV} < \texttt{18} >, \ \texttt{ZPZV} < \texttt{28} >, \ \texttt{ZPZV} < 
                               ZPZV<12>, ZPZV<5»; }; // NOLINT</pre>
02865 template<> struct ConwayPolynomial<23, 13> { using ZPZ = aerobus::zpz<23>; using type = POLYV<ZPZV<1>,
                               ZPZV<0>, ZPZV<0>
                                ZPZV<9>, ZPZV<18»; }; // NOLINT</pre>
 02866 template<> struct ConwayPolynomial<23, 14> { using ZPZ = aerobus::zpz<23>; using type = POLYV<ZPZV<1>,
                              ZPZV<0>, ZPZV<0>, ZPZV<0>, ZPZV<0>, ZPZV<0>, ZPZV<1>, ZPZV<1>, ZPZV<16>, ZPZV<16>, ZPZV<18>, ZPZV<19>,
ZPZV<1>, ZPZV<22>, ZPZV<5»; }; // NOLINT</pre>
02867 template<> struct ConwayPolynomial<23, 15> { using ZPZ = aerobus::zpz<23>; using type = POLYV<ZPZV<1>,
                                ZPZV<0>, ZPZV<0>, ZPZV<0>, ZPZV<0>, ZPZV<0>, ZPZV<0>, ZPZV<0>, ZPZV<0>, ZPZV<0>, ZPZV<2>, ZPZV<2>, ZPZV<3</pre>
                                ZPZV<9>, ZPZV<7>, ZPZV<18>, ZPZV<18»; }; // NOLINT</pre>
 02868 template<> struct ConwayPolynomial<23, 16> { using ZPZ = aerobus::zpz<23>; using type = POLYV<ZPZV<1>,
                               ZPZV<0>, ZPZV<1>, ZPZV<19>, ZPZV<19>, ZPZV<16>,
ZPZV<13>, ZPZV<1>, ZPZV<14>, ZPZV<14>, ZPZV<15,; }; // NOLINT

02869 template<> struct ConwayPolynomial<23, 17> { using ZPZ = aerobus::zpz<23>; using type = POLYV<ZPZV<1>, ZPZV<0>, ZP
                                \texttt{ZPZV} < \texttt{0>, ZPZV} < \texttt{1>, ZPZV} < \texttt{1>, ZPZV} < \texttt{2>, ZPZV} < \texttt{2>, ZPZV} < \texttt{1>, ZPZV} < \texttt{1>, ZPZV} < \texttt{3>, ZPZV} < \texttt{3
                               02871 template<> struct ConwayPolynomial<23, 19> { using ZPZ = aerobus::zpz<23>; using type = POLYV<ZPZV<1>,
                              ZPZV<0>, ZPZ
 02872 template<> struct ConwayPolynomial<29, 1> { using ZPZ = aerobus::zpz<29>; using type = POLYV<ZPZV<1>,
                               ZPZV<27»; }; // NOLINT</pre>
 02873 template<> struct ConwayPolynomial<29, 2> { using ZPZ = aerobus::zpz<29>; using type = POLYV<ZPZV<1>,
                               ZPZV<24>, ZPZV<2»; }; // NOLINT</pre>
02874 template<> struct ConwayPolynomial<29, 3> { using ZPZ = aerobus::zpz<29>; using type = POLYV<ZPZV<1>,
                              ZPZV<0>, ZPZV<2>, ZPZV<27»; }; // NOLINT</pre>
 02875 template<> struct ConwayPolynomial<29, 4> { using ZPZ = aerobus::zpz<29>; using type = POLYV<ZPZV<1>,
                                ZPZV<0>, ZPZV<2>, ZPZV<15>, ZPZV<2»; };</pre>
                                                                                                                                                                                                                                        // NOLINT
 02876 template<> struct ConwayPolynomial<29, 5> { using ZPZ = aerobus::zpz<29>; using type = POLYV<ZPZV<1>,
ZPZV<0>, ZPZV<0>, ZPZV<0>, ZPZV<3>, ZPZV<27»; }; // NOLINT

02877 template<> struct ConwayPolynomial<29, 6> { using ZPZ = aerobus::zpz<29>; using type = POLYV<ZPZV<1>, ZPZV<0>, ZPZV<1>, ZPZV<25>, ZPZV<17>, ZPZV<13>, ZPZV<2»; }; // NOLINT
 02878 template<> struct ConwayPolynomial<29, 7> { using ZPZ = aerobus::zpz<29>; using type = POLYV<ZPZV<1>,
                               ZPZV<0>, ZPZV<0>, ZPZV<0>, ZPZV<0>, ZPZV<2>, ZPZV<2>, ZPZV<27»; }; // NOLINT</pre>
 02879 template<> struct ConwayPolynomial<29, 8> { using ZPZ = aerobus::zpz<29>; using type = POLYV<ZPZV<1>,
                              ZPZV<0>, ZPZV<0>, ZPZV<0>, ZPZV<3>, ZPZV<24>, ZPZV<26>, ZPZV<23>, ZPZV<2»; }; // NOLINT</pre>
02880 template<> struct ConwayPolynomial<29, 9> { using ZPZ = aerobus::zpz<29>; using type = POLYV<ZPZV<1>,
ZPZV<0>, ZPZV<0>, ZPZV<0>, ZPZV<0>, ZPZV<0>, ZPZV<0>, ZPZV<4>, ZPZV<4>, ZPZV<2>, ZPZV<2>, ZPZV<2>, ZPZV<27»; ); // NOLINT

02881 template<> struct ConwayPolynomial<29, 10> { using ZPZ = aerobus::zpz<29>; using type = POLYV<ZPZV<1>,
                                ZPZV<0>, ZPZV<0>, ZPZV<0>, ZPZV<1>, ZPZV<25>, ZPZV<8>, ZPZV<17>, ZPZV<2>, ZPZV<2>, ZPZV<2>, ZPZV<2>; }; //
                               NOLINT
02882 template<> struct ConwayPolynomial<29, 11> { using ZPZ = aerobus::zpz<29>; using type = POLYV<ZPZV<1>,
                               ZPZV<0>, ZPZV<0>, ZPZV<0>, ZPZV<0>, ZPZV<0>, ZPZV<0>, ZPZV<0>, ZPZV<0>, ZPZV<2>, ZPZV<2>, ZPZV<20>, ZPZV<2
                                }; // NOLINT
 02883 template<> struct ConwayPolynomial<29, 12> { using ZPZ = aerobus::zpz<29>; using type = POLYV<ZPZV<1>,
                                ZPZV<0>, ZPZV<0>, ZPZV<0>, ZPZV<3>, ZPZV<19>, ZPZV<28>, ZPZV<9>, ZPZV<16>, ZPZV<25>, ZPZV<1>, ZPZV<1>,
                               ZPZV<2»; }; // NOLINT</pre>
02884 template<> struct ConwayPolynomial<29, 13> { using ZPZ = aerobus::zpz<29>; using type = POLYV<ZPZV<1>,
                              ZPZV<0>, ZPZ
02885 template<> struct ConwayPolynomial<29, 14> { using ZPZ = aerobus::zpz<29>; using type = POLYV<ZPZV<1>,
                                ZPZV<0>, ZPZV<0>, ZPZV<0>, ZPZV<0>, ZPZV<0>, ZPZV<0>, ZPZV<1>, ZPZV<1>, ZPZV<3>, ZPZV<14>, ZPZV<10>, ZPZV<10>, ZPZV<21>, ZPZV<18>,
                               ZPZV<27>, ZPZV<5>, ZPZV<2»; }; // NOLINT</pre>
02886 template<> struct ConwayPolynomial<29, 15> { using ZPZ = aerobus::zpz<29>; using type = POLYV<ZPZV<1>,
ZPZV<0>, ZPZV<0>, ZPZV<0>, ZPZV<0>, ZPZV<0>, ZPZV<0>, ZPZV<0>, ZPZV<0>, ZPZV<0>, ZPZV<1>, ZPZV<13>, ZPZV<14>, ZPZV<14>, ZPZV<14>, ZPZV<15, ZPZV<12>, ZPZV<12>, ZPZV<16>, ZPZV<26>, ZPZV<27»; }; // NOLINT</pre>
02887 template<>> struct ConwayPolynomial<29, 16> { using ZPZ = aerobus::zpz<29>; using type = POLYV<ZPZV<1>,
                                ZPZV<0>, ZPZV<0>, ZPZV<0>, ZPZV<0>, ZPZV<0>, ZPZV<0>, ZPZV<0>, ZPZV<0>, ZPZV<6>, ZPZV<6>, ZPZV<6>, ZPZV<2>, ZPZV<18>,
                                ZPZV<23>, ZPZV<1>, ZPZV<27>, ZPZV<10>, ZPZV<2»; }; // NOLINT</pre>
 02888 template<> struct ConwayPolynomial<29, 17> { using ZPZ = aerobus::zpz<29>; using type = POLYV<ZPZV<1>,
ZPZV<0>, ZPZV<0>
                                ZPZV<0>, ZPZV<0>, ZPZV<0>, ZPZV<0>, ZPZV<0>, ZPZV<0>, ZPZV<24>, ZPZV<1>, ZPZV<1>, ZPZV<6>, ZPZV<26>,
ZPZV<2>, ZPZV<10>, ZPZV<8>, ZPZV<16>, ZPZV<19>, ZPZV<14>, ZPZV<2»; ); // NOLINT

02890 template<> struct ConwayPolynomial<29, 19> { using ZPZ = aerobus::zpz<29>; using type = POLYV<ZPZV<1>,
                               ZPZV<0>, ZPZV<0>
ZPZV<0>, ZPZV<0>, ZPZV<0>, ZPZV<0>, ZPZV<0>, ZPZV<0>, ZPZV<0>, ZPZV<0>, ZPZV<0>, ZPZV<2>, ZPZV<2>, ZPZV<2>, ZPZV<3>, ZPZV<3 , ZPZ
                               ZPZV<28»; }; // NOLINT
 02892 template<> struct ConwayPolynomial<31, 2> { using ZPZ = aerobus::zpz<31>; using type = POLYV<ZPZV<1>,
                               ZPZV<29>, ZPZV<3»; }; // NOLINT</pre>
 02893 template<> struct ConwayPolynomial<31, 3> { using ZPZ = aerobus::zpz<31>; using type = POLYV<ZPZV<1>,
                              ZPZV<0>, ZPZV<1>, ZPZV<28»; }; // NOLINT</pre>
 02894 template<> struct ConwayPolynomial<31, 4> { using ZPZ = aerobus::zpz<31>; using type = POLYV<ZPZV<1>,
```

```
// NOLINT
                         ZPZV<0>, ZPZV<3>, ZPZV<16>, ZPZV<3»; };</pre>
 02895 template<> struct ConwayPolynomial<31, 5> { using ZPZ = aerobus::zpz<31>; using type = POLYV<ZPZV<1>,
                        ZPZV<0>, ZPZV<0>, ZPZV<0>, ZPZV<7>, ZPZV<28»; }; // NOLINT</pre>
 02896 template<> struct ConwayPolynomial<31, 6> { using ZPZ = aerobus::zpz<31>; using type = POLYV<ZPZV<1>,
ZPZV<0>, ZPZV<0>, ZPZV<19>, ZPZV<16>, ZPZV<8>, ZPZV<3»; }; // NOLINT
02897 template<> struct ConwayPolynomial<31, 7> { using ZPZ = aerobus::zpz<31>; using type = POLYV<ZPZV<1>,
                        ZPZV<0>, ZPZV<0>, ZPZV<0>, ZPZV<0>, ZPZV<0>, ZPZV<1>, ZPZV<28»; }; // NOLINT
 02898 template<> struct ConwayPolynomial<31, 8> { using ZPZ = aerobus::zpz<31>; using type = POLYV<ZPZV<1>,
ZPZV<0>, ZPZV<0>, ZPZV<0>, ZPZV<0>, ZPZV<30>, ZPZV<26>, ZPZV<13>, ZPZV<13>, ZPZV<13>, ZPZV<13>, ZPZV<3»; }; //</pre>
 02901 template<> struct ConwayPolynomial<31, 11> { using ZPZ = aerobus::zpz<31>; using type = POLYV<ZPZV<1>,
                         ZPZV<0>, ZPZV<0>
                          l: // NOT.TNT
02902 template<> struct ConwayPolynomial<31, 12> { using ZPZ = aerobus::zpz<31>; using type = POLYV<ZPZV<1>,
                          ZPZV<0>, ZPZV<0>, ZPZV<0>, ZPZV<0>, ZPZV<4>, ZPZV<14>, ZPZV<28>, ZPZV<2>, ZPZV<2>, ZPZV<9>, ZPZV<25>, ZPZV<12>,
                         ZPZV<3»; }; // NOLINT</pre>
 02903 template<> struct ConwayPolynomial<31, 13> { using ZPZ = aerobus::zpz<31>; using type = POLYV<ZPZV<1>,
                         ZPZV<0>, ZPZ
ZPZV<6>, ZPZV<28»; }; // NOLINT

02904 template<> struct ConwayPolynomial<31, 14> { using ZPZ = aerobus::zpz<31>; using type = POLYV<ZPZV<1>, ZPZV<0>, ZPZV<0>, ZPZV<0>, ZPZV<0>, ZPZV<0>, ZPZV<1>, ZPZV<1
                          ZPZV<18>, ZPZV<6>, ZPZV<3»; }; // NOLINT</pre>
 02905 template<> struct ConwayPolynomial<31, 15> { using ZPZ = aerobus::zpz<31>; using type = POLYV<ZPZV<1>,
                         ZPZV<0>, ZPZV<0>, ZPZV<0>, ZPZV<0>, ZPZV<0>, ZPZV<0>, ZPZV<0>, ZPZV<0>, ZPZV<0>, ZPZV<30>, ZPZV<30>, ZPZV<30>, ZPZV<29>, ZPZV<12>,
ZPZV<13>, ZPZV<23>, ZPZV<25>, ZPZV<28»; }; // NOLINT
02906 template<> struct ConwayPolynomial<31, 16> { using ZPZ = aerobus::zpz<31>; using type = POLYV<ZPZV<1>,
                        ZPZV<0>, ZPZV<0>, ZPZV<0>, ZPZV<0>, ZPZV<2>, ZPZV<26>, ZPZV<26>, ZPZV<26>, ZPZV<26>, ZPZV<26>, ZPZV<28>, ZPZV<28 , Z
 02907 template<> struct ConwayPolynomial<31, 17> { using ZPZ = aerobus::zpz<31>; using type = POLYV<ZPZV<1>,
                         ZPZV<0>, ZPZV<0>
ZPZV<0>, ZPZV<0>, ZPZV<0>, ZPZV<0>, ZPZV<0>, ZPZV<2>, ZPZV<2>, ZPZV<28»; }; // NOLINT

02908 template<> struct ConwayPolynomial<31, 18> { using ZPZ = aerobus::zpz<31>; using type = POLYV<ZPZV<1>,
                         ZPZV<0>, ZPZV<0>, ZPZV<0>, ZPZV<0>, ZPZV<0>, ZPZV<1>, ZPZV<27>, ZPZV<5>, ZPZV<24>, ZPZV<24>, ZPZV<2>, ZPZV<7>,
ZPZV<12>, ZPZV<11>, ZPZV<25>, ZPZV<25>, ZPZV<10>, ZPZV<6>, ZPZV<33; }; // NOLINT 02909 template<> struct ConwayPolynomial<31, 19> { using ZPZ = aerobus::zpz<31>; using type
ZPZV<0>, ZPZ
                        ZPZV<35»; }; // NOLINT
02911 template<> struct ConwayPolynomial<37, 2> { using ZPZ = aerobus::zpz<37>; using type = POLYV<ZPZV<1>,
                         ZPZV<33>, ZPZV<2»; }; // NOLINT
 02912 template<> struct ConwayPolynomial<37, 3> { using ZPZ = aerobus::zpz<37>; using type = POLYV<ZPZV<1>,
                        ZPZV<0>, ZPZV<6>, ZPZV<35»; }; // NOLINT</pre>
 02913 template<> struct ConwayPolynomial<37, 4> { using ZPZ = aerobus::zpz<37>; using type = POLYV<ZPZV<1>,
ZPZV<0>, ZPZV<6>, ZPZV<24>, ZPZV<2*, }; // NOLINT

02914 template<> struct ConwayPolynomial<37, 5> { using ZPZ = aerobus::zpz<37>; using type = POLYV<ZPZV<1>,
                         ZPZV<0>, ZPZV<0>, ZPZV<0>, ZPZV<10>, ZPZV<35»; }; // NOLINT</pre>
 02915 template<> struct ConwayPolynomial<37, 6> { using ZPZ = aerobus::zpz<37>; using type = POLYV<ZPZV<1>,
                         ZPZV<0>, ZPZV<0>, ZPZV<35>, ZPZV<4>, ZPZV<30>, ZPZV<2»; }; // NOLINT
02916 template<> struct ConwayPolynomial<37, 7> { using ZPZ = aerobus::zpz<37>; using type = POLYV<ZPZV<1>, ZPZV<0>, ZPZV<0>, ZPZV<0>, ZPZV<0>, ZPZV<0>, ZPZV<7>, ZPZV<35»; }; // NOLINT
 02917 template<> struct ConwayPolynomial<37, 8> { using ZPZ = aerobus::zpz<37>; using type = POLYV<ZPZV<1>,
                         ZPZV<0>, ZPZV<0>, ZPZV<0>, ZPZV<7>, ZPZV<20>, ZPZV<27>, ZPZV<1>, ZPZV<2»; }; // NOLINT</pre>
 02918 template<> struct ConwayPolynomial<37, 9> { using ZPZ = aerobus::zpz<37>; using type = POLYV<ZPZV<1>,
ZPZV<0>, ZPZV<0>, ZPZV<0>, ZPZV<0>, ZPZV<0>, ZPZV<6>, ZPZV<6>, ZPZV<2>, ZPZV<32>, ZPZV<32>, ZPZV<35»; }; // NOLINT 02919 template<> struct ConwayPolynomial<37, 10> { using ZPZ = aerobus::zpz<37>; using type = POLYV<ZPZV<1>,
                         ZPZV<0>, ZPZV<0>, ZPZV<0>, ZPZV<0>, ZPZV<0>, ZPZV<2>, ZPZV<18>, ZPZV<111, ZPZV<4>, ZPZV<2»; }; //</pre>
                        NOLINT
02920 template<> struct ConwayPolynomial<37, 11> { using ZPZ = aerobus::zpz<37>; using type = POLYV<ZPZV<1>,
                         ZPZV<0>, ZPZV<2>, ZPZV<35»; };</pre>
                          // NOLINT
02921 template<> struct ConwayPolynomial<37, 12> { using ZPZ = aerobus::zpz<37>; using type = POLYV<ZPZV<1>,
                         ZPZV<0>, ZPZV<0>, ZPZV<0>, ZPZV<0>, ZPZV<4>, ZPZV<31>, ZPZV<10>, ZPZV<23>, ZPZV<23>, ZPZV<28, ZPZV<10>, ZPZV<20
                         ZPZV<33>, ZPZV<2»; }; // NOLINT
02922 template<> struct ConwayPolynomial<37, 13> { using ZPZ = aerobus::zpz<37>; using type = POLYV<ZPZV<1>,
                          ZPZV<0>, ZPZV<0</pre>, ZPZV<0>, ZP
                          ZPZV<6>, ZPZV<35»; }; // NOLINT</pre>
02923 template<> struct ConwayPolynomial<37, 14> { using ZPZ = aerobus::zpz<37>; using type = POLYV<ZPZV<1>,
                        ZPZV<0>, ZPZV<0>, ZPZV<0>, ZPZV<0>, ZPZV<0>, ZPZV<4>, ZPZV<35>, ZPZV<35>, ZPZV<1>, ZPZV<32>, ZPZV<16>,
ZPZV<1>, ZPZV<29>, ZPZV<29; }; // NOLINT</pre>
02924 template<> struct ConwayPolynomial<37, 15> { using ZPZ = aerobus::zpz<37>; using type = POLYV<ZPZV<1>,
                          ZPZV<0>, ZPZV<0>, ZPZV<0>, ZPZV<0>, ZPZV<0>, ZPZV<0>, ZPZV<0>, ZPZV<2>, ZPZV<1>, ZPZV<31>, ZPZV<28>, ZPZV<27>,
ZPZV<13>, ZPZV<34>, ZPZV<33>, ZPZV<35»; }; // NOLINT

02925 template<> struct ConwayPolynomial<37, 17> { using ZPZ = aerobus::zpz<37>; using type = POLYV<ZPZV<1>,
                         ZPZV<0>, ZPZV<0>
ZPZV<0>, ZPZV<0>, ZPZV<0>, ZPZV<0>, ZPZV<3>, ZPZV<3>, ZPZV<35»; }; // NOLINT

02926 template<> struct ConwayPolynomial<37, 18> { using ZPZ = aerobus::zpz<37>; using type = POLYV<ZPZV<1>,
 ZPZV<0>, ZPZ
```

```
ZPZV<35»; }; // NOLINT</pre>
 02929 template<> struct ConwayPolynomial<41, 2> { using ZPZ = aerobus::zpz<41>; using type = POLYV<ZPZV<1>,
                      ZPZV<38>, ZPZV<6»; }; // NOLINT</pre>
 02930 template<> struct ConwayPolynomial<41, 3> { using ZPZ = aerobus::zpz<41>; using type = POLYV<ZPZV<1>,
                      ZPZV<0>, ZPZV<1>, ZPZV<35»; }; // NOLINT</pre>
02931 template<> struct ConwayPolynomial<41, 4> { using ZPZ = aerobus::zpz<41>; using type = POLYV<ZPZV<1>,
                      ZPZV<0>, ZPZV<0>, ZPZV<23>, ZPZV<6»; }; // NOLINT</pre>
 02932 template<> struct ConwayPolynomial<41, 5> { using ZPZ = aerobus::zpz<41>; using type = POLYV<ZPZV<1>,
                      ZPZV<0>, ZPZV<0>, ZPZV<40>, ZPZV<14>, ZPZV<35»; }; // NOLINT</pre>
 02933 template<> struct ConwayPolynomial<41, 6> { using ZPZ = aerobus::zpz<41>; using type = POLYV<ZPZV<1>,
ZPZV<0>, ZPZV<4>, ZPZV<3>, ZPZV<6>, ZPZV<6>, ZPZV<6>, ZPZV<6>; }; // NOLINT

02934 template<> struct ConwayPolynomial<41, 7> { using ZPZ = aerobus::zpz<41>; using type = POLYV<ZPZV<1>,
                      ZPZV<0>, ZPZV<0>, ZPZV<0>, ZPZV<0>, ZPZV<0>, ZPZV<6>, ZPZV<6>, ZPZV<35»; }; // NOLINT
 02935 template<> struct ConwayPolynomial<41, 8> { using ZPZ = aerobus::zpz<41>; using type = POLYV<ZPZV<1>,
                      ZPZV<0>, ZPZV<0>, ZPZV<0>, ZPZV<5>, ZPZV<32>, ZPZV<20>, ZPZV<6>, ZPZV<6»; }; // NOLINT</pre>
02936 template<> struct ConwayPolynomial<41, 9> { using ZPZ = aerobus::zpz<41>; using type = POLYV<ZPZV<1>, ZPZV<0>, ZPZV<0>, ZPZV<0>, ZPZV<0>, ZPZV<0>, ZPZV<0>, ZPZV<4>, ZPZV<31>, ZPZV<5>, ZPZV<35»; }; // NOLINT 02937 template<> struct ConwayPolynomial<41, 10> { using ZPZ = aerobus::zpz<41>; using type = POLYV<ZPZV<1>,
                       ZPZV<0>, ZPZV<0>, ZPZV<0>, ZPZV<0>, ZPZV<3>, ZPZV<3>, ZPZV<31>, ZPZV<8>, ZPZV<20>, ZPZV<30>, ZPZV<6»; }; //
02938 template<> struct ConwayPolynomial<41, 11> { using ZPZ = aerobus::zpz<41>; using type = POLYV<ZPZV<1>, ZPZV<0>, ZP
                      }; // NOLINT
02939 template<> struct ConwayPolynomial<41, 12> { using ZPZ = aerobus::zpz<41>; using type = POLYV<ZPZV<1>, ZPZV<0>, ZPZV<0>, ZPZV<0>, ZPZV<0>, ZPZV<0>, ZPZV<26>, ZPZV<34>, ZPZV<34>, ZPZV<24>, ZPZV<21>,
                       ZPZV<27>, ZPZV<6»; }; // NOLINT</pre>
 02940 template<> struct ConwayPolynomial<41, 13> { using ZPZ = aerobus::zpz<41>; using type = POLYV<ZPZV<1>,
                      ZPZV<0>, ZPZV<0>
02941 template<> struct ConwayPolynomial<41, 14> { using ZPZ = aerobus::zpz<41>; using type = POLYV<ZPZV<1>,
                      ZPZV<0>, ZPZV<0>, ZPZV<0>, ZPZV<0>, ZPZV<0>, ZPZV<1>, ZPZV<15>, ZPZV<4>, ZPZV<27>, ZPZV<11>,
                       ZPZV<39>, ZPZV<10>, ZPZV<6»; }; // NOLINT</pre>
 02942 template<> struct ConwayPolynomial<41, 15> { using ZPZ = aerobus::zpz<41>; using type = POLYV<ZPZV<1>,
                       ZPZV<0>, ZPZV<0>, ZPZV<0>, ZPZV<0>, ZPZV<0>, ZPZV<0>, ZPZV<0>, ZPZV<0>, ZPZV<0>, ZPZV<2>,
ZPZV<35>, ZPZV<10>, ZPZV<21>, ZPZV<35»; }; // NOLINT
02943 template<> struct ConwayPolynomial<41, 17> { using ZPZ = aerobus::zpz<41>; using type = POLYV<ZPZV<1>,
                      ZPZV<0>, ZPZ
 02944 template<> struct ConwayPolynomial<41, 18> { using ZPZ = aerobus::zpz<41>; using type = POLYV<ZPZV<1>,
                      ZPZV<0>, ZPZV<0>, ZPZV<0>, ZPZV<0>, ZPZV<0>, ZPZV<0>, ZPZV<3>, ZPZV<3>, ZPZV<3>, ZPZV<3>, ZPZV<3>, ZPZV<3>, ZPZV<3>, ZPZV<3>, ZPZV<3</pre>
02945 template<> struct ConwayPolynomial<41, 19> { using ZPZ = aerobus::zpz<41>; using type = POLYV<ZPZV<1>, ZPZV<0>, ZPZV<0 , ZP
 02946 template<> struct ConwayPolynomial<43, 1> { using ZPZ = aerobus::zpz<43>; using type = POLYV<ZPZV<1>,
                      ZPZV<40»; }; // NOLINT</pre>
 02947 template<> struct ConwayPolynomial<43, 2> { using ZPZ = aerobus::zpz<43>; using type = POLYV<ZPZV<1>,
                      ZPZV<42>, ZPZV<3>; // NOLINT
 02948 template<> struct ConwayPolynomial<43, 3> { using ZPZ = aerobus::zpz<43>; using type = POLYV<ZPZV<1>,
                      ZPZV<0>, ZPZV<1>, ZPZV<40»; }; // NOLINT</pre>
02949 template<> struct ConwayPolynomial<43, 4> { using ZPZ = aerobus::zpz<43>; using type = POLYV<ZPZV<1>,
                      ZPZV<0>, ZPZV<5>, ZPZV<42>, ZPZV<3»; };
                                                                                                                                                                         // NOLINT
 02950 template<> struct ConwayPolynomial<43, 5> { using ZPZ = aerobus::zpz<43>; using type = POLYV<ZPZV<1>,
ZPZV<0>, ZPZV<0>, ZPZV<6>, ZPZV<8>, ZPZV<40»; }; // NOLINT
02951 template<> struct ConwayPolynomial<43, 6> { using ZPZ = aerobus::zpz<43>; using type = POLYV<ZPZV<1>,
ZPZV<0>, ZPZV<1>, ZPZV<28>, ZPZV<28>, ZPZV<28; }; // NOLINT
02952 template<> struct ConwayPolynomial<43, 7> { using ZPZ = aerobus::zpz<43>; using type = POLYV<ZPZV<1>,
                      ZPZV<0>, ZPZV<0>, ZPZV<0>, ZPZV<0>, ZPZV<42>, ZPZV<7>, ZPZV<40»; }; // NOLINT</pre>
 02953 template<> struct ConwayPolynomial<43, 8> { using ZPZ = aerobus::zpz<43>; using type = POLYV<ZPZV<1>,
                      ZPZV<0>, ZPZV<0>, ZPZV<0>, ZPZV<1>, ZPZV<39>, ZPZV<20>, ZPZV<24>, ZPZV<3»; };</pre>
                                                                                                                                                                                                                                                                                                                     // NOLINT
02954 template<> struct ConwayPolynomial<43, 9> { using ZPZ = aerobus::zpz<43>; using type = POLYV<ZPZV<1>, ZPZV<0>, ZPZV<0>, ZPZV<0>, ZPZV<0>, ZPZV<0>, ZPZV<0>, ZPZV<0>, ZPZV<0>, ZPZV<12>, ZPZV<39>, ZPZV<1>, ZPZV<40»; }; // NOLINT
02955 template<> struct ConwayPolynomial<43, 10> { using ZPZ = aerobus::zpz<43>; using type = POLYV<ZPZV<1>,
                       ZPZV<0>, ZPZV<0>, ZPZV<0>, ZPZV<2>, ZPZV<26>, ZPZV<36>, ZPZV<5>, ZPZV<27>, ZPZV<24>, ZPZV<24>, ZPZV<3»; }; //</pre>
                      NOT.TNT
02956 template<> struct ConwayPolynomial<43, 11> { using ZPZ = aerobus::zpz<43>; using type = POLYV<ZPZV<1>,
                      ZPZV<0>, ZPZV<0>
                       // NOLINT
02957 template<> struct ConwayPolynomial<43, 12> { using ZPZ = aerobus::zpz<43>; using type = POLYV<ZPZV<1>,
                       ZPZV<0>, ZPZV<0>, ZPZV<0>, ZPZV<0>, ZPZV<3>, ZPZV<27>, ZPZV<16>, ZPZV<17>, ZPZV<6>, ZPZV<23>,
                       ZPZV<38>, ZPZV<3»; }; // NOLINT</pre>
02958 template<> struct ConwayPolynomial<43, 13> { using ZPZ = aerobus::zpz<43>; using type = POLYV<ZPZV<1>,
                      ZPZV<0>, ZPZV<0>
02959 template<> struct ConwayPolynomial<43, 14> { using ZPZ = aerobus::zpz<43>; using type = POLYV<ZPZV<1>,
                       ZPZV<0>, ZPZV<0>, ZPZV<0>, ZPZV<0>, ZPZV<0>, ZPZV<0>, ZPZV<38>, ZPZV<22>, ZPZV<24>, ZPZV<37>,
                       ZPZV<18>, ZPZV<4>, ZPZV<19>, ZPZV<3»; };  // NOLINT</pre>
02960 template<> struct ConwayPolynomial<43, 15> { using ZPZ = aerobus::zpz<43>; using type = POLYV<ZPZV<1>,
ZPZV<0>, ZPZV<0>
ZPZV<0>, ZPZV<0>, ZPZV<0>, ZPZV<0>, ZPZV<0>, ZPZV<1>, ZPZV<3>, ZPZV<38, ZPZV<28, ZPZV<41, ZPZV<24, ZPZV<24, ZPZV<24, ZPZV<26, ZPZV<26, ZPZV<28, ZPZV<29, ZPZV<16>, ZPZV<34, ZPZV<37>, ZPZV<18>, ZPZV<3»; }; // NOLINT</pre>
02963 template<> struct ConwayPolynomial<43, 19> { using ZPZ = aerobus::zpz<43>; using type = POLYV<ZPZV<1>,
```

```
ZPZV<0>, ZPZV<0>
                          ZPZV<0>, ZPZV<0>, ZPZV<0>, ZPZV<0>, ZPZV<0>, ZPZV<0>, ZPZV<30>, ZPZV<40»; }; // NOLINT</pre>
02964 template<> struct ConwayPolynomial<47, 1> { using ZPZ = aerobus::zpz<47>; using type = POLYV<ZPZV<1>,
                         ZPZV<42»; }; // NOLINT
 02965 template<> struct ConwayPolynomial<47, 2> { using ZPZ = aerobus::zpz<47>; using type = POLYV<ZPZV<1>,
                         ZPZV<45>, ZPZV<5»; }; // NOLINT
 02966 template<> struct ConwayPolynomial<47, 3> { using ZPZ = aerobus::zpz<47>; using type = POLYV<ZPZV<1>,
                          ZPZV<0>, ZPZV<3>, ZPZV<42»; }; // NOLINT</pre>
02967 template<> struct ConwayPolynomial<47, 4> { using ZPZ = aerobus::zpz<47>; using type = POLYV<ZPZV<1>, ZPZV<0>, ZPZV<8>, ZPZV<40>, ZPZV<5»; }; // NOLINT
02968 template<> struct ConwayPolynomial<47, 5> { using ZPZ = aerobus::zpz<47>; using type = POLYV<ZPZV<1>,
                        ZPZV<0>, ZPZV<0>, ZPZV<0>, ZPZV<1>, ZPZV<42»: }: // NOLINT
02969 template<> struct ConwayPolynomial<47, 6> { using ZPZ = aerobus::zpz<47>; using type = POLYV<ZPZV<1>,
                         ZPZV<0>, ZPZV<2>, ZPZV<35>, ZPZV<9>, ZPZV<41>, ZPZV<5»; }; // NOLINT</pre>
 02970 template<> struct ConwayPolynomial<47, 7> { using ZPZ = aerobus::zpz<47>; using type = POLYV<ZPZV<1>,
ZPZV<0>, ZPZV<0>, ZPZV<0>, ZPZV<0>, ZPZV<0>, ZPZV<0>, ZPZV<0>, ZPZV<0>, ZPZV<1>, ZPZV<1>, ZPZV<1>, ZPZV<42»; }; // NOLINT

02971 template<> struct ConwayPolynomial<47, 8> { using ZPZ = aerobus::zpz<47>; using type = POLYV<ZPZV<1>, ZPZV<0>, ZPZV<0>, ZPZV<0>, ZPZV<0>, ZPZV<0>, ZPZV<29>, ZPZV<19>, ZPZV<3>, ZPZV<3>; }; // NOLINT

02972 template<> struct ConwayPolynomial<47, 9> { using ZPZ = aerobus::zpz<47>; using type = POLYV<ZPZV<1>,
                         ZPZV<0>, ZPZV<0>, ZPZV<0>, ZPZV<0>, ZPZV<0>, ZPZV<1>, ZPZV<19>, ZPZV<1>, ZPZV<42»; }; // NOLINI</pre>
 02973 template<> struct ConwayPolynomial<47, 10> { using ZPZ = aerobus::zpz<47>; using type = POLYV<ZPZV<1>, ZPZV<0>, ZPZV<0>, ZPZV<0>, ZPZV<0>, ZPZV<42>, ZPZV<14>, ZPZV<18>, ZPZV<45>, ZPZV<45>, ZPZV<45>, ZPZV<5»; }; //
                         NOLINT
02974 template<> struct ConwayPolynomial<47, 11> { using ZPZ = aerobus::zpz<47>; using type = POLYV<ZPZV<1>,
                         ZPZV<0>, ZPZV<42»; );
                          // NOLINT
 02975 template<> struct ConwayPolynomial<47, 12> { using ZPZ = aerobus::zpz<47>; using type = POLYV<ZPZV<1>,
                         ZPZV<0>, ZPZV<0>, ZPZV<0>, ZPZV<0>, ZPZV<46>, ZPZV<40>, ZPZV<45>, ZPZV<45>, ZPZV<45>, ZPZV<46>, ZPZV<46 , ZPZV<47 , ZPZV<46 , ZPZV<47 , 
                         ZPZV<9>, ZPZV<5»; }; // NOLINT</pre>
02976 template<> struct ConwayPolynomial<47, 13> { using ZPZ = aerobus::zpz<47>; using type = POLYV<ZPZV<1>,
                         ZPZV<0>, ZPZV<0>
                          ZPZV<5>, ZPZV<42»; }; // NOLINT</pre>
 02977 template<> struct ConwayPolynomial<47, 14> { using ZPZ = aerobus::zpz<47>; using type = POLYV<ZPZV<1>,
ZPZV<0>, ZPZV<0>, ZPZV<0>, ZPZV<0>, ZPZV<0>, ZPZV<0>, ZPZV<0>, ZPZV<20>, ZPZV<20>, ZPZV<30>, ZPZV<17>, ZPZV<24>, ZPZV<39>, ZPZV<32>, ZPZV<5»; }; // NOLINT

02978 template<> struct ConwayPolynomial<47, 15> { using ZPZ = aerobus::zpz<47>; using type = POLYV<ZPZV<1>,
                        ZPZV<0>, ZPZV<0>, ZPZV<0>, ZPZV<0>, ZPZV<0>, ZPZV<0>, ZPZV<0>, ZPZV<14>,
ZPZV<42>, ZPZV<13>, ZPZV<14>,
 02979 template<> struct ConwayPolynomial<47, 17> { using ZPZ = aerobus::zpz<47>; using type = POLYV<ZPZV<1>,
                        ZPZV<0>, ZPZV<0>
02980 template<> struct ConwayPolynomial<47, 18> { using ZPZ = aerobus::zpz<47>; using type = POLYV<ZPZV<1>,
                         ZPZV<0>, ZPZV<0>, ZPZV<0>, ZPZV<0>, ZPZV<0>, ZPZV<41>, ZPZV<42>, ZPZV<26>, ZPZV<44>,
ZPZV<24>, ZPZV<22>, ZPZV<11>, ZPZV<55, ZPZV<33>, ZPZV<5»; ); // NOLINT
02981 template<> struct ConwayPolynomial<47, 19> { using ZPZ = aerobus::zpz<47>; using type = POLYV<ZPZV<1>,
                          ZPZV<0>, ZPZ
                         02982 template<> struct ConwayPolynomial<53, 1> { using ZPZ = aerobus::zpz<53>; using type = POLYV<ZPZV<1>,
                         ZPZV<51»; }; // NOLINT</pre>
 02983 template<> struct ConwayPolynomial<53, 2> { using ZPZ = aerobus::zpz<53>; using type = POLYV<ZPZV<1>,
                         ZPZV<49>, ZPZV<2»; }; // NOLINT</pre>
 02984 template<> struct ConwayPolynomial<53, 3> { using ZPZ = aerobus::zpz<53>; using type = POLYV<ZPZV<1>,
                         ZPZV<0>, ZPZV<3>, ZPZV<51»; }; // NOLINT</pre>
02985 template<> struct ConwayPolynomial<53, 4> { using ZPZ = aerobus::zpz<53>; using type = POLYV<ZPZV<1>, ZPZV<0>, ZPZV<9>, ZPZV<38>, ZPZV<2»; }; // NOLINT
02986 template<> struct ConwayPolynomial<53, 5> { using ZPZ = aerobus::zpz<53>; using type = POLYV<ZPZV<1>,
                         ZPZV<0>, ZPZV<0>, ZPZV<0>, ZPZV<3>, ZPZV<51»; }; // NOLINT</pre>
 02987 template<> struct ConwayPolynomial<53, 6> { using ZPZ = aerobus::zpz<53>; using type = POLYV<ZPZV<1>,
                         ZPZV<0>, ZPZV<1>, ZPZV<7>, ZPZV<4>, ZPZV<45>, ZPZV<2»; }; // NOLINT</pre>
 02988 template<> struct ConwayPolynomial<53, 7> { using ZPZ = aerobus::zpz<53>; using type = POLYV<ZPZV<1>,
ZPZV<0>, ZPZV<0>, ZPZV<0>, ZPZV<0>, ZPZV<0>, ZPZV<0>, ZPZV<0>, ZPZV<0>, ZPZV<2>, ZPZV<5>; ;/ NOLINT

02989 template<> struct ConwayPolynomial<53, 8> { using ZPZ = aerobus::zpz<53>; using type = POLYV<ZPZV<1>,
                         ZPZV<0>, ZPZV<0>, ZPZV<0>, ZPZV<8>, ZPZV<29>, ZPZV<18>, ZPZV<1>, ZPZV<2»; }; // NOLINT
 02990 template<> struct ConwayPolynomial<53, 9> { using ZPZ = aerobus::zpz<53>; using type = POLYV<ZPZV<1>, ZPZV<0>, ZPZV<0>, ZPZV<0>, ZPZV<0>, ZPZV<0>, ZPZV<0>, ZPZV<51>; // NOLINT
 02991 template<> struct ConwayPolynomial<53, 10> { using ZPZ = aerobus::zpz<53>; using type = POLYV<ZPZV<1>,
                         ZPZV<0>, ZPZV<0>, ZPZV<0>, ZPZV<1>, ZPZV<1>, ZPZV<2>, ZPZV<25>, ZPZV<25>, ZPZV<29>, ZPZV<29>; //
                         NOLINT
02992 template<> struct ConwayPolynomial<53, 11> { using ZPZ = aerobus::zpz<53>; using type = POLYV<ZPZV<1>,
                          ZPZV<0>, ZPZV<0>, ZPZV<0>, ZPZV<0>, ZPZV<0>, ZPZV<0>, ZPZV<0>, ZPZV<0>, ZPZV<0>, ZPZV<51»;</pre>
                          }; // NOLINT
 02993 template<> struct ConwayPolynomial<53, 12> { using ZPZ = aerobus::zpz<53>; using type = POLYV<ZPZV<1>,
                          \texttt{ZPZV} < \texttt{0>, } \texttt{ZPZV} < \texttt{0>, } \texttt{ZPZV} < \texttt{0>, } \texttt{ZPZV} < \texttt{2>, } \texttt{ZPZV} < \texttt{34>, } \texttt{ZPZV} < \texttt{4>, } \texttt{ZPZV} < \texttt{13>, } \texttt{ZPZV} < \texttt{10>, } \texttt{ZPZV} < \texttt{42>, } \texttt{ZPZV} < \texttt{34>, } \texttt{2PZV} < \texttt{34>, } \texttt{2PZV} < \texttt{34>, } \texttt{
                         ZPZV<41>, ZPZV<2»; }; // NOLINT</pre>
02994 template<> struct ConwayPolynomial<53, 13> { using ZPZ = aerobus::zpz<53>; using type = POLYV<ZPZV<1>,
                          ZPZV<0>, ZPZV<0>
                          ZPZV<28>, ZPZV<51»; }; // NOLINT</pre>
02995 template<> struct ConwayPolynomial<53, 14> { using ZPZ = aerobus::zpz<53>; using type = POLYV<ZPZV<1>,
                        ZPZV<0>, ZPZV<0>, ZPZV<0>, ZPZV<0>, ZPZV<0>, ZPZV<1>, ZPZV<45>, ZPZV<23>, ZPZV<52>, ZPZV<52>, ZPZV<37>,
ZPZV<12>, ZPZV<23>, ZPZV<23>, ZPZV<2»; }; // NOLINT</pre>
02996 template<> struct ConwayPolynomial<53, 15> { using ZPZ = aerobus::zpz<53>; using type = POLYV<ZPZV<1>,
                          ZPZV<0>, ZPZV<0>, ZPZV<0>, ZPZV<0>, ZPZV<0>, ZPZV<0>, ZPZV<0>, ZPZV<1>>,
 ZPZV<11>, ZPZV<20>, ZPZV<4>, ZPZV<51»; }; // NOLINT

02997 template<> struct ConwayPolynomial<53, 17> { using ZPZ = aerobus::zpz<53>; using type = POLYV<ZPZV<1>,
ZPZV<0>, ZPZ
```

```
ZPZV<0>, ZPZV<0>, ZPZV<0>, ZPZV<0>, ZPZV<0>, ZPZV<0>, ZPZV<51>, ZPZV<51>, ZPZV<51>, ZPZV<27>, ZPZV<0>,
ZPZV<39>, ZPZV<44>, ZPZV<6>, ZPZV<8>, ZPZV<10>, ZPZV<11>, ZPZV<2»; }; // NOLINT

02999 template<> struct ConwayPolynomial<53, 19> { using ZPZ = aerobus::zpz<53>; using type = POLYV<ZPZV<1>,
                         ZPZV<0>, ZPZV<0>
03000 template<> struct ConwayPolynomial<59, 1> { using ZPZ = aerobus::zpz<59>; using type = POLYV<ZPZV<1>,
                          ZPZV<57»; }; // NOLINT
 03001 template<> struct ConwayPolynomial<59, 2> { using ZPZ = aerobus::zpz<59>; using type = POLYV<ZPZV<1>,
                          ZPZV<58>, ZPZV<2»; }; // NOLINT</pre>
 03002 template<> struct ConwayPolynomial<59, 3> { using ZPZ = aerobus::zpz<59>; using type = POLYV<ZPZV<1>,
                          ZPZV<0>, ZPZV<5>, ZPZV<57»; }; // NOLINT</pre>
03003 template<> struct ConwayPolynomial<59, 4> { using ZPZ = aerobus::zpz<59>; using type = POLYV<ZPZV<1>,
                                                                                                                                                                                                     // NOLINT
                          ZPZV<0>, ZPZV<2>, ZPZV<40>, ZPZV<2»; };</pre>
 03004 template<> struct ConwayPolynomial<59, 5> { using ZPZ = aerobus::zpz<59>; using type = POLYV<ZPZV<1>,
                           ZPZV<0>, ZPZV<0>, ZPZV<0>, ZPZV<8>, ZPZV<57»; }; // NOLINT</pre>
 03005 template<> struct ConwayPolynomial<59, 6> { using ZPZ = aerobus::zpz<59>; using type = POLYV<ZPZV<1>,
                         ZPZV<0>, ZPZV<2>, ZPZV<18>, ZPZV<38>, ZPZV<0>, ZPZV<2»; }; // NOLINT</pre>
03006 template<> struct ConwayPolynomial<59, 7> { using ZPZ = aerobus::zpz<59>; using type = POLYV<ZPZV<1>, ZPZV<0>, ZPZV<0 , ZPZ
03007 template<> struct ConwayPolynomial<59, 8> { using ZPZ = aerobus::zpz<59>; using type = POLYV<ZPZV<1>,
ZPZV<0>, ZPZV<0>, ZPZV<16>, ZPZV<16>, ZPZV<32>, ZPZV<50>, ZPZV<2»; }; // NOLINT
03008 template<> struct ConwayPolynomial<59, 9> { using ZPZ = aerobus::zpz<59>; using type = POLYV<ZPZV<1>,
NOLINT
 03010 template<> struct ConwayPolynomial<59, 11> { using ZPZ = aerobus::zpz<59>; using type = POLYV<ZPZV<1>,
                          ZPZV<0>, ZPZV<0>
                            // NOLINT
03011 template<> struct ConwayPolynomial<59, 12> { using ZPZ = aerobus::zpz<59>; using type = POLYV<ZPZV<1>,
                          ZPZV<0>, ZPZV<0>, ZPZV<0>, ZPZV<0>, ZPZV<3>, ZPZV<51>, ZPZV<51>, ZPZV<21>, ZPZV<38>, ZPZV<8>,
                           ZPZV<1>, ZPZV<2»; }; // NOLINT</pre>
 03012 template<> struct ConwayPolynomial<59, 13> { using ZPZ = aerobus::zpz<59>; using type = POLYV<ZPZV<1>,
                           ZPZV<0>, ZPZV<0>,
ZPZV<3>, ZPZV<57»; }; // NOLINT
03013 template<> struct ConwayPolynomial<59, 14> { using ZPZ = aerobus::zpz<59>; using type = POLYV<ZPZV<1>,
                         ZPZV<0>, ZPZV<0>, ZPZV<0>, ZPZV<0>, ZPZV<0>, ZPZV<1>, ZPZV<33>, ZPZV<51>, ZPZV<1>, ZPZV<1>, ZPZV<25>, ZPZV<26>, ZPZV<26 , ZPZV<27 , ZPZV
 03014 template<> struct ConwayPolynomial<59, 15> { using ZPZ = aerobus::zpz<59>; using type = POLYV<ZPZV<1>,
                           ZPZV<0>, ZPZV<0>, ZPZV<0>, ZPZV<0>, ZPZV<0>, ZPZV<0>, ZPZV<0>, ZPZV<0>, ZPZV<0>, ZPZV<57>, ZPZV<24>, ZPZV<23>,
ZPZV<13>, ZPZV<39>, ZPZV<58>, ZPZV<57»; }; // NOLINT
03015 template<> struct ConwayPolynomial<59, 17> { using ZPZ = aerobus::zpz<59>; using type = POLYV<ZPZV<1>,
ZPZV<0>, ZPZ
ZPZV<0>, ZPZV<0>, ZPZV<0>, ZPZV<0>, ZPZV<0>, ZPZV<5>, ZPZV×5>, ZPZ
                           ZPZV<0>, ZPZV<0>, ZPZV<0>, ZPZV<0>, ZPZV<0>, ZPZV<1>, ZPZV<37>, ZPZV<38>, ZPZV<27>, ZPZV<11>,
                          03017 template<> struct ConwayPolynomial<59, 19> { using ZPZ = aerobus::zpz<59>; using type = POLYV<ZPZV<1>,
                         ZPZV<0>, ZPZ
03018 template<> struct ConwayPolynomial<61, 1> { using ZPZ = aerobus::zpz<61>; using type = POLYV<ZPZV<1>,
                          ZPZV<59»; }; // NOLINT
 03019 template<> struct ConwayPolynomial<61, 2> { using ZPZ = aerobus::zpz<61>; using type = POLYV<ZPZV<1>,
ZPZV<60>, ZPZV<2»; }; // NOLINT
03020 template<> struct ConwayPolynomial<61, 3> { using ZPZ = aerobus::zpz<61>; using type = POLYV<ZPZV<1>,
                          ZPZV<0>, ZPZV<7>, ZPZV<59»; }; // NOLINT</pre>
 03021 template<> struct ConwayPolynomial<61, 4> { using ZPZ = aerobus::zpz<61>; using type = POLYV<ZPZV<1>,
                           ZPZV<0>, ZPZV<3>, ZPZV<40>, ZPZV<2»; }; // NOLINT</pre>
 03022 template<> struct ConwayPolynomial<61, 5> { using ZPZ = aerobus::zpz<61>; using type = POLYV<ZPZV<1>,
                          ZPZV<0>, ZPZV<0>, ZPZV<0>, ZPZV<12>, ZPZV<59»; };</pre>
                                                                                                                                                                                                                                                 // NOLINT
03023 template<> struct ConwayPolynomial<61, 6> { using ZPZ = aerobus::zpz<61>; using type = POLYV<ZPZV<1>,
                         ZPZV<0>, ZPZV<0>, ZPZV<49>, ZPZV<3>, ZPZV<29>, ZPZV<2»; }; // NOLINT
03024 template<> struct ConwayPolynomial<61, 7> { using ZPZ = aerobus::zpz<61>; using type = POLYV<ZPZV<1>,
                          ZPZV<0>, ZPZV<0>, ZPZV<0>, ZPZV<0>, ZPZV<0>, ZPZV<2>, ZPZV<59»; }; // NOLINT
 03025 template<> struct ConwayPolynomial<61, 8> { using ZPZ = aerobus::zpz<61>; using type = POLYV<ZPZV<1>,
                           \texttt{ZPZV} < 0>, \ \texttt{ZPZV} < 0>, \ \texttt{ZPZV} < 0>, \ \texttt{ZPZV} < 0>, \ \texttt{ZPZV} < 5>, \ \texttt{ZPZV} < 1>, \ \texttt{ZPZV} < 56>, \ \texttt{ZPZV} < 2»; \ \}; \ \ // \ \texttt{NOLINT} 
03026 template<> struct ConwayPolynomial<61, 9> { using ZPZ = aerobus::zpz<61>; using type = POLYV<ZPZV<1>, ZPZV<0>, ZPZV<0>, ZPZV<0>, ZPZV<0>, ZPZV<0>, ZPZV<0>, ZPZV<0>, ZPZV<50>, ZPZV<
                          ZPZV<0>, ZPZV<0>, ZPZV<0>, ZPZV<0>, ZPZV<28>, ZPZV<15>, ZPZV<44>, ZPZV<16>, ZPZV<6>, ZPZV<6>, ZPZV<2»; };</pre>
03028 template<> struct ConwayPolynomial<61, 11> { using ZPZ = aerobus::zpz<61>; using type = POLYV<ZPZV<1>,
                          ZPZV<0>, ZPZV<0>
                           }; // NOLINT
03029 template<> struct ConwayPolynomial<61, 12> { using ZPZ = aerobus::zpz<61>; using type = POLYV<ZPZV<1>,
                           ZPZV<0>, ZPZV<0>, ZPZV<0>, ZPZV<2>, ZPZV<42>, ZPZV<33>, ZPZV<8>, ZPZV<38>, ZPZV<14>, ZPZV<1>,
                           ZPZV<15>, ZPZV<2»; }; // NOLINT</pre>
03030 template<> struct ConwayPolynomial<61, 13> { using ZPZ = aerobus::zpz<61>; using type = POLYV<ZPZV<1>,
                          ZPZV<0>, ZPZ
                          ZPZV<3>, ZPZV<59»; }; // NOLINT</pre>
03031 template<> struct ConwayPolynomial<61, 14> { using ZPZ = aerobus::zpz<61>; using type = POLYV<ZPZV<1>,
                           ZPZV<0>, ZPZV<0>, ZPZV<0>, ZPZV<0>, ZPZV<0>, ZPZV<0>, ZPZV<1>, ZPZV<48>, ZPZV<26>, ZPZV<11>, ZPZV<8>, ZPZV<30>,
                           ZPZV<54>, ZPZV<48>, ZPZV<2»; };  // NOLINT</pre>
 03032 template<> struct ConwayPolynomial<61, 15> { using ZPZ = aerobus::zpz<61>; using type = POLYV<ZPZV<1>,
                          ZPZV<0>, ZPZV<0>, ZPZV<0>, ZPZV<0>, ZPZV<0>, ZPZV<0>, ZPZV<0>, ZPZV<0>, ZPZV<0>, ZPZV<3>, ZPZV<35>, ZPZV<44>,
ZPZV<25>, ZPZV<23>, ZPZV<51>, ZPZV<59»; }; // NOLINT
03033 template<>> struct ConwayPolynomial<61, 17> { using ZPZ = aerobus::zpz<61>; using type = POLYV<ZPZV<1>,
```

```
ZPZV<0>, ZPZV<0>
ZPZV<0>, ZPZV<0>, ZPZV<0>, ZPZV<0>, ZPZV<0>, ZPZV<0>, ZPZV<5>»; }; // NOLINT
03034 template<> struct ConwayPolynomial<61, 18> { using ZPZ = aerobus::zpz<61>; using type = POLYV<ZPZV<1>,
ZPZV<0>, ZPZV<0>, ZPZV<0>, ZPZV<0>, ZPZV<0>, ZPZV<3>, ZPZV<35>, ZPZV<36>, ZPZV<36>, ZPZV<36>, ZPZV<34>, ZPZV<35>, ZPZV<57>, ZPZV<42>, ZPZV<55>, ZPZV<52>, ZPZV<52>, ZPZV<2»; }; // NOLINT
03035 template<> struct ConwayPolynomial<61, 19> { using ZPZ = aerobus::zpz<61>; using type = POLYV<ZPZV<1>,
                        ZPZV<0>, ZPZ
03036 template<> struct ConwayPolynomial<67, 1> { using ZPZ = aerobus::zpz<67>; using type = POLYV<ZPZV<1>,
                        ZPZV<65»; }; // NOLINT</pre>
03037 template<> struct ConwayPolynomial<67, 2> { using ZPZ = aerobus::zpz<67>; using type = POLYV<ZPZV<1>,
                        ZPZV<63>, ZPZV<2»; }; // NOLINT
03038 template<> struct ConwayPolynomial<67, 3> { using ZPZ = aerobus::zpz<67>; using type = POLYV<ZPZV<1>,
                        ZPZV<0>, ZPZV<6>, ZPZV<65»; }; // NOLINT</pre>
 03039 template<> struct ConwayPolynomial<67, 4> { using ZPZ = aerobus::zpz<67>; using type = POLYV<ZPZV<1>,
ZPZV<0>, ZPZV<8>, ZPZV<54>, ZPZV<2»; }; // NOLINT
03040 template<> struct ConwayPolynomial<67, 5> { using ZPZ = aerobus::zpz<67>; using type = POLYV<ZPZV<1>,
ZPZV<0>, ZPZV<0>, ZPZV<0>, ZPZV<0>, ZPZV<65; }; // NOLINT
03041 template<> struct ConwayPolynomial<67, 6> { using ZPZ = aerobus::zpz<67>; using type = POLYV<ZPZV<1>,
                        ZPZV<0>, ZPZV<0>, ZPZV<63>, ZPZV<49>, ZPZV<55>, ZPZV<2»; }; // NOLINT</pre>
 03042 template<> struct ConwayPolynomial<67, 7> { using ZPZ = aerobus::zpz<67>; using type = POLYV<ZPZV<1>,
                       ZPZV<0>, ZPZV<0>, ZPZV<0>, ZPZV<0>, ZPZV<0>, ZPZV<7>, ZPZV<65»; }; // NOLINT</pre>
03043 template<> struct ConwayPolynomial<67, 8> { using ZPZ = aerobus::zpz<67>; using type = POLYV<ZPZV<1>,
ZPZV<0>, ZPZV<0>, ZPZV<0>, ZPZV<3>, ZPZV<46>, ZPZV<17>, ZPZV<64>, ZPZV<64>, ZPZV<2»; }; // NOLINT
03044 template<> struct ConwayPolynomial<67, 9> { using ZPZ = aerobus::zpz<67>; using type = POLYV<ZPZV<1>,
                        ZPZV<0>, ZPZV<0>, ZPZV<0>, ZPZV<0>, ZPZV<0>, ZPZV<2>, ZPZV<49>, ZPZV<55>, ZPZV<65»; }; // NOLINT</pre>
 03045 template<> struct ConwayPolynomial<67, 10> { using ZPZ = aerobus::zpz<67>; using type = POLYV<ZPZV<1>,
                        ZPZV<0>, ZPZV<0>, ZPZV<0>, ZPZV<2>, ZPZV<21>, ZPZV<0>, ZPZV<16>, ZPZV<7>, ZPZV<23>, ZPZV<2»; }; //</pre>
                        NOLINT
03046 template<> struct ConwayPolynomial<67, 11> { using ZPZ = aerobus::zpz<67>; using type = POLYV<ZPZV<1>,
                        ZPZV<0>, ZPZV<0>, ZPZV<0>, ZPZV<0>, ZPZV<0>, ZPZV<0>, ZPZV<0>, ZPZV<0>, ZPZV<66>, ZPZV<66>, ZPZV<65»;
                         }; // NOLINT
 03047 template<> struct ConwayPolynomial<67, 12> { using ZPZ = aerobus::zpz<67>; using type = POLYV<ZPZV<1>,
                        ZPZV<0>, ZPZV<0>, ZPZV<0>, ZPZV<3>, ZPZV<57>, ZPZV<27>, ZPZV<4>, ZPZV<55>, ZPZV<64>, ZPZV<64>, ZPZV<61>,
ZPZV<27>, ZPZV<2»; }; // NOLINT
03048 template<> struct ConwayPolynomial<67, 13> { using ZPZ = aerobus::zpz<67>; using type = POLYV<ZPZV<1>,
                       ZPZV<0>, ZPZ
 03049 template<> struct ConwayPolynomial<67, 14> { using ZPZ = aerobus::zpz<67>; using type = POLYV<ZPZV<1>,
                       ZPZV<0>, ZPZV<1>, ZPZV<22>, ZPZV<5>, ZPZV<5>, ZPZV<5>, ZPZV<5>, ZPZV<1>, ZPZV<37>, ZPZV<37>, ZPZV<2»; }; // NOLINT</pre>
03050 template<> struct ConwayPolynomial<67, 15> { using ZPZ = aerobus::zpz<67>; using type = POLYV<ZPZV<1>,
ZPZV<0>, ZPZV<0>, ZPZV<0>, ZPZV<0>, ZPZV<0>, ZPZV<0>, ZPZV<0>, ZPZV<0>, ZPZV<0>, ZPZV<1>, ZPZV<1>, ZPZV<1>, ZPZV<1>, ZPZV<41>, ZPZV<20>, ZPZV<21>, ZPZV<46>, ZPZV<65»; }; // NOLINT

03051 template<> struct ConwayPolynomial<67, 17> { using ZPZ = aerobus::zpz<67>; using type = POLYV<ZPZV<1>,
                         ZPZV<0>, ZPZ
                        ZPZV<0>, ZPZV<0>, ZPZV<0>, ZPZV<0>, ZPZV<5>, ZPZV<65»; }; // NOLINT
03052 template<> struct ConwayPolynomial<67, 18> { using ZPZ = aerobus::zpz<67>; using type = POLYV<ZPZV<1>, ZPZV<0>, ZPZV<0>, ZPZV<0>, ZPZV<0>, ZPZV<0>, ZPZV<5, ZPZV<5>, ZPZV<5-, ZPZ
ZPZV<55>, ZPZV<28>, ZPZV<29>, ZPZV<65), ZPZV<65), ZPZV<59>, ZPZV<59>, ZPZV<13>, ZPZV<28; ); // NOLINT
03053 template<> struct ConwayPolynomial<67, 19> { using ZPZ = aerobus::zpz<67>; using type = POLYV<ZPZV<1>,
                        ZPZV<0>, ZPZV<0>
03054 template<> struct ConwayPolynomial<71, 1> { using ZPZ = aerobus::zpz<71>; using type = POLYV<ZPZV<1>,
                       ZPZV<64»; }; // NOLINT
03055 template<> struct ConwayPolynomial<71, 2> { using ZPZ = aerobus::zpz<71>; using type = POLYV<ZPZV<1>,
                         ZPZV<69>, ZPZV<7»; };  // NOLINT</pre>
 03056 template<> struct ConwayPolynomial<71, 3> { using ZPZ = aerobus::zpz<71>; using type = POLYV<ZPZV<1>,
                        ZPZV<0>, ZPZV<4>, ZPZV<64»; }; // NOLINT</pre>
03057 template<> struct ConwayPolynomial<71, 4> { using ZPZ = aerobus::zpz<71>; using type = POLYV<ZPZV<1>, ZPZV<0>, ZPZV<4>, ZPZV<41>, ZPZV<7»; }; // NOLINT
03058 template<> struct ConwayPolynomial<71, 5> { using ZPZ = aerobus::zpz<71>; using type = POLYV<ZPZV<1>,
                        ZPZV<0>, ZPZV<0>, ZPZV<0>, ZPZV<18>, ZPZV<64»; }; // NOLINT</pre>
 03059 template<> struct ConwayPolynomial<71, 6> { using ZPZ = aerobus::zpz<71>; using type = POLYV<ZPZV<1>,
                        ZPZV<0>, ZPZV<1>, ZPZV<10>, ZPZV<13>, ZPZV<29>, ZPZV<7»; }; // NOLINT</pre>
 03060 template<> struct ConwayPolynomial<71, 7> { using ZPZ = aerobus::zpz<71>; using type = POLYV<ZPZV<1>,
ZPZV<0>, ZPZV<0>, ZPZV<0>, ZPZV<0>, ZPZV<0>, ZPZV<2>, ZPZV<64»; }; // NOLINT
03061 template<> struct ConwayPolynomial<71, 8> { using ZPZ = aerobus::zpz<71>; using type = POLYV<ZPZV<1>,
                       ZPZV<0>, ZPZV<0>, ZPZV<0>, ZPZV<0>, ZPZV<53>, ZPZV<22>, ZPZV<19>, ZPZV<7»; }; // NOLINT
 03062 template<> struct ConwayPolynomial<71, 9> { using ZPZ = aerobus::zpz<71>, using type = POLYV<ZPZV<1>,
ZPZV<0>, ZPZV<0>, ZPZV<0>, ZPZV<0>, ZPZV<0>, ZPZV<4>, ZPZV<43>, ZPZV<42>, ZPZV<62>, ZPZV<64*; }; // NOLINT
03063 template<> struct ConwayPolynomial<71, 10> { using ZPZ = aerobus::zpz<71>; using type = POLYV<ZPZV<1>,
                        ZPZV<0>, ZPZV<0>, ZPZV<0>, ZPZV<0>, ZPZV<53>, ZPZV<17>, ZPZV<26>, ZPZV<1>, ZPZV<40>, ZPZV<7»; }; //</pre>
                        NOLINT
03064 template<> struct ConwayPolynomial<71, 11> { using ZPZ = aerobus::zpz<71>; using type = POLYV<ZPZV<1>,
                         ZPZV<0>, ZPZ
                         }; // NOLINT
03065 template<> struct ConwayPolynomial<71, 12> { using ZPZ = aerobus::zpz<71>; using type = POLYV<ZPZV<1>,
                        ZPZV<0>, ZPZV<0>, ZPZV<0>, ZPZV<0>, ZPZV<0>, ZPZV<1>, ZPZV<28>, ZPZV<29>, ZPZV<55>, ZPZV<21, ZPZV<58>,
                        ZPZV<23>, ZPZV<7»; }; // NOLINT</pre>
03066 template<> struct ConwayPolynomial<71, 13> { using ZPZ = aerobus::zpz<71>; using type = POLYV<ZPZV<1>,
                         ZPZV<0>, ZPZV<0>
                        ZPZV<27>, ZPZV<64»; }; // NOLINT</pre>
 03067 template<> struct ConwayPolynomial<71, 15> { using ZPZ = aerobus::zpz<71>; using type = POLYV<ZPZV<1>,
ZPZV<0>, ZPZV<0>, ZPZV<0>, ZPZV<0>, ZPZV<0>, ZPZV<0>, ZPZV<0>, ZPZV<0>, ZPZV<0>, ZPZV<2>, ZPZV<20>, ZPZV<20>, ZPZV<32>, ZPZV<32>, ZPZV<32>, ZPZV<49>, ZPZV<49>, ZPZV<64»; }; // NOLINT</pre>
03068 template<> struct ConwayPolynomial<71, 17> { using ZPZ = aerobus::zpz<71>; using type = POLYV<ZPZV<1>,
```

```
ZPZV<0>, ZPZV<0>
                          ZPZV<0>, ZPZV<0>, ZPZV<0>, ZPZV<0>, ZPZV<3>, ZPZV<64»; }; // NOLINT</pre>
03069 template<> struct ConwayPolynomial<71, 19> { using ZPZ = aerobus::zpz<71>; using type = POLYV<ZPZV<1>,
                        ZPZV<0>, ZPZV<0>
03070 template<> struct ConwayPolynomial<73, 1> { using ZPZ = aerobus::zpz<73>; using type = POLYV<ZPZV<1>,
                         ZPZV<68»; }; // NOLINT
 03071 template<> struct ConwayPolynomial<73, 2> { using ZPZ = aerobus::zpz<73>; using type = POLYV<ZPZV<1>,
                         ZPZV<70>, ZPZV<5»; }; // NOLINT</pre>
 03072 template<> struct ConwayPolynomial<73, 3> { using ZPZ = aerobus::zpz<73>; using type = POLYV<ZPZV<1>,
                         ZPZV<0>, ZPZV<2>, ZPZV<68»; }; // NOLINT</pre>
 03073 template<> struct ConwayPolynomial<73, 4> { using ZPZ = aerobus::zpz<73>; using type = POLYV<ZPZV<1>,
 ZPZV<0>, ZPZV<16>, ZPZV<56>, ZPZV<55>; }; // NOLINT
03074 template<> struct ConwayPolynomial<73, 5> { using ZPZ = aerobus::zpz<73>; using type = POLYV<ZPZV<1>,
                         ZPZV<0>, ZPZV<0>, ZPZV<0>, ZPZV<9>, ZPZV<68»; }; // NOLINT</pre>
 03075 template<> struct ConwayPolynomial<73, 6> { using ZPZ = aerobus::zpz<73>; using type = POLYV<ZPZV<1>,
                        ZPZV<0>, ZPZV<0>, ZPZV<45>, ZPZV<23>, ZPZV<48>, ZPZV<5»; }; // NOLINT</pre>
03076 template<> struct ConwayPolynomial<73, 7> { using ZPZ = aerobus::zpz<73>; using type = POLYV<ZPZV<1>, ZPZV<0>, ZPZV<0 , ZPZ
03077 template<> struct ConwayPolynomial<73, 8> { using ZPZ = aerobus::zpz<73>; using type = POLYV<ZPZV<1>,
ZPZV<0>, ZPZV<0>, ZPZV<3>, ZPZV<3>, ZPZV<5>, ZPZV<53>, ZPZV<3>, ZPZV<39>, ZPZV<18>, ZPZV<5»; }; // NOLINT 03078 template<> struct ConwayPolynomial<73, 9> { using ZPZ = aerobus::zpz<73>; using type = POLYV<ZPZV<1>,
NOLINT
 03080 template<> struct ConwayPolynomial<73, 11> { using ZPZ = aerobus::zpz<73>; using type = POLYV<ZPZV<1>,
                         ZPZV<0>, ZPZV<0>
                           // NOLINT
03081 template<> struct ConwayPolynomial<73, 12> { using ZPZ = aerobus::zpz<73>; using type = POLYV<ZPZV<1>,
                         ZPZV<0>, ZPZV<0>, ZPZV<0>, ZPZV<1>, ZPZV<69>, ZPZV<52>, ZPZV<26>, ZPZV<20>, ZPZV<46>, ZPZV<29>,
                          ZPZV<25>, ZPZV<5»; }; // NOLINT</pre>
 03082 template<> struct ConwayPolynomial<73, 13> { using ZPZ = aerobus::zpz<73>; using type = POLYV<ZPZV<1>,
                          ZPZV<0>, ZPZV<0>,
                         ZPZV<7>, ZPZV<68»; }; // NOLINT
03083 template<> struct ConwayPolynomial<73, 15> { using ZPZ = aerobus::zpz<73>; using type = POLYV<ZPZV<1>,
                        ZPZV<0>, ZPZV<0>, ZPZV<0>, ZPZV<0>, ZPZV<0>, ZPZV<0>, ZPZV<0>, ZPZV<0>, ZPZV<0>, ZPZV<33>,
ZPZV<57>, ZPZV<57>, ZPZV<62>, ZPZV<68»; }; // NOLINT</pre>
 03084 template<> struct ConwayPolynomial<73, 17> { using ZPZ = aerobus::zpz<73>; using type = POLYV<ZPZV<1>,
ZPZV<0>, ZPZV<0 , ZPZ
 03086 template<> struct ConwayPolynomial<79, 1> { using ZPZ = aerobus::zpz<79>; using type = POLYV<ZPZV<1>,
                         ZPZV<76»; }; // NOLINT</pre>
 03087 template<> struct ConwayPolynomial<79, 2> { using ZPZ = aerobus::zpz<79>; using type = POLYV<ZPZV<1>,
                        ZPZV<78>, ZPZV<3»; }; // NOLINT</pre>
03088 template<> struct ConwayPolynomial<79, 3> { using ZPZ = aerobus::zpz<79>; using type = POLYV<ZPZV<1>,
                         ZPZV<0>, ZPZV<9>, ZPZV<76»; }; // NOLINT</pre>
03089 template<> struct ConwayPolynomial<79, 4> { using ZPZ = aerobus::zpz<79>; using type = POLYV<ZPZV<1>,
                          ZPZV<0>, ZPZV<2>, ZPZV<66>, ZPZV<3»; };
                                                                                                                                                                                             // NOLINT
 03090 template<> struct ConwayPolynomial<79, 5> { using ZPZ = aerobus::zpz<79>; using type = POLYV<ZPZV<1>,
ZPZV<0>, ZPZV<0>, ZPZV<0>, ZPZV<5>, ZPZV<76»; }; // NOLINT

03091 template<> struct ConwayPolynomial<79, 6> { using ZPZ = aerobus::zpz<79>; using type = POLYV<ZPZV<1>,
ZPZV<0>, ZPZV<1>, ZPZV<28>, ZPZV<28>, ZPZV<3»; }; // NOLINT
03092 template<> struct ConwayPolynomial<79, 7> { using ZPZ = aerobus::zpz<79>; using type = POLYV<ZPZV<1>,
                         ZPZV<0>, ZPZV<0>, ZPZV<0>, ZPZV<0>, ZPZV<0>, ZPZV<4>, ZPZV<76»; }; // NOLINT</pre>
 03093 template<> struct ConwayPolynomial<79, 8> { using ZPZ = aerobus::zpz<79>; using type = POLYV<ZPZV<1>,
                         03094 template<> struct ConwayPolynomial<79, 9> { using ZPZ = aerobus::zpz<79>; using type = POLYV<ZPZV<1>, ZPZV<0>, ZPZV<0>, ZPZV<0>, ZPZV<0>, ZPZV<0>, ZPZV<0>, ZPZV<57>, ZPZV<57>, ZPZV<57>; using type = POLYV<ZPZV<1>, 03095 template<> struct ConwayPolynomial<79, 10> { using ZPZ = aerobus::zpz<79>; using type = POLYV<ZPZV<1>, ZPZV<57>, ZPZV<57>; using type = POLYV<ZPZV<1>, ZPZV<57; using type = POLYV<2PZV<1>, ZPZV<57; using type = POLYV<2PZV<1>, ZPZV<57; using typ
                          ZPZV<0>, ZPZV<0>, ZPZV<0>, ZPZV<4>, ZPZV<44>, ZPZV<51>, ZPZV<1>, ZPZV<30>, ZPZV<42>, ZPZV<3»; }; //
03096 template<> struct ConwayPolynomial<79, 11> { using ZPZ = aerobus::zpz<79>; using type = POLYV<ZPZV<1>,
                        ZPZV<0>, ZPZV<0>
                           // NOLINT
03097 template<> struct ConwayPolynomial<79, 12> { using ZPZ = aerobus::zpz<79>; using type = POLYV<ZPZV<1>,
                          ZPZV<0>, ZPZV<0>, ZPZV<0>, ZPZV<1>, ZPZV<29>, ZPZV<45>, ZPZV<52>, ZPZV<7>, ZPZV<40>, ZPZV<59>,
                          ZPZV<62>, ZPZV<3»; }; // NOLINT</pre>
 03098 template<> struct ConwayPolynomial<79, 13> { using ZPZ = aerobus::zpz<79>; using type = POLYV<ZPZV<1>,
                        ZPZV<0>, ZPZV<78>,
ZPZV<4>, ZPZV<6*, ZPZV<6*, ZPZV<76*; }; // NOLINT
03099 template<> struct ConwayPolynomial<79, 17> { using ZPZ = aerobus::zpz<79>; using type = POLYV<ZPZV<1>,
                          ZPZV<0>, ZPZ
                          ZPZV<0>, ZPZV<0>, ZPZV<0>, ZPZV<0>, ZPZV<25>, ZPZV<76»; };  // NOLINT</pre>
03100 template<> struct ConwayPolynomial<79, 19> { using ZPZ = aerobus::zpz<79>; using type = POLYV<ZPZV<1>,
ZPZV<0>, ZPZ
                         ZPZV<81»; }; // NOLINT
 03102 template<> struct ConwayPolynomial<83, 2> { using ZPZ = aerobus::zpz<83>; using type = POLYV<ZPZV<1>,
                         ZPZV<82>, ZPZV<2»; }; // NOLINT</pre>
 03103 template<> struct ConwayPolynomial<83, 3> { using ZPZ = aerobus::zpz<83>; using type = POLYV<ZPZV<1>,
                        ZPZV<0>, ZPZV<3>, ZPZV<81»; }; // NOLINT</pre>
 03104 template<> struct ConwayPolynomial<83, 4> { using ZPZ = aerobus::zpz<83>; using type = POLYV<ZPZV<1>,
```

```
// NOLINT
                   ZPZV<0>, ZPZV<4>, ZPZV<42>, ZPZV<2»; };</pre>
03105 template<> struct ConwayPolynomial<83, 5> { using ZPZ = aerobus::zpz<83>; using type = POLYV<ZPZV<1>,
                  ZPZV<0>, ZPZV<0>, ZPZV<0>, ZPZV<9>, ZPZV<81»; }; // NOLINT</pre>
 03106 template<> struct ConwayPolynomial<83, 6> { using ZPZ = aerobus::zpz<83>; using type = POLYV<ZPZV<1>,
                   \mbox{ZPZV<0>, ZPZV<1>, ZPZV<76>, ZPZV<32>, ZPZV<17>, ZPZV<2*; }; // \mbox{NOLINT} 
03107 template<> struct ConwayPolynomial<83, 7> { using ZPZ = aerobus::zpz<83>; using type = POLYV<ZPZV<1>,
                   ZPZV<0>, ZPZV<0>, ZPZV<0>, ZPZV<0>, ZPZV<3>, ZPZV<81»; }; // NOLINT</pre>
 03108 template<> struct ConwayPolynomial<83, 8> { using ZPZ = aerobus::zpz<83>; using type = POLYV<ZPZV<1>,
ZPZV<0>, ZPZV<0>, ZPZV<1>, ZPZV<1>, ZPZV<65>, ZPZV<23>, ZPZV<42>, ZPZV<2»; }; // NOLINT
03109 template<> struct ConwayPolynomial<83, 9> { using ZPZ = aerobus::zpz<83>; using type = POLYV<ZPZV<1>,
ZPZV<0>, ZPZV<0>, ZPZV<0>, ZPZV<0>, ZPZV<0>, ZPZV<0>, ZPZV<1>, ZPZV<24>, ZPZV<18>, ZPZV<18); }; // NOLINT
03110 template<> struct ConwayPolynomial<83, 10> { using ZPZ = aerobus::zpz<83>; using type = POLYV<ZPZV<1>,
                   ZPZV<0>, ZPZV<0>, ZPZV<0>, ZPZV<0>, ZPZV<7>, ZPZV<7>, ZPZV<73>, ZPZV<7>, ZPZV<73>, ZPZV<53>, ZPZV<2»; };</pre>
 03111 template<> struct ConwayPolynomial<83, 11> { using ZPZ = aerobus::zpz<83>; using type = POLYV<ZPZV<1>,
                   ZPZV<0>, ZPZV<0>
                   l: // NOLINT
03112 template<> struct ConwayPolynomial<83, 12> { using ZPZ = aerobus::zpz<83>; using type = POLYV<ZPZV<1>, ZPZV<0>, ZPZV<0>, ZPZV<0>, ZPZV<0>, ZPZV<0>, ZPZV<3>, ZPZV<12>, ZPZV<31>, ZPZV<19>, ZPZV<65>, ZPZV<55>,
                   ZPZV<75>, ZPZV<2»; }; // NOLINT</pre>
 03113 template<> struct ConwayPolynomial<83, 13> { using ZPZ = aerobus::zpz<83>; using type = POLYV<ZPZV<1>,
                  ZPZV<0>, ZPZV<0>
03114 template<> struct ConwayPolynomial<83, 17> { using ZPZ = aerobus::zpz<83>; using type = POLYV<ZPZV<1>, ZPZV<0>, ZP
 03115 template<> struct ConwayPolynomial<83, 19> { using ZPZ = aerobus::zpz<83>; using type = POLYV<ZPZV<1>,
ZPZV<0>, ZPZ
                  ZPZV<86»; }; // NOLINT</pre>
03117 template<> struct ConwayPolynomial<89, 2> { using ZPZ = aerobus::zpz<89>; using type = POLYV<ZPZV<1>,
                   03118 template<> struct ConwayPolynomial<89, 3> { using ZPZ = aerobus::zpz<89>; using type = POLYV<ZPZV<1>,
                   ZPZV<0>, ZPZV<3>, ZPZV<86»; }; // NOLINT
 03119 template<> struct ConwayPolynomial<89, 4> { using ZPZ = aerobus::zpz<89>; using type = POLYV<ZPZV<1>,
                   ZPZV<0>, ZPZV<4>, ZPZV<72>, ZPZV<3»; }; // NOLINT</pre>
 03120 template<> struct ConwayPolynomial<89, 5> { using ZPZ = aerobus::zpz<89>; using type = POLYV<ZPZV<1>,
                   ZPZV<0>, ZPZV<0>, ZPZV<0>, ZPZV<1>, ZPZV<86»; }; // NOLINT</pre>
 03121 template<> struct ConwayPolynomial<89, 6> { using ZPZ = aerobus::zpz<89>; using type = POLYV<ZPZV<1>,
                  ZPZV<0>, ZPZV<1>, ZPZV<82>, ZPZV<80>, ZPZV<15>, ZPZV<3»; }; // NOLINT</pre>
03122 template<> struct ConwayPolynomial<89, 7> { using ZPZ = aerobus::zpz<89>; using type = POLYV<ZPZV<1>,
                  ZPZV<0>, ZPZV<0>, ZPZV<0>, ZPZV<0>, ZPZV<7>, ZPZV<86»; }; // NOLINT</pre>
03123 template<> struct ConwayPolynomial<89, 8> { using ZPZ = aerobus::zpz<89>; using type = POLYV<ZPZV<1>,
                   ZPZV<0>, ZPZV<0>, ZPZV<0>, ZPZV<0>, ZPZV<65>, ZPZV<40>, ZPZV<79>, ZPZV<3»; }; // NOLINT</pre>
 03124 template<> struct ConwayPolynomial<89, 9> { using ZPZ = aerobus::zpz<89>; using type = POLYV<ZPZV<1>,
ZPZV<0>, ZPZV<0>, ZPZV<0>, ZPZV<0>, ZPZV<0>, ZPZV<5>, ZPZV<5>, ZPZV<5>, ZPZV<6>, ZPZV<66, ZPZV<66, ZPZV<66, ZPZV<66, ZPZV<68; ); // NOLINT 03125 template<> struct ConwayPolynomial<89, 10> { using ZPZ = aerobus::zpz<89>; using type = POLYV<ZPZV<1>,
                   ZPZV<0>, ZPZV<0>, ZPZV<0>, ZPZV<1>, ZPZV<16>, ZPZV<33>, ZPZV<82>, ZPZV<52>, ZPZV<4>, ZPZV<3»; };
                   NOLTNT
03126 template<> struct ConwayPolynomial<89, 11> { using ZPZ = aerobus::zpz<89>; using type = POLYV<ZPZV<1>,
                   ZPZV<0>, ZPZV<86»;
                     }; // NOLINT
03127 template<> struct ConwayPolynomial<89, 12> { using ZPZ = aerobus::zpz<89>; using type = POLYV<ZPZV<1>,
                   ZPZV<0>, ZPZV<0>, ZPZV<0>, ZPZV<2>, ZPZV<85>, ZPZV<15>, ZPZV<44>, ZPZV<51>, ZPZV<51>, ZPZV<70>,
                   ZPZV<52>, ZPZV<3»; }; // NOLINT</pre>
03128 template<> struct ConwayPolynomial<89, 13> { using ZPZ = aerobus::zpz<89>; using type = POLYV<ZPZV<1>,
                   ZPZV<0>, ZPZV<0>,
                   ZPZV<17>, ZPZV<86»; }; // NOLINT</pre>
03129 template<> struct ConwayPolynomial<89, 17> { using ZPZ = aerobus::zpz<89>; using type = POLYV<ZPZV<1>,
ZPZV<0>, ZPZ
                  ZPZV<0>, ZPZ
03131 template<> struct ConwayPolynomial<97, 1> { using ZPZ = aerobus::zpz<97>; using type = POLYV<ZPZV<1>,
                   ZPZV < 92»; }; // NOLINT
03132 template<> struct ConwayPolynomial<97, 2> { using ZPZ = aerobus::zpz<97>; using type = POLYV<ZPZV<1>,
                  ZPZV<96>, ZPZV<5»; }; // NOLINT</pre>
 03133 template<> struct ConwayPolynomial<97, 3> { using ZPZ = aerobus::zpz<97>; using type = POLYV<ZPZV<1>,
                   ZPZV<0>, ZPZV<9>, ZPZV<92»; }; // NOLINT</pre>
 03134 template<> struct ConwayPolynomial<97, 4> { using ZPZ = aerobus::zpz<97>; using type = POLYV<ZPZV<1>,
ZPZV<0>, ZPZV<6>, ZPZV<80>, ZPZV<5»; }; // NOLINT
03135 template<> struct ConwayPolynomial<97, 5> { using ZPZ = aerobus::zpz<97>; using type = POLYV<ZPZV<1>,
 ZPZV<0>, ZPZV<0>, ZPZV<0>, ZPZV<0>, ZPZV<0>, ZPZV<0>, ZPZV<0>, ZPZV<2>; }; // NOLINT
03136 template<> struct ConwayPolynomial<97, 6> { using ZPZ = aerobus::zpz<97>; using type = POLYV<ZPZV<1>,
                   ZPZV<0>, ZPZV<0>, ZPZV<92>, ZPZV<58>, ZPZV<88>, ZPZV<5»; }; // NOLINT</pre>
 03137 template<> struct ConwayPolynomial<97, 7> { using ZPZ = aerobus::zpz<97>; using type = POLYV<ZPZV<1>,
ZPZV<0>, ZPZV<0>, ZPZV<0>, ZPZV<0>, ZPZV<0>, ZPZV<5>, ZPZV<92»; }; // NOLINT
03138 template<> struct ConwayPolynomial<97, 8> { using ZPZ = aerobus::zpz<97>; using type = POLYV<ZPZV<1>,
                   ZPZV<0>, ZPZV<0>, ZPZV<0>, ZPZV<0>, ZPZV<65>, ZPZV<1>, ZPZV<32>, ZPZV<5»; }; // NOLINT
 03139 template<> struct ConwayPolynomial<97, 9> { using ZPZ = aerobus::zpz<97>; using type = POLYV<ZPZV<1>,
 ZPZV<0>, ZPZV<0>, ZPZV<0>, ZPZV<0>, ZPZV<0>, ZPZV<0>, ZPZV<1>, ZPZV<12>, ZPZV<5>, ZPZV<7>, ZPZV<92; }; // NOLINT 03140 template<> struct ConwayPolynomial<97, 10> { using ZPZ = aerobus::zpz<97>; using type = POLYV<ZPZV<1>,
                   ZPZV<0>, ZPZV<0>, ZPZV<0>, ZPZV<0>, ZPZV<22>, ZPZV<66>, ZPZV<34>, ZPZV<34>, ZPZV<20>, ZPZV<5»; }; //</pre>
                   NOLINT
 03141 template<> struct ConwayPolynomial<97, 11> { using ZPZ = aerobus::zpz<97>; using type = POLYV<ZPZV<1>,
```

```
ZPZV<0>, ZPZV<0>; };
03142 template<> struct ConwayPolynomial<97, 12> { using ZPZ = aerobus::zpz<97>; using type = POLYV<ZPZV<1>,
                           \texttt{ZPZV} < \texttt{0>, ZPZV} < \texttt{0>, ZPZV} < \texttt{0>, ZPZV} < \texttt{0>, ZPZV} < \texttt{30>, ZPZV} < \texttt{59>, ZPZV} < \texttt{81>, ZPZV} < \texttt{0>, ZPZV} < \texttt{86>, ZPZV} < \texttt{78>, ZPZV} < \texttt{81>, ZPZV} < \texttt{81>, ZPZV} < \texttt{82>, ZPZV} < \texttt{83>, ZPZV} < \texttt{84>, ZPZV} 
                          ZPZV<94>, ZPZV<5»; }; // NOLINT
 03143 template>> struct ConwayPolynomial<97, 13> { using ZPZ = aerobus::zpz<97>; using type = POLYV<ZPZV<1>,
                           ZPZV<0>, ZPZV<0>
                           ZPZV<3>, ZPZV<92»; }; // NOLINT
03144 template<> struct ConwayPolynomial<97, 17> { using ZPZ = aerobus::zpz<97>; using type = POLYV<ZPZV<1>,
ZPZV<0>, ZPZ
                          ZPZV<0>, ZPZ
 03146 template<> struct ConwayPolynomial<101, 1> { using ZPZ = aerobus::zpz<101>; using type =
                          POLYV<ZPZV<1>, ZPZV<99»; }; // NOLINT
03147 template<> struct ConwayPolynomial<101, 2> { using ZPZ = aerobus::zpz<101>; using type =
POLYV<ZPZV<1>, ZPZV<97>, ZPZV<2»; }; // NOLINT

03148 template<> struct ConwayPolynomial<101, 3> { using ZPZ = aerobus::zpz<101>; using type =
                          POLYV<ZPZV<1>, ZPZV<0>, ZPZV<3>, ZPZV<99»; }; // NOLINT
 03149 template<> struct ConwayPolynomial<101, 4> { using ZPZ = aerobus::zpz<101>; using type =
POLYY<ZPZV<1>, ZPZV<0>, ZPZV<1>, ZPZV<78>, ZPZV<2»; }; // NOLINT
03150 template<> struct ConwayPolynomial<101, 5> { using ZPZ = aerobus::zpz<101>; using type =
                          POLYV<ZPZV<1>, ZPZV<0>, ZPZV<0>, ZPZV<0>, ZPZV<2>, ZPZV<99»; }; // NOLINT
03151 template<> struct ConwayPolynomial<101, 6> { using ZPZ = aerobus::zpz<101>; using type =
                          POLYV<ZPZV<1>, ZPZV<0>, ZPZV<2>, ZPZV<90>, ZPZV<20>, ZPZV<67>, ZPZV<2»; }; // NOLINT
 03152 template<> struct ConwayPolynomial<101, 7> { using ZPZ = aerobus::zpz<101>; using type
                          POLYV<ZPZV<1>, ZPZV<0>, ZPZV<0>, ZPZV<0>, ZPZV<0>, ZPZV<6>, ZPZV<6>, ZPZV<6>, ZPZV<99»; }; // NOLINT
 03153 template<> struct ConwayPolynomial<101, 8> { using ZPZ = aerobus::zpz<101>; using type =
                          POLYV<ZPZV<1>, ZPZV<0>, ZPZV<0>, ZPZV<0>, ZPZV<4>, ZPZV<76>, ZPZV<29>, ZPZV<24>, ZPZV<2»; }; //
                          NOLINT
03154 template<> struct ConwayPolynomial<101, 9> { using ZPZ = aerobus::zpz<101>; using type
                          POLYV<ZPZV<1>, ZPZV<0>, ZPZV<0>, ZPZV<0>, ZPZV<0>, ZPZV<0>, ZPZV<0>, ZPZV<0>, ZPZV<64>, ZPZV<47>, ZPZV<99»; };
03155 template<> struct ConwayPolynomial<101, 10> { using ZPZ = aerobus::zpz<101>; using type = POLYV<ZPZV<1>, ZPZV<0>, ZPZV<0>, ZPZV<0>, ZPZV<1>, ZPZV<67>, ZPZV<49>, ZPZV<100>, ZPZV<100>, ZPZV<52>,
                          ZPZV<2»; }; // NOLINT</pre>
 03156 template<> struct ConwayPolynomial<101, 11> { using ZPZ = aerobus::zpz<101>; using type
                          POLYV<ZPZV<1>, ZPZV<0>, ZPZV<0
                          ZPZV<31>, ZPZV<99»; };</pre>
                                                                                                                           // NOLINT
03157 template<> struct ConwayPolynomial<101, 12> { using ZPZ = aerobus::zpz<101>; using type =
                          POLYV<ZPZV<1>, ZPZV<0>, ZPZV<0>, ZPZV<0>, ZPZV<1>, ZPZV<79>, ZPZV<64>, ZPZV<39>, ZPZV<78>, ZPZV<48>, ZPZV<84>, ZPZV<39>, ZPZV<78>, ZPZV<84>, ZPZV<84 , ZPZV<
03158 template<> struct ConwayPolynomial<101, 13> { using ZPZ = aerobus::zpz<101>; using type =
                          POLYV<ZPZV<1>, ZPZV<0>, ZPZV<0
                           ZPZV<0>, ZPZV<0>, ZPZV<7>, ZPZV<99»; };</pre>
                                                                                                                                                                                                          // NOLINT
03159 template<> struct ConwayPolynomial<101, 17> { using ZPZ = aerobus::zpz<101>; using type =
POLYV<ZPZV<1>, ZPZV<0>, ZPZV<0 , ZPZV<0
                           ZPZV<0>, ZPZV<0>, ZPZV<0>, ZPZV<0>, ZPZV<0>, ZPZV<0>, ZPZV<0>, ZPZV<0>, ZPZV<4>, ZPZV<99»; }; //</pre>
                          NOLINT
03161 template<> struct ConwayPolynomial<103, 1> { using ZPZ = aerobus::zpz<103>; using type =
                          POLYV<ZPZV<1>, ZPZV<98»; }; // NOLINT
03162 template<> struct ConwayPolynomial<103, 2> { using ZPZ = aerobus::zpz<103>; using type =
                          POLYV<ZPZV<1>, ZPZV<102>, ZPZV<5»; }; // NOLINT
 03163 template<> struct ConwayPolynomial<103, 3> { using ZPZ = aerobus::zpz<103>; using type =
                          POLYV<ZPZV<1>, ZPZV<0>, ZPZV<2>, ZPZV<98»; }; // NOLINT
 03164 template<> struct ConwayPolynomial<103, 4> { using ZPZ = aerobus::zpz<103>; using type =
POLYV<ZPZV<1>, ZPZV<2>, ZPZV<88>, ZPZV<5»; }; // NOLINT

03165 template<> struct ConwayPolynomial<103, 5> { using ZPZ = aerobus::zpz<103>; using type =
                          POLYV<ZPZV<1>, ZPZV<0>, ZPZV<0>, ZPZV<0>, ZPZV<11>, ZPZV<98»; }; // NOLINT
 03166 template<> struct ConwayPolynomial<103, 6> { using ZPZ = aerobus::zpz<103>; using type =
                          POLYV<ZPZV<1>, ZPZV<0>, ZPZV<0>, ZPZV<96>, ZPZV<95>, ZPZV<30>, ZPZV<5»; }; // NOLINT
 03167 template<> struct ConwayPolynomial<103, 7> { using ZPZ = aerobus::zpz<103>; using type =
                          POLYV<ZPZV<1>, ZPZV<0>, ZPZV<0>, ZPZV<0>, ZPZV<0>, ZPZV<0>, ZPZV<5>, ZPZV<98»; };
03168 template<> struct ConwayPolynomial<103, 8> { using ZPZ = aerobus::zpz<103>; using type =
                          POLYV<ZPZV<1>, ZPZV<0>, ZPZV<0>, ZPZV<0>, ZPZV<1>, ZPZV<71>, ZPZV<71>, ZPZV<49>, ZPZV<5»; }; //
                          NOLINT
03169 template<> struct ConwayPolynomial<103, 9> { using ZPZ = aerobus::zpz<103>; using type =
                          POLYV<ZPZV<1>, ZPZV<0>, ZPZV<0>, ZPZV<0>, ZPZV<0>, ZPZV<0>, ZPZV<0>, ZPZV<97>, ZPZV<97>, ZPZV<51>, ZPZV<98»; };
                             // NOLINT
03170 template<> struct ConwayPolynomial<103, 10> { using ZPZ = aerobus::zpz<103>; using type =
                          POLYV<ZPZV<1>, ZPZV<0>, ZPZV<0>, ZPZV<0>, ZPZV<1>, ZPZV<101>, ZPZV<86>, ZPZV<101>, ZPZV<94>, ZPZV<11>, ZPZV<5»; }; // NOLINT
 03171 template<> struct ConwayPolynomial<103, 11> { using ZPZ = aerobus::zpz<103>; using type
                          POLYV<ZPZV<1>, ZPZV<0>, ZPZV<0
03172 template<> struct ConwayPolynomial<103, 12> { using ZPZ = aerobus::zpz<103>; using type =
                          POLYV<ZPZV<1>, ZPZV<0>, ZPZV<0>, ZPZV<0>, ZPZV<1>, ZPZV<74>, ZPZV<23>, ZPZV<94>, ZPZV<20>, ZPZV<81>,
                           ZPZV<29>, ZPZV<88>, ZPZV<5»; };</pre>
                                                                                                                                                                        // NOLINT
 03173 template<> struct ConwayPolynomial<103, 13> { using ZPZ = aerobus::zpz<103>; using type
                          POLYV<ZPZV<1>, ZPZV<0>, ZPZV<0>, ZPZV<0>, ZPZV<0>, ZPZV<0>, ZPZV<0>, ZPZV<0>, ZPZV<0>, ZPZV<0>, ZPZV<0>,
ZPZV<0>, ZPZV<0>, ZPZV<5>, ZPZV<98»; }; // NOLINT
03174 template<> struct ConwayPolynomial<103, 17> { using ZPZ = aerobus::zpz<103>; using type =
POLYV<ZPZV<1>, ZPZV<0>, ZPZV<0>,
```

```
ZPZV<0>, ZPZV<0>, ZPZV<0>, ZPZV<0>, ZPZV<0>, ZPZV<102>, ZPZV<8>, ZPZV<98»; };</pre>
03175 template<> struct ConwayPolynomial<103, 19> { using ZPZ = aerobus::zpz<103>; using type =
                              POLYV<ZPZV<1>, ZPZV<0>, ZPZV<0>, ZPZV<0>, ZPZV<0>, ZPZV<0>, ZPZV<0>, ZPZV<0>, ZPZV<0>, ZPZV<0>, ZPZV<0>,
                               \texttt{ZPZV} < \texttt{0>, ZPZV} < \texttt{2>, ZPZV} < \texttt{2>, ZPZV} < \texttt{3}; \ // \texttt
                              NOLINT
03176 template<> struct ConwayPolynomial<107, 1> { using ZPZ = aerobus::zpz<107>; using type =
                             POLYV<ZPZV<1>, ZPZV<105»; }; // NOLINT
 03177 template<> struct ConwayPolynomial<107, 2> { using ZPZ = aerobus::zpz<107>; using type =
POLYV<ZPZV<1>, ZPZV<103, ZPZV<2»; }; // NOLINT
03178 template<> struct ConwayPolynomial<107, 3> { using ZPZ = aerobus::zpz<107>; using type =
POLYV<ZPZV<1>, ZPZV<0>, ZPZV<5>, ZPZV<105»; }; // NOLINT

03179 template<> struct ConwayPolynomial<107, 4> { using ZPZ = aerobus::zpz<107>; using type = POLYV<ZPZV<1>, ZPZV<0>, ZPZV<13>, ZPZV<79>, ZPZV<2»; }; // NOLINT

03180 template<> struct ConwayPolynomial<107, 5> { using ZPZ = aerobus::zpz<107>; using type =
                              POLYV<ZPZV<1>, ZPZV<0>, ZPZV<0>, ZPZV<0>, ZPZV<8>, ZPZV<105»; }; // NOLINT
03181 template<> struct ConwayPolynomial</br>
O3181 template
POLYV<ZPZV<1>, ZPZV<0>, ZPZV<1>, ZPZV<52>, ZPZV<22>, ZPZV<79>, ZPZV<2»; }; // NOLINT</pre>
03182 template<> struct ConwayPolynomial<107, 7> { using ZPZ = aerobus::zpz<107>; using type
                             POLYV<ZPZV<1>, ZPZV<0>, ZPZV<0>, ZPZV<0>, ZPZV<0>, ZPZV<0>, ZPZV<16>, ZPZV<16>, ZPZV<105»; }; // NOLINT
 03183 template<> struct ConwayPolynomial<107, 8> { using ZPZ = aerobus::zpz<107>; using type
                              POLYV<ZPZV<1>, ZPZV<0>, ZPZV<0>, ZPZV<0>, ZPZV<2>, ZPZV<105>, ZPZV<24>, ZPZV<95>, ZPZV<95>, ZPZV<2»; }; //
                              NOLINT
03184 template<> struct ConwayPolynomial<107, 9> { using ZPZ = aerobus::zpz<107>; using type =
                              POLYV<ZPZV<1>, ZPZV<0>, ZPZV<0>, ZPZV<0>, ZPZV<0>, ZPZV<3>, ZPZV<3>, ZPZV<366>, ZPZV<105»; };
                                // NOLINT
03185 template<> struct ConwayPolynomial<107, 10> { using ZPZ = aerobus::zpz<107>; using type
                              POLYV<ZPZV<1>, ZPZV<0>, ZPZV<0>, ZPZV<0>, ZPZV<9>, ZPZV<94>, ZPZV<61>, ZPZV<83>, ZPZV<83>, ZPZV<85>,
                              ZPZV<2»; }; // NOLINT</pre>
03186 template<> struct ConwayPolynomial<107, 11> { using ZPZ = aerobus::zpz<107>; using type
                             POLYV<ZPZV<1>, ZPZV<0>, ZPZV<0
03187 template<> struct ConwayPolynomial<107, 12> { using ZPZ = aerobus::zpz<107>; using type
                              POLYY<ZPZV<1>, ZPZV<0>, ZPZV<0>, ZPZV<0>, ZPZV<0>, ZPZV<0>, ZPZV<6>, ZPZV<6>, ZPZV<61>, ZPZV<42>, ZPZV<48>, ZPZV<6>, ZPZV<6>, ZPZV<61>, ZPZV<42>, ZPZV<57>, ZPZV<29; }; // NOLINT
03188 template<> struct ConwayPolynomial<107, 13> { using ZPZ = aerobus::zpz<107>; using type =
                              POLYV<ZPZV<1>, ZPZV<0>, ZPZV<0
ZPZV<0>, ZPZV<0>, ZPZV<4>, ZPZV<105»; }; // NOLINT
03189 template<> struct ConwayPolynomial<107, 17> { using ZPZ = aerobus::zpz<107>; using type
                              POLYV<ZPZV<1>, ZPZV<0>, ZPZV<0
ZPZV<0>, ZPZV<0>, ZPZV<0>, ZPZV<0>, ZPZV<0>, ZPZV<0>, ZPZV<0>, ZPZV<2>, ZPZV<2>, ZPZV<205, ZPZV<
                             POLYV<ZPZV<0>, ZPZV<0>, ZPZV<0
03191 template<> struct ConwayPolynomial<109, 1> { using ZPZ = aerobus::zpz<109>; using type =
                             POLYV<ZPZV<1>, ZPZV<103»; }; // NOLINT
 03192 template<> struct ConwayPolynomial<109, 2> { using ZPZ = aerobus::zpz<109>; using type =
POLYV<ZPZV<1>, ZPZV<108>, ZPZV<6»; }; // NOLINT
03193 template<> struct ConwayPolynomial<109, 3> { using ZPZ = aerobus::zpz<109>; using type =
                             POLYV<ZPZV<1>, ZPZV<0>, ZPZV<1>, ZPZV<103»; }; // NOLINT
03194 template<> struct ConwayPolynomial<109, 4> { using ZPZ = aerobus::zpz<109>; using type =
POLYV<ZPZV<1>, ZPZV<0>, ZPZV<11>, ZPZV<98>, ZPZV<6»; }; // NOLINT
03195 template<> struct ConwayPolynomial<109, 5> { using ZPZ = aerobus::zpz<109>; using type =
                             POLYV<ZPZV<1>, ZPZV<0>, ZPZV<0>, ZPZV<4>, ZPZV<103»; }; // NOLINT
 03196 template<> struct ConwayPolynomial<109, 6> { using ZPZ = aerobus::zpz<109>; using type =
                             POLYV<ZPZV<1>, ZPZV<0>, ZPZV<0>, ZPZV<107>, ZPZV<102>, ZPZV<66>, ZPZV<68>; }; // NOLINT
 03197 template<> struct ConwayPolynomial<109, 7> { using ZPZ = aerobus::zpz<109); using type =
                              POLYV<ZPZV<1>, ZPZV<0>, ZPZV<0>, ZPZV<0>, ZPZV<0>, ZPZV<0>, ZPZV<14>, ZPZV<14>, ZPZV<103»; };
 03198 template<> struct ConwayPolynomial<109, 8> { using ZPZ = aerobus::zpz<109>; using type =
                              POLYV<ZPZV<1>, ZPZV<0>, ZPZV<0>, ZPZV<0>, ZPZV<1>, ZPZV<102>, ZPZV<34>, ZPZV<86>, ZPZV<6»; }; //
                              NOLINT
03199 template<> struct ConwayPolynomial<109, 9> { using ZPZ = aerobus::zpz<109>; using type =
                             POLYV<ZPZV<1>, ZPZV<0>, ZPZV<0>, ZPZV<0>, ZPZV<0>, ZPZV<0>, ZPZV<0>, ZPZV<0>, ZPZV<9>, ZPZV<9 , ZPZV<9
 03200 template<> struct ConwayPolynomial<109, 10> { using ZPZ = aerobus::zpz<109>; using type =
                              POLYV<ZPZV<1>, ZPZV<0>, ZPZV<0>, ZPZV<0>, ZPZV<1>, ZPZV<71>, ZPZV<55>, ZPZV<16>, ZPZV<75>, ZPZV<69>,
                              ZPZV<6»; }; // NOLINT</pre>
03201 template<> struct ConwayPolynomial<109, 11> { using ZPZ = aerobus::zpz<109>; using type = POLYV<ZPZV<1>, ZPZV<0>, ZPZV<11>, ZPZV<103»; }; // NOLINT
03202 template<> struct ConwayPolynomial<109, 12> { using ZPZ = aerobus::zpz<109>; using type
                              POLYV<ZPZV<1>, ZPZV<0>, ZPZV<0>, ZPZV<0>, ZPZV<1>, ZPZV<50>, ZPZV<53>, ZPZV<37>, ZPZV<37>, ZPZV<85>,
                              ZPZV<103>, ZPZV<28>, ZPZV<6»; }; // NOLINT</pre>
03203 template<> struct ConwayPolynomial<109, 13> { using ZPZ = aerobus::zpz<109>; using type =
                              POLYV<ZPZV<1>, ZPZV<0>, ZPZV<0
                               ZPZV<0>, ZPZV<0>, ZPZV<1>, ZPZV<103»; }; // NOLINT
 03204 template<> struct ConwayPolynomial<109, 17> { using ZPZ = aerobus::zpz<109>; using type
                              POLÝV<ZPZV<1>, ZPZV<0>, ZPZV<0
ZPZV<0>, ZPZV<0>, ZPZV<0>, ZPZV<0>, ZPZV<0>, ZPZV<0>, ZPZV<1>, ZPZV<1>, ZPZV<103»; }; // NOLINT
03205 template<> struct ConwayPolynomial<109, 19> { using ZPZ = aerobus::zpz<109>; using type =
POLYV<ZPZV<1>, ZPZV<0>, ZPZV<0>
                               ZPZV<0>, ZPZV<15>, ZPZV<103»; }; //</pre>
 03206 template<> struct ConwayPolynomial<113, 1> { using ZPZ = aerobus::zpz<113>; using type =
                             POLYV<ZPZV<1>, ZPZV<110»; }; // NOLINT
 03207 template<> struct ConwayPolynomial<113, 2> { using ZPZ = aerobus::zpz<113>; using type =
                              POLYV<ZPZV<1>, ZPZV<101>, ZPZV<3»; };
```

```
03208 template<> struct ConwayPolynomial<113, 3> { using ZPZ = aerobus::zpz<113>; using type =
                   POLYV<ZPZV<1>, ZPZV<0>, ZPZV<8>, ZPZV<110»; }; // NOLINT
 03209 template<> struct ConwayPolynomial<113, 4> { using ZPZ = aerobus::zpz<113>; using type =
POLYV<ZPZV<1>, ZPZV<0>, ZPZV<0>, ZPZV<62>, ZPZV<3»; }; // NOLINT
03210 template<> struct ConwayPolynomial<113, 5> { using ZPZ = aerobus::zpz<113>; using type =
                   POLYV<ZPZV<1>, ZPZV<0>, ZPZV<0>, ZPZV<0>, ZPZV<7>, ZPZV<110»; }; // NOLINT
 03211 template<> struct ConwayPolynomial<113, 6> { using ZPZ = aerobus::zpz<113>; using type =
                   POLYV<ZPZV<1>, ZPZV<0>, ZPZV<1>, ZPZV<59>, ZPZV<30>, ZPZV<71>, ZPZV<3»; }; // NOLINT
 03212 template<> struct ConwayPolynomial<113, 7> { using ZPZ = aerobus::zpz<113>; using type
                  POLYV<ZPZV<1>, ZPZV<0>, ZPZV<0>, ZPZV<0>, ZPZV<0>, ZPZV<0>, ZPZV<5>, ZPZV<5>, ZPZV<110»; };
 03213 template<> struct ConwayPolynomial<113, 8> { using ZPZ = aerobus::zpz<113>; using type =
                   POLYV<ZPZV<1>, ZPZV<0>, ZPZV<0>, ZPZV<3>, ZPZV<98>, ZPZV<38>, ZPZV<28>, ZPZV<28>, ZPZV<3»; }; //
 03214 template<> struct ConwayPolynomial<113, 9> { using ZPZ = aerobus::zpz<113>; using type =
                   POLYV<ZPZV<1>, ZPZV<0>, ZPZV<0>, ZPZV<0>, ZPZV<0>, ZPZV<0>, ZPZV<0>, ZPZV<0>, ZPZV<8>, ZPZV<87>, ZPZV<71>, ZPZV<110»; };
03215 template<> struct ConwayPolynomial<113, 10> { using ZPZ = aerobus::zpz<113>; using type =
                   POLYV<ZPZV<1>, ZPZV<0>, ZPZV<0>, ZPZV<0>, ZPZV<0>, ZPZV<56>,
                   ZPZV<3»; }; // NOLINT</pre>
 03216 template<> struct ConwayPolynomial<113, 11> { using ZPZ = aerobus::zpz<113>; using type
                   POLYV<ZPZV<1>, ZPZV<0>, ZPZV<0
03217 template<> struct ConwayPolynomial<113, 12> { using ZPZ = aerobus::zpz<113>; using type =
                   POLYV<ZPZV<1>, ZPZV<0>, ZPZV<0>, ZPZV<0>, ZPZV<1>, ZPZV<23>, ZPZV<62>, ZPZV<44>, ZPZV<98>, ZPZV<56>,
                   ZPZV<10>, ZPZV<27>, ZPZV<3»; }; // NOLINT</pre>
03218 template<> struct ConwayPolynomial<113, 13> { using ZPZ = aerobus::zpz<113>; using type
                   POLYV<ZPZV<1>, ZPZV<0>, ZPZV<0
                   ZPZV<0>, ZPZV<0>, ZPZV<4>, ZPZV<110»; }; // NOLINT</pre>
03219 template<> struct ConwayPolynomial<113, 17> { using ZPZ = aerobus::zpz<113>; using type
POLYY<ZPZV<1>, ZPZV<0>, ZPZV<110»; }; // NOLINT 03220 template<> struct ConwayPolynomial<113, 19> { using ZPZ = aerobus::zpz<113>; using type =
                   POLYV<ZPZV<1>, ZPZV<0>, ZPZV<0
                   ZPZV<0>, ZPZV<0>, ZPZV<0>, ZPZV<0>, ZPZV<0>, ZPZV<0>, ZPZV<0>, ZPZV<0>, ZPZV<22>, ZPZV<110»; }; //</pre>
                   NOLINT
03221 template<> struct ConwayPolynomial<127, 1> { using ZPZ = aerobus::zpz<127>; using type =
                  POLYV<ZPZV<1>, ZPZV<124»; }; // NOLINT
 03222 template<> struct ConwayPolynomial<127, 2> { using ZPZ = aerobus::zpz<127>; using type =
POLYV<ZPZV<1>, ZPZV<126>, ZPZV<3»; }; // NOLINT
03223 template<> struct ConwayPolynomial<127, 3> { using ZPZ = aerobus::zpz<127>; using type =
                  POLYV<ZPZV<1>, ZPZV<0>, ZPZV<3>, ZPZV<124»; }; // NOLINT
03224 template<> struct ConwayPolynomial<127, 4> { using ZPZ = aerobus::zpz<127>; using type =
                  POLYV<ZPZV<1>, ZPZV<0>, ZPZV<2>, ZPZV<97>, ZPZV<3»; }; // NOLINT
03225 template<> struct ConwayPolynomial<127, 5> { using ZPZ = aerobus::zpz<127>; using type =
                  POLYV<ZPZV<1>, ZPZV<0>, ZPZV<0>, ZPZV<0>, ZPZV<7>, ZPZV<124»; }; // NOLINT
 03226 template<> struct ConwayPolynomial<127, 6> { using ZPZ = aerobus::zpz<127>; using type =
                   \texttt{POLYV} < \texttt{ZPZV} < 1>, \ \texttt{ZPZV} < 0>, \ \texttt{ZPZV} < 0>, \ \texttt{ZPZV} < 84>, \ \texttt{ZPZV} < 115>, \ \texttt{ZPZV} < 82>, \ \texttt{ZPZV} < 3»; \ \}; \ \ // \ \texttt{NOLINT} 
03227 template<> struct ConwayPolynomial<127, 7> { using ZPZ = aerobus::zpz<127>; using type
                  POLYV<ZPZV<1>, ZPZV<0>, ZPZV<0>, ZPZV<0>, ZPZV<0>, ZPZV<0>, ZPZV<1>, ZPZV<15, ZPZV<124»; }; // NOLINT
 03228 template<> struct ConwayPolynomial<127, 8> { using ZPZ = aerobus::zpz<127>; using type =
                   POLYV<ZPZV<1>, ZPZV<0>, ZPZV<0>, ZPZV<0>, ZPZV<3>, ZPZV<104>, ZPZV<55>, ZPZV<8>, ZPZV<3»; };
 03229 template<> struct ConwayPolynomial<127, 9> { using ZPZ = aerobus::zpz<127>; using type :
                   POLYV<ZPZV<1>, ZPZV<0>, ZPZV<0>, ZPZV<0>, ZPZV<0>, ZPZV<0>, ZPZV<0, ZPZV<14>, ZPZV<119>, ZPZV<126>, ZPZV<124»;
                   }; // NOLINT
 03230 template<> struct ConwayPolynomial<127, 10> { using ZPZ = aerobus::zpz<127>; using type
                   POLYV<ZPZV<1>, ZPZV<0>, ZPZV<0>, ZPZV<0>, ZPZV<0>, ZPZV<107>, ZPZV<64>, ZPZV<64>, ZPZV<65>, ZPZV<60>, ZPZV<4>,
                   ZPZV<3»; }; // NOLINT</pre>
 03231 template<> struct ConwayPolynomial<127, 11> { using ZPZ = aerobus::zpz<127>; using type =
                   POLYV<ZPZV<1>, ZPZV<0>, ZPZV<0
                   ZPZV<11>, ZPZV<124»; }; // NOLINT</pre>
03232 template<> struct ConwayPolynomial<127, 12> { using ZPZ = aerobus::zpz<127>; using type = POLYV<ZPZV<1>, ZPZV<0>, ZPZV<0>, ZPZV<0>, ZPZV<0>, ZPZV<119>, ZPZV<25>, ZPZV<33>, ZPZV<97>, ZPZV<15>,
                   ZPZV<99>, ZPZV<8>, ZPZV<3»; };</pre>
                                                                                                                  // NOLINT
 03233 template<> struct ConwayPolynomial<127, 13> { using ZPZ = aerobus::zpz<127>; using type =
                   POLYV<ZPZV<1>, ZPZV<0>, ZPZV<0
ZPZV<0>, ZPZV<0>, ZPZV<8>, ZPZV<124»; }; // NOLINT
03234 template<> struct ConwayPolynomial<127, 17> { using ZPZ = aerobus::zpz<127>; using type =
                   POLYV<ZPZV<1>, ZPZV<0>, ZPZV<0
                   ZPZV<0>, ZPZV<0>, ZPZV<0>, ZPZV<0>, ZPZV<0>, ZPZV<0>, ZPZV<9>, ZPZV<124»; }; // NOLINT</pre>
03235 template<> struct ConwayPolynomial<127, 19> { using ZPZ = aerobus::zpz<127>; using type =
                   POLYV<ZPZV<1>, ZPZV<0>, ZPZV<0>, ZPZV<0>, ZPZV<0>, ZPZV<0>, ZPZV<0>, ZPZV<0>, ZPZV<0>, ZPZV<0>,
                   ZPZV<0>, ZPZV<0>, ZPZV<0>, ZPZV<0>, ZPZV<0>, ZPZV<0>, ZPZV<12*, }; //</pre>
                   NOLINT
03236 template<> struct ConwayPolynomial<131, 1> { using ZPZ = aerobus::zpz<131>; using type =
                   POLYV<ZPZV<1>, ZPZV<129»; }; // NOLINT
 03237 template<> struct ConwayPolynomial<131, 2> { using ZPZ = aerobus::zpz<131>; using type =
                   POLYV<ZPZV<1>, ZPZV<127>, ZPZV<2»; }; // NOLINT
 03238 template<> struct ConwayPolynomial<131, 3> { using ZPZ = aerobus::zpz<131>; using type =
                  POLYV<ZPZV<1>, ZPZV<0>, ZPZV<3>, ZPZV<129»; }; // NOLINT
 03239 template<> struct ConwayPolynomial<131, 4> { using ZPZ = aerobus::zpz<131>; using type =
 POLYV<ZPZV<1>, ZPZV<0>, ZPZV<10>, ZPZV<20>, ZPZV<20; ; // NOLINT
03240 template<> struct ConwayPolynomial<131, 5> { using ZPZ = aerobus::zpz<131>; using type =
                  POLYV<ZPZV<1>, ZPZV<0>, ZPZV<0>, ZPZV<0>, ZPZV<19>, ZPZV<129»; }; // NOLINT
 03241 template<> struct ConwayPolynomial<131, 6> { using ZPZ = aerobus::zpz<131>; using type =
POLYV<ZPZV<1>, ZPZV<0>, ZPZV<2>, ZPZV<66>, ZPZV<4>, ZPZV<2>, ZPZV<2>, ZPZV<2>, ZPZV<2>, ZPZV<22>, ZPZV<22>, ZPZV<22»; }; // NOLINT 03242 template<> struct ConwayPolynomial<131, 7> { using ZPZ = aerobus::zpz<131>; using type =
```

```
POLYV<ZPZV<1>, ZPZV<0>, ZPZV<0>, ZPZV<0>, ZPZV<0>, ZPZV<0>, ZPZV<10>, ZPZV<10>, ZPZV<129»; };
03243 template<> struct ConwayPolynomial<131, 8> { using ZPZ = aerobus::zpz<131>; using type
                   POLYV<ZPZV<1>, ZPZV<0>, ZPZV<0>, ZPZV<0>, ZPZV<3>, ZPZV<72>, ZPZV<116>, ZPZV<104>, ZPZV<2»; }; //
                   NOLINT
03244 template<> struct ConwayPolynomial<131, 9> { using ZPZ = aerobus::zpz<131>; using type =
                   POLYV<ZPZV<1>, ZPZV<0>, ZPZV<0>, ZPZV<0>, ZPZV<0>, ZPZV<0>, ZPZV<6>, ZPZV<6>, ZPZV<6>, ZPZV<6>, ZPZV<19>, ZPZV<129»; };
 03245 template<> struct ConwayPolynomial<131, 10> { using ZPZ = aerobus::zpz<131>; using type
                   POLYV<ZPZV<1>, ZPZV<0>, ZPZV<0>, ZPZV<0>, ZPZV<0>, ZPZV<4>, ZPZV<9>, ZPZV<9>, ZPZV<944>, ZPZV<2»; }; // NOLINT
03246 template<> struct ConwayPolynomial<131, 11> { using ZPZ = aerobus::zpz<131>; using type =
                   POLYV<ZPZV<1>, ZPZV<0>, ZPZV<0
                   ZPZV<6>, ZPZV<129»; };</pre>
                                                                                                // NOLINT
 03247 template<> struct ConwayPolynomial<131, 12> { using ZPZ = aerobus::zpz<131>; using type =
                   POLÝV<ZPZV<1>, ZPZV<0>, ZPZV<0>, ZPZV<0>, ZPZV<0>, ZPZV<5>, ZPZV<50>, ZPZV<122>, ZPZV<40>, ZPZV<83>, ZPZV<125>,
ZPZV<28>, ZPZV<103>, ZPZV<2»; }; // NOLINT
03248 template<> struct ConwayPolynomial<131, 13> { using ZPZ = aerobus::zpz<131>; using type =
                   POLYVCZPZV<1>, ZPZV<0>, ZPZV<0
03249 template<> struct ConwayPolynomial<131, 17> { using ZPZ = aerobus::zpz<131>; using type
                    POLYV<ZPZV<1>, ZPZV<0>, ZPZV<0
                   03250 template<> struct ConwayPolynomial<131, 19> { using ZPZ = aerobus::zpz<131>; using type =
                   POLYV<ZPZV<1>, ZPZV<0>, ZPZV<0
03251 template<> struct ConwayPolynomial<137, 1> { using ZPZ = aerobus::zpz<137>; using type =
                   POLYV<ZPZV<1>, ZPZV<134»; }; // NOLINT
 03252 template<> struct ConwayPolynomial<137, 2> { using ZPZ = aerobus::zpz<137>; using type =
POLYV<ZPZV<1>, ZPZV<131>, ZPZV<3»; }; // NOLINT

03253 template<> struct ConwayPolynomial<137, 3> { using ZPZ = aerobus::zpz<137>; using type =
                   POLYV<ZPZV<1>, ZPZV<0>, ZPZV<6>, ZPZV<134»; }; // NOLINT
03254 template<> struct ConwayPolynomial<137, 4> { using ZPZ = aerobus::zpz<137>; using type = POLYV<ZPZV<1>, ZPZV<0>, ZPZV<1>, ZPZV<95>, ZPZV<3»; }; // NOLINT
03255 template<> struct ConwayPolynomial<137, 5> { using ZPZ = aerobus::zpz<137>; using type =
                   POLYV<ZPZV<1>, ZPZV<0>, ZPZV<0>, ZPZV<0>, ZPZV<7>, ZPZV<134»; }; // NOLINT
03256 template<> struct ConwayPolynomial<137, 6> { using ZPZ = aerobus::zpz<137>; using type =
                   POLYV<ZPZV<1>, ZPZV<0>, ZPZV<1>, ZPZV<116>, ZPZV<102>, ZPZV<3>, ZPZV<3»; }; // NOLINT
 03257 template<> struct ConwayPolynomial<137, 7> { using ZPZ = aerobus::zpz<137>; using type
                   POLYV<ZPZV<1>, ZPZV<0>, ZPZV<0>, ZPZV<0>, ZPZV<0>, ZPZV<0>, ZPZV<1>, ZPZV<134»; };
03258 template<> struct ConwayPolynomial<137, 8> { using ZPZ = aerobus::zpz<137>; using type =
                   POLYV<ZPZV<1>, ZPZV<0>, ZPZV<0>, ZPZV<0>, ZPZV<4>, ZPZV<105>, ZPZV<21>, ZPZV<34>, ZPZV<3*; }; //
                   NOLINT
03259 template<> struct ConwayPolynomial<137, 9> { using ZPZ = aerobus::zpz<137>; using type =
                   POLYV<ZPZV<1>, ZPZV<0>, ZPZV<0>, ZPZV<0>, ZPZV<0>, ZPZV<0>, ZPZV<1>, ZPZV<1>, ZPZV<80>, ZPZV<122>, ZPZV<134»;
                     }; // NOLINT
03260 template<> struct ConwayPolynomial<137, 10> { using ZPZ = aerobus::zpz<137>; using type =
                   POLYV<ZPZV<1>, ZPZV<0>, ZPZV<0>, ZPZV<0>, ZPZV<2>, ZPZV<2>, ZPZV<20>, ZPZV<67>, ZPZV<93>, ZPZV<119>,
                   ZPZV<3»: }: // NOLINT
03261 template<> struct ConwayPolynomial<137, 11> { using ZPZ = aerobus::zpz<137>; using type = POLYV<ZPZV<1>, ZPZV<0>, ZPZV<1>, ZPZV<134»; }; // NOLINT
 03262 template<> struct ConwayPolynomial<137, 12> { using ZPZ = aerobus::zpz<137>; using type =
                   POLYV<ZPZV<1>, ZPZV<0>, ZPZV<0>, ZPZV<0>, ZPZV<1>, ZPZV<61>, ZPZV<40>, ZPZV<40>, ZPZV<36>, ZPZV<36>, ZPZV<135>, ZPZV<61>, ZPZV<61>, ZPZV<3»; }; // NOLINT
 03263 template<> struct ConwayPolynomial<137, 13> { using ZPZ = aerobus::zpz<137>; using type
                    POLYV<ZPZV<1>, ZPZV<0>, ZPZV<0
                    ZPZV<0>, ZPZV<0>, ZPZV<14>, ZPZV<134»; }; // NOLINT</pre>
 03264 template<> struct ConwayPolynomial<137, 17> { using ZPZ = aerobus::zpz<137>; using type =
                   POLYV<ZPZV<1>, ZPZV<0>, ZPZV<0>, ZPZV<0>, ZPZV<0>, ZPZV<0>, ZPZV<0>, ZPZV<0>, ZPZV<0>, ZPZV<0>,
ZPZV<0>, ZPZV<0>, ZPZV<0>, ZPZV<0>, ZPZV<0>, ZPZV<0>, ZPZV<0>, ZPZV<136>, ZPZV<4>, ZPZV<134»; }; // NOLINT

03265 template<> struct ConwayPolynomial<137, 19> { using ZPZ = aerobus::zpz<137>; using type = POLYV<ZPZV<1>, ZPZV<0>, ZPZV<0 , 
                    ZPZV<0>, ZPZV<0>, ZPZV<0>, ZPZV<0>, ZPZV<0>, ZPZV<0>, ZPZV<134»; }; //</pre>
                   NOLTNT
03266 template<> struct ConwayPolynomial<139, 1> { using ZPZ = aerobus::zpz<139>; using type =
                   POLYV<ZPZV<1>, ZPZV<137»; }; // NOLINT
03267 template<> struct ConwayPolynomial<139, 2> { using ZPZ = aerobus::zpz<139>; using type =
                   POLYV<ZPZV<1>, ZPZV<138>, ZPZV<2»; }; // NOLINT
 03268 template<> struct ConwayPolynomial<139, 3> { using ZPZ = aerobus::zpz<139>; using type =
                   POLYV<ZPZV<1>, ZPZV<0>, ZPZV<6>, ZPZV<137»; }; // NOLINT
 03269 template<> struct ConwayPolynomial<139, 4> { using ZPZ = aerobus::zpz<139>; using type =
POLYV<ZPZV<1>, ZPZV<0>, ZPZV<7>, ZPZV<96>, ZPZV<2»; }; // NOLINT
03270 template<> struct ConwayPolynomial<139, 5> { using ZPZ = aerobus::zpz<139>; using type =
 POLYV<ZPZV<1>, ZPZV<0>, ZPZV<0>, ZPZV<0>, ZPZV<10>, ZPZV<103, ZPZV<137»; }; // NOLINT
03271 template<> struct ConwayPolynomial<139, 6> { using ZPZ = aerobus::zpz<139>; using type =
                   POLYV<ZPZV<1>, ZPZV<0>, ZPZV<4>, ZPZV<46>, ZPZV<10>, ZPZV<118>, ZPZV<2»; }; // NOLINT
 03272 template<> struct ConwayPolynomial<139, 7> { using ZPZ = aerobus::zpz<139>; using type =
                   POLYV<ZPZV<1>, ZPZV<0>, ZPZV<0>, ZPZV<0>, ZPZV<0>, ZPZV<0>, ZPZV<0>, ZPZV<0>, ZPZV<9>, ZPZV<137»; };
                                                                                                                                                                                                                                                                                             // NOLINT
 03273 template<> struct ConwayPolynomial<139, 8> { using ZPZ = aerobus::zpz<139>; using type =
                   POLYV<ZPZV<1>, ZPZV<0>, ZPZV<0>, ZPZV<0>, ZPZV<4>, ZPZV<103>, ZPZV<36>, ZPZV<21>, ZPZV<2»; }; //
 03274 template<> struct ConwayPolynomial<139, 9> { using ZPZ = aerobus::zpz<139>; using type =
                   POLYV<ZPZV<1>, ZPZV<0>, ZPZV<0>, ZPZV<0>, ZPZV<0>, ZPZV<0>, ZPZV<3>, ZPZV<70>, ZPZV<70>, ZPZV<87>, ZPZV<137»; };
                     // NOLINT
03275 template<> struct ConwayPolynomial<139, 10> { using ZPZ = aerobus::zpz<139>; using type = POLYV<ZPZV<1>, ZPZV<0>, ZPZV<0>, ZPZV<0>, ZPZV<0>, ZPZV<110>, ZPZV<48>, ZPZV<130>, ZPZV<66>,
```

```
ZPZV<106>, ZPZV<2»; };</pre>
 03276 template<> struct ConwayPolynomial<139, 11> { using ZPZ = aerobus::zpz<139>; using type =
                        POLYV<ZPZV<1>, ZPZV<0>, ZPZV<137»; }; // NOLINT
03277 template<> struct ConwayPolynomial<139, 12> { using ZPZ = aerobus::zpz<139>; using type = POLYV<ZPZV<1>, ZPZV<0>, ZPZV<0>, ZPZV<0>, ZPZV<0>, ZPZV<120>, ZPZV<75>, ZPZV<41>, ZPZV<77>, ZPZV<106>, ZPZV<8>, ZPZV<8>, ZPZV<10>, ZPZV<2»; }; // NOLINT
 03278 template<> struct ConwayPolynomial<139, 13> { using ZPZ = aerobus::zpz<139>; using type
                         POLYV<ZPZV<1>, ZPZV<0>, ZPZV<0
ZPZV<0>, ZPZV<0>, ZPZV<22>, ZPZV<137»; }; // NOLINT
03279 template<> struct ConwayPolynomial<139, 17> { using ZPZ = aerobus::zpz<139>; using type =
                        POLYV<ZPZV<1>, ZPZV<0>, ZPZV<0
ZPZV<0>, ZPZV<0>, ZPZV<0>, ZPZV<0>, ZPZV<0>, ZPZV<0>, ZPZV<0>, ZPZV<0>, ZPZV<2>, ZPZV<2>, ZPZV<2>, ZPZV<2>, ZPZV<23>; is in type =
                         POLYV<ZPZV<1>, ZPZV<0>, ZPZV<0
                         ZPZV<0>, ZPZV<0>, ZPZV<0>, ZPZV<0>, ZPZV<0>, ZPZV<0>, ZPZV<0>, ZPZV<0>, ZPZV<23>, ZPZV<137»; }; //</pre>
                         NOLTNT
03281 template<> struct ConwayPolynomial<149, 1> { using ZPZ = aerobus::zpz<149>; using type =
                         POLYV<ZPZV<1>, ZPZV<147»; }; // NOLINT
03282 template<> struct ConwayPolynomial<149, 2> { using ZPZ = aerobus::zpz<149>; using type =
                         POLYV<ZPZV<1>, ZPZV<145>, ZPZV<2»; }; // NOLINT
 03283 template<> struct ConwayPolynomial<149, 3> { using ZPZ = aerobus::zpz<149>; using type =
                        POLYV<ZPZV<1>, ZPZV<0>, ZPZV<3>, ZPZV<147»; }; // NOLINT
03284 template<> struct ConwayPolynomial<149, 4> { using ZPZ = aerobus::zpz<149>; using type = POLYV<ZPZV<1>, ZPZV<0>, ZPZV<7>, ZPZV<107>, ZPZV<2»; }; // NOLINT
03285 template<> struct ConwayPolynomial<149, 5> { using ZPZ = aerobus::zpz<149>; using type =
                         POLYV<ZPZV<1>, ZPZV<0>, ZPZV<0>, ZPZV<0>, ZPZV<2>, ZPZV<147»; }; // NOLINT
 03286 template<> struct ConwayPolynomial<149, 6> { using ZPZ = aerobus::zpz<149>; using type =
POLYV<ZPZV<1>, ZPZV<0>, ZPZV<1>, ZPZV<105>, ZPZV<33>, ZPZV<55>, ZPZV<2»; }; // NOLINT
03287 template<> struct ConwayPolynomial<149, 7> { using ZPZ = aerobus::zpz<149>; using type =
POLYV<ZPZV<1>, ZPZV<0>, ZPZV<0>, ZPZV<0>, ZPZV<0>, ZPZV<0>, ZPZV<19>, ZPZV<147»; }; // NOLINT
 03288 template<> struct ConwayPolynomial<149, 8> { using ZPZ = aerobus::zpz<149>; using type
                         POLYV<ZPZV<1>, ZPZV<0>, ZPZV<0>, ZPZV<0>, ZPZV<2>, ZPZV<140>, ZPZV<25>, ZPZV<123>, ZPZV<22»; }; //
                         NOLINT
03289 template<> struct ConwayPolynomial<149, 9> { using ZPZ = aerobus::zpz<149>; using type = POLYV<ZPZV<1>, ZPZV<0>, ZPZV<0>, ZPZV<0>, ZPZV<0>, ZPZV<0>, ZPZV<146>, ZPZV<20>, ZPZV<147»;
                         }; // NOLINT
 03290 template<> struct ConwayPolynomial<149, 10> { using ZPZ = aerobus::zpz<149>; using type
                         POLYV<ZPZV<1>, ZPZV<0>, ZPZV<0>, ZPZV<0>, ZPZV<0>, ZPZV<0>, ZPZV<74>, ZPZV<442>, ZPZV<148>, ZPZV<143>, ZPZV<51>,
                         ZPZV<2»; }; // NOLINT</pre>
03291 template<> struct ConwayPolynomial<149, 11> { using ZPZ = aerobus::zpz<149>; using type =
                        POLYV<ZPZV<1>, ZPZV<0>, ZPZV<0
03292 template<> struct ConwayPolynomial<149, 12> { using ZPZ = aerobus::zpz<149>; using type =
                         POLYV<ZPZV<1>, ZPZV<0>, ZPZV<0>, ZPZV<0>, ZPZV<0>, ZPZV<0>, ZPZV<0>, ZPZV<121>, ZPZV<91>, ZPZV<91>, ZPZV<9>,
                         ZPZV<104>, ZPZV<110>, ZPZV<2»; };  // NOLINT</pre>
03293 template<> struct ConwayPolynomial<149, 13> { using ZPZ = aerobus::zpz<149>; using type =
                        POLYV<ZPZV<1>, ZPZV<0>, ZPZV<0
03294 template<> struct ConwayPolynomial<149, 17> { using ZPZ = aerobus::zpz<149>; using type = POLYV<ZPZV<1>, ZPZV<0>, ZPZV<0 , 
                         ZPZV<0>, ZPZV<0>, ZPZV<0>, ZPZV<0>, ZPZV<0>, ZPZV<0>, ZPZV<29>, ZPZV<147»; }; // NOLINT</pre>
03295 template<> struct ConwayPolynomial<149, 19> { using ZPZ = aerobus::zpz<149>; using type =
                        POLYV<ZPZV<1>, ZPZV<0>, ZPZV<0
                         NOLINT
 03296 template<> struct ConwayPolynomial<151, 1> { using ZPZ = aerobus::zpz<151>; using type =
                         POLYV<ZPZV<1>, ZPZV<145»; }; // NOLINT
 03297 template<> struct ConwayPolynomial<151, 2> { using ZPZ = aerobus::zpz<151>; using type =
                        POLYV<ZPZV<1>, ZPZV<149>, ZPZV<6»; };
                                                                                                                                                                                  // NOLINT
03298 template<> struct ConwayPolynomial<151, 3> { using ZPZ = aerobus::zpz<151>; using type =
                        POLYV<ZPZV<1>, ZPZV<0>, ZPZV<1>, ZPZV<145»; }; // NOLINT
 03299 template<> struct ConwayPolynomial<151, 4> { using ZPZ = aerobus::zpz<151>; using type =
POLYV<ZPZV<1>, ZPZV<0>, ZPZV<13>, ZPZV<89>, ZPZV<6»; }; // NOLINT
03300 template<> struct ConwayPolynomial<151, 5> { using ZPZ = aerobus::zpz<151>; using type =
                        POLYV<ZPZV<1>, ZPZV<0>, ZPZV<0>, ZPZV<0>, ZPZV<11>, ZPZV<145»; }; // NOLINT
 03301 template<> struct ConwayPolynomial<151, 6> { using ZPZ = aerobus::zpz<151>; using type =
                        POLYV<ZPZV<1>, ZPZV<0>, ZPZV<0>, ZPZV<125>, ZPZV<18>, ZPZV<15>, ZPZV<6»; }; // NOLINT
 03302 template<> struct ConwayPolynomial<151, 7> { using ZPZ = aerobus::zpz<151>; using type
                        POLYV<ZPZV<1>, ZPZV<0>, ZPZV<0>, ZPZV<0>, ZPZV<0>, ZPZV<0>, ZPZV<0>, ZPZV<0>, ZPZV<9>, ZPZV<145»; };
 03303 template<> struct ConwayPolynomial<151, 8> { using ZPZ = aerobus::zpz<151>; using type =
                         POLYV<ZPZV<1>, ZPZV<0>, ZPZV<0>, ZPZV<0>, ZPZV<9>, ZPZV<140>, ZPZV<122>, ZPZV<43>, ZPZV<6»; }; //
                         NOLINT
03304 template<> struct ConwayPolynomial<151, 9> { using ZPZ = aerobus::zpz<151>; using type =
                         POLYV<ZPZV<1>, ZPZV<0>, ZPZV<0>, ZPZV<0>, ZPZV<0>, ZPZV<0>, ZPZV<6>, ZPZV<6>, ZPZV<126>, ZPZV<196>, ZPZV<145»;
 03305 template<> struct ConwayPolynomial<151, 10> { using ZPZ = aerobus::zpz<151>; using type
                        POLYV<ZPZV<1>, ZPZV<0>, ZPZV<0>, ZPZV<0>, ZPZV<1>, ZPZV<21>, ZPZV<104>, ZPZV<49>, ZPZV<20>, ZPZV<142>, ZPZV<6»; }; // NOLINT
03306 template<> struct ConwayPolynomial<151, 11> { using ZPZ = aerobus::zpz<151>; using type =
                        POLYV<ZPZV<1>, ZPZV<0>, ZPZV<0
                         ZPZV<1>, ZPZV<145»; };</pre>
                                                                                                                         // NOLINT
 03307 template<> struct ConwayPolynomial<151, 12> { using ZPZ = aerobus::zpz<151>; using type
                         POLYV<ZPZV<1>, ZPZV<0>, ZPZV<0>, ZPZV<0>, ZPZV<1>, ZPZV<109>, ZPZV<121>, ZPZV<101>, ZPZV<101>, ZPZV<6>, ZPZV<77>,
ZPZV<107>, ZPZV<147>, ZPZV<6»; }; // NOLINT
03308 template<> struct ConwayPolynomial<151, 13> { using ZPZ = aerobus::zpz<151>; using type =
POLYV<ZPZV<1>, ZPZV<0>, ZPZV<0>,
```

```
ZPZV<0>, ZPZV<0>, ZPZV<12>, ZPZV<145»; };</pre>
03309 template<> struct ConwayPolynomial<151, 17> { using ZPZ = aerobus::zpz<151>; using type =
                                POLYV<ZPZV<1>, ZPZV<0>, ZPZV<0
03310 template<> struct ConwayPolynomial<151, 19> { using ZPZ = aerobus::zpz<151>; using type = POLYV<ZPZV<1>, ZPZV<0>, 
                                  ZPZV<0>, ZPZV<0>, ZPZV<0>, ZPZV<0>, ZPZV<0>, ZPZV<0>, ZPZV<0>, ZPZV<0>, ZPZV<0>, ZPZV<0>, ZPZV<0>, ZPZV<0>, ZPZV<0>, ZPZV<0>, ZPZV<0>, ZPZV<0>, ZPZV<0>, ZPZV<0>, ZPZV<0>, ZPZV<0>, ZPZV<0>, ZPZV<0>, ZPZV<0>, ZPZV<0>, ZPZV<0>, ZPZV<0>, ZPZV<0>, ZPZV<0>, ZPZV<0>, ZPZV<0>, ZPZV<0>, ZPZV<0>, ZPZV<0>, ZPZV<0>, ZPZV<0>, ZPZV<0>, ZPZV<0>, ZPZV<0>, ZPZV<0>, ZPZV<0>, ZPZV<0>, ZPZV<0>, ZPZV<0>, ZPZV<0>, ZPZV<0>, ZPZV<0>, ZPZV<0>, ZPZV<0>, ZPZV<0>, ZPZV<0>, ZPZV<0>, ZPZV<0>, ZPZV<0>, ZPZV<0>, ZPZV<0>, ZPZV<0>, ZPZV<0>, ZPZV<0>, ZPZV<0>, ZPZV<0>, ZPZV<0>, ZPZV<0>, ZPZV<0>, ZPZV<0>, ZPZV<0>, ZPZV<0>, ZPZV<0>, ZPZV<0>, ZPZV<0>, ZPZV<0>, ZPZV<0>, ZPZV<0>, ZPZV<0>, ZPZV<0>, ZPZV<0>, ZPZV<0>, ZPZV<0>, ZPZV<0>, ZPZV<0>, ZPZV<0>, ZPZV<0>, ZPZV<0>, ZPZV<0>, ZPZV<0>, ZPZV<0>, ZPZV<0>, ZPZV<0>, ZPZV<0>, ZPZV<0>, ZPZV<0>, ZPZV<0>, ZPZV<0>, ZPZV<0>, ZPZV<0>, ZPZV<0>, ZPZV<0>, ZPZV<0>, ZPZV<0>, ZPZV<0>, ZPZV<0>, ZPZV<0>, ZPZV<0>, ZPZV<0>, ZPZV<0>, ZPZV<0>, ZPZV<0>, ZPZV<0>, ZPZV<0>, ZPZV<0>, ZPZV<0>, ZPZV<0>, ZPZV<0>, ZPZV<0>, ZPZV<0>, ZPZV<0>, ZPZV<0>, ZPZV<0>, ZPZV<0>, ZPZV<0>, ZPZV<0>, ZPZV<0>, ZPZV<0>, ZPZV<0>, ZPZV<0>, ZPZV<0>, ZPZV<0>, ZPZV<0>, ZPZV<0>, ZPZV<0>, ZPZV<0>, ZPZV<0>, ZPZV<0>, ZPZV<0>, ZPZV<0>, ZPZV<0>, ZPZV<0>, ZPZV<0>, ZPZV<0>, ZPZV<0>, ZPZV<0>, ZPZV<0>, ZPZV<0>, ZPZV<0>, ZPZV<0>, ZPZV<0>, ZPZV<0>, ZPZV<0>, ZPZV<0>, ZPZV<0>, ZPZV<0>, ZPZV<0>, ZPZV<0>, ZPZV<0>, ZPZV<0>, ZPZV<0>, ZPZV<0>, ZPZV<0>, ZPZV<0>, ZPZV<0>, ZPZV<0>, ZPZV<0>, ZPZV<0>, ZPZV<0>, ZPZV<0>, ZPZV<0>, ZPZV<0>, ZPZV<0>, ZPZV<0>, ZPZV<0>, ZPZV<0>, ZPZV<0>, ZPZV<0>, ZPZV<0>, ZPZV<0>, ZPZV<0>, ZPZV<0>, ZPZV<0>, ZPZV<0>, ZPZV<0>, ZPZV<0>, ZPZV<0>, ZPZV<0>, ZPZV<0>, ZPZV<0>, ZPZV<0>, ZPZV<0>, ZPZV<0>, ZPZV<0>, ZPZV<0>, ZPZV<0>, ZPZV<0>, ZPZV<0>, ZPZV<0>, ZPZV<0>, ZPZV<0>, ZPZV<0>, ZPZV<0>, ZPZV<0>, ZPZV<0>, ZPZV<0>, ZPZV<0>, ZPZV<0>, ZPZV<0>, ZPZV<0>, ZPZV<0>, ZPZV<0>, ZPZV<0>, ZPZV<0>, ZPZV<0>, ZPZV<0>, ZPZV<0>, ZPZV<0>, ZPZV<0>, ZPZV<0>, ZPZV<0>, ZPZV<0>, ZPZV<0>, ZPZV<0>, ZPZV<0>, ZPZV<0>, ZPZV<0>, ZPZV<0>, ZPZV<0>, ZPZV<0>, ZPZV<0>, ZPZV<0>, Z
03311 template<> struct ConwayPolynomial<157, 1> { using ZPZ = aerobus::zpz<157>; using type =
                                POLYV<ZPZV<1>, ZPZV<152»; }; // NOLINT
03312 template<> struct ConwayPolynomial<157, 2> { using ZPZ = aerobus::zpz<157>; using type =
POLYV<ZPZV<1>, ZPZV<152>, ZPZV<5»; }; // NOLINT
03313 template<> struct ConwayPolynomial<157, 3> { using ZPZ = aerobus::zpz<157>; using type =
                                 POLYV<ZPZV<1>, ZPZV<0>, ZPZV<1>, ZPZV<152»; }; // NOLINT
 03314 template<> struct ConwayPolynomial<157, 4> { using ZPZ = aerobus::zpz<157>; using type =
POLYV<ZPZV<1>, ZPZV<0>, ZPZV<11>, ZPZV<136>, ZPZV<5»; }; // NOLINT
03315 template<> struct ConwayPolynomial<157, 5> { using ZPZ = aerobus::zpz<157>; using type =
                                POLYV<ZPZV<1>, ZPZV<0>, ZPZV<0>, ZPZV<7>, ZPZV<152»; }; // NOLINT
 03316 template<> struct ConwayPolynomial<157, 6> { using ZPZ = aerobus::zpz<157>; using type =
                                 POLYV<ZPZV<1>, ZPZV<0>, ZPZV<3>, ZPZV<130>, ZPZV<44>, ZPZV<144>, ZPZV<5»; }; // NOLINT
 03317 template<> struct ConwayPolynomial<157, 7> { using ZPZ = aerobus::zpz<157>; using type =
                                POLYV<ZPZV<1>, ZPZV<0>, ZPZV<0>, ZPZV<0>, ZPZV<0>, ZPZV<0>, ZPZV<14>, ZPZV<145, ZPZV<152»; };
03318 template<> struct ConwayPolynomial<157, 8> { using ZPZ = aerobus::zpz<157>; using type =
                                POLYV<ZPZV<1>, ZPZV<0>, ZPZV<0>, ZPZV<3>, ZPZV<97>, ZPZV<40>, ZPZV<153>, ZPZV<5»; }; //
                                 NOLINT
03319 template<> struct ConwayPolynomial<157, 9> { using ZPZ = aerobus::zpz<157>; using type :
                                 POLYV<ZPZV<1>, ZPZV<0>, ZPZV<0>, ZPZV<0>, ZPZV<0>, ZPZV<0>, ZPZV<0>, ZPZV<2>, ZPZV<114>, ZPZV<52>, ZPZV<152»;
                                   }; // NOLINT
 03320 template<> struct ConwayPolynomial<157, 10> { using ZPZ = aerobus::zpz<157>; using type
                                POLYV<ZPZV<1>, ZPZV<0>, ZPZV<0>, ZPZV<0>, ZPZV<61>, ZPZV<22>, ZPZV<124>, ZPZV<61>, ZPZV<93>, ZPZV<5»; }; // NOLINT
03321 template<> struct ConwayPolynomial<157, 11> { using ZPZ = aerobus::zpz<157>; using type
                                 POLYV<ZPZV<1>, ZPZV<0>, ZPZV<0
03322 template<> struct ConwayPolynomial<157, 12> { using ZPZ = aerobus::zpz<157>; using type = POLYV<ZPZV<1>, ZPZV<0>, ZPZV<0>, ZPZV<0>, ZPZV<0>, ZPZV<77>, ZPZV<110>, ZPZV<72>, ZPZV<137>, ZPZV<43>,
                                 ZPZV<152>, ZPZV<57>, ZPZV<5»; }; // NOLINT</pre>
 03323 template<> struct ConwayPolynomial<157, 13> { using ZPZ = aerobus::zpz<157>; using type
                                  POLYV<ZPZV<1>, ZPZV<0>, ZPZV<0
ZPZV<0>, ZPZV<156>, ZPZV<9>, ZPZV<152»; }; // NOLINT
03324 template<> struct ConwayPolynomial<157, 17> { using ZPZ = aerobus::zpz<157>; using type =
POLYV<ZPZV<1>, ZPZV<0>, ZPZV<0>; ZPZV<0>; ZPZV<0>, ZPZV<0>; ZPZV<0>, ZPZV<0>; ZPZV<0
                                  POLYV<ZPZV<1>, ZPZV<0>, ZPZV<0
                                  ZPZV<0>, ZPZV<0>, ZPZV<0>, ZPZV<0>, ZPZV<0>, ZPZV<0>, ZPZV<0>, ZPZV<14>, ZPZV<152»; }; //</pre>
03326 template<> struct ConwayPolynomial<163, 1> { using ZPZ = aerobus::zpz<163>; using type =
                                POLYV<ZPZV<1>, ZPZV<161»; }; // NOLINT
03327 template<> struct ConwayPolynomial<163, 2> { using ZPZ = aerobus::zpz<163>; using type =
                                 POLYV<ZPZV<1>, ZPZV<159>, ZPZV<2»; }; // NOLINT
 03328 template<> struct ConwayPolynomial<163, 3> { using ZPZ = aerobus::zpz<163>; using type =
                                POLYV<ZPZV<1>, ZPZV<0>, ZPZV<7>, ZPZV<161»; }; // NOLINT
03329 template<> struct ConwayPolynomial<163, 4> { using ZPZ = aerobus::zpz<163>; using type = POLYV<ZPZV<1>, ZPZV<0>, ZPZV<8>, ZPZV<91>, ZPZV<2»; }; // NOLINT
03330 template<> struct ConwayPolynomial<163, 5> { using ZPZ = aerobus::zpz<163>; using type =
                                 POLYV<ZPZV<1>, ZPZV<0>, ZPZV<0>, ZPZV<0>, ZPZV<3>, ZPZV<161»; }; // NOLINT
 03331 template<> struct ConwayPolynomial<163, 6> { using ZPZ = aerobus::zpz<163>; using type =
                                 POLYV<ZPZV<1>, ZPZV<0>, ZPZV<0>, ZPZV<83>, ZPZV<25>, ZPZV<156>, ZPZV<2»; }; // NOLINT
 03332 template<> struct ConwayPolynomial<163, 7> { using ZPZ = aerobus::zpz<163>; using type :
                                POLYV<ZPZV<1>, ZPZV<0>, ZPZV<0>, ZPZV<0>, ZPZV<0>, ZPZV<0>, ZPZV<0>, ZPZV<9>, ZPZV<161»; };
03333 template<> struct ConwayPolynomial<163, 8> { using ZPZ = aerobus::zpz<163>; using type = POLYV<ZPZV<1>, ZPZV<0>, ZPZV<0>, ZPZV<0>, ZPZV<1>, ZPZV<132>, ZPZV<83>, ZPZV<6>, ZPZV<2»; };
03334 template<> struct ConwayPolynomial<163, 9> { using ZPZ = aerobus::zpz<163>; using type =
                                 POLYV<ZPZV<1>, ZPZV<0>, ZPZV<0>, ZPZV<0>, ZPZV<0>, ZPZV<0>, ZPZV<15>, ZPZV<162>, ZPZV<127>, ZPZV<161»;
                                   }; // NOLINT
03335 template<> struct ConwayPolynomial<163, 10> { using ZPZ = aerobus::zpz<163>; using type = POLYV<ZPZV<1>, ZPZV<0>, ZPZV<0>, ZPZV<0>, ZPZV<3>, ZPZV<111>, ZPZV<120>, ZPZV<125>, ZPZV<15>, ZPZV<0>,
                                  ZPZV<2»; }; // NOLINT</pre>
03336 template<> struct ConwayPolynomial<163, 11> { using ZPZ = aerobus::zpz<163>; using type = POLYV<ZPZV<1>, ZPZV<0>, 
                                 ZPZV<11>, ZPZV<161»; }; // NOLINT</pre>
03337 template<> struct ConwayPolynomial<163, 12> { using ZPZ = aerobus::zpz<163>; using type = POLYV<ZPZV<1>, ZPZV<0>, ZPZV<0>, ZPZV<0>, ZPZV<0>, ZPZV<39>, ZPZV<112>, ZPZV<31>, ZPZV<38>, ZPZV<103>, ZPZV<69>, ZPZV<2»; }; // NOLINT
 03338 template<> struct ConwayPolynomial<163, 13> { using ZPZ = aerobus::zpz<163>; using type
POLYV<ZPZV<1>, ZPZV<0>, ZPZV<0
                                 POLYV<ZPZV<1>, ZPZV<0>, ZPZV<0
                                  ZPZV<0>, ZPZV<0>, ZPZV<0>, ZPZV<0>, ZPZV<0>, ZPZV<0>, ZPZV<71>, ZPZV<761>, ZPZV<161»; }; // NOLINT</pre>
 03340 template<> struct ConwayPolynomial<163, 19> { using ZPZ = aerobus::zpz<163>; using type =
                                 POLYV<ZPZV<1>, ZPZV<0>, ZPZV<0>, ZPZV<0>, ZPZV<0>, ZPZV<0>, ZPZV<0>, ZPZV<0>, ZPZV<0>, ZPZV<0>, ZPZV<0>,
                                  \texttt{ZPZV} < \texttt{0} >, \ \texttt{Z
                                 NOLINT
 03341 template<> struct ConwayPolynomial<167, 1> { using ZPZ = aerobus::zpz<167>; using type =
```

```
POLYV<ZPZV<1>, ZPZV<162»; }; // NOLINT
03342 template<> struct ConwayPolynomial<167, 2> { using ZPZ = aerobus::zpz<167>; using type = POLYV<ZPZV<1>, ZPZV<166>, ZPZV<5»; }; // NOLINT
03343 template<> struct ConwayPolynomial<167, 3> { using ZPZ = aerobus::zpz<167>; using type =
POLYY<ZPZV<1>, ZPZV<0>, ZPZV<7>, ZPZV<162»; }; // NOLINT
03344 template<> struct ConwayPolynomial<167, 4> { using ZPZ = aerobus::zpz<167>; using type =
POLYV<ZPZV<1>, ZPZV<0>, ZPZV<3>, ZPZV<120, ZPZV<5>; // NOLINT

03345 template<> struct ConwayPolynomial<167, 5> { using ZPZ = aerobus::zpz<167>; using type =
                    POLYV<ZPZV<1>, ZPZV<0>, ZPZV<0>, ZPZV<0>, ZPZV<3>, ZPZV<162»; }; // NOLINT
03346 template<> struct ConwayPolynomial<167, 6> { using ZPZ = aerobus::zpz<167>; using type = POLYV<ZPZV<1>, ZPZV<0>, ZPZV<2>, ZPZV<75>, ZPZV<38>, ZPZV<2>, ZPZV<5»; }; // NOLINT
03347 template<> struct ConwayPolynomial<167, 7> { using ZPZ = aerobus::zpz<167>; using type =
                    POLYV<ZPZV<1>, ZPZV<0>, ZPZV<0>, ZPZV<0>, ZPZV<0>, ZPZV<0>, ZPZV<10>, ZPZV<162»; }; // NOLINT
 03348 template<> struct ConwayPolynomial<167, 8> { using ZPZ = aerobus::zpz<167>; using type =
                     POLYV<ZPZV<1>, ZPZV<0>, ZPZV<0>, ZPZV<0>, ZPZV<2>, ZPZV<149>, ZPZV<56>, ZPZV<113>, ZPZV<5»; }; //
03349 template<> struct ConwayPolynomial<167, 9> { using ZPZ = aerobus::zpz<167>; using type =
                    POLYV<ZPZV<1>, ZPZV<0>, ZPZV<0>, ZPZV<0>, ZPZV<0>, ZPZV<0>, ZPZV<0>, ZPZV<165>, ZPZV<165>, ZPZV<162»;
 03350 template<> struct ConwayPolynomial<167, 10> { using ZPZ = aerobus::zpz<167>; using type
                    POLYV<ZPZV<1>, ZPZV<0>, ZPZV<0>, ZPZV<0>, ZPZV<0>, ZPZV<85>, ZPZV<68>, ZPZV<109>, ZPZV<143>, ZPZV<148>, ZPZV<5»; }; // NOLINT
03351 template<> struct ConwayPolynomial<167, 11> { using ZPZ = aerobus::zpz<167>; using type =
                     POLYV<ZPZV<1>, ZPZV<0>, ZPZV<0
                     ZPZV<24>, ZPZV<162»; };</pre>
                                                                                                       // NOLINT
03352 template<> struct ConwayPolynomial<167, 12> { using ZPZ = aerobus::zpz<167>; using type = POLYV<ZPZV<1>, ZPZV<0>, ZPZV<0>, ZPZV<0>, ZPZV<0>, ZPZV<142>, ZPZV<10>, ZPZV<142>, ZPZV<142
ZPZV<140>, ZPZV<41>, ZPZV<57>, ZPZV<5»; }; // NOLINT
03353 template<> struct ConwayPolynomial<167, 13> { using ZPZ = aerobus::zpz<167>; using type
POLYV<ZPZV<1>, ZPZV<0>, ZPZV<0
                     POLYV<ZPZV<1>, ZPZV<0>, ZPZV<0>, ZPZV<0>, ZPZV<0>, ZPZV<0>, ZPZV<0>, ZPZV<0>, ZPZV<0>, ZPZV<0>, ZPZV<0>,
                     ZPZV<0>, ZPZV<0>, ZPZV<0>, ZPZV<0>, ZPZV<0>, ZPZV<0>, ZPZV<32>, ZPZV<162»; }; // NOLINT</pre>
03355 template<> struct ConwayPolynomial<167, 19> { using ZPZ = aerobus::zpz<167>; using type = POLYV<ZPZV<1>, ZPZV<0>, 
                     ZPZV<0>, ZPZV<0>, ZPZV<0>, ZPZV<0>, ZPZV<0>, ZPZV<0>, ZPZV<0>, ZPZV<0>, ZPZV<0>, ZPZV<14>, ZPZV<162»; }; //</pre>
03356 template<> struct ConwayPolynomial<173, 1> { using ZPZ = aerobus::zpz<173>; using type =
                     POLYV<ZPZV<1>, ZPZV<171»; }; // NOLINT
 03357 template<> struct ConwayPolynomial<173, 2> { using ZPZ = aerobus::zpz<173>; using type =
POLYV<ZPZV<1>, ZPZV<169, ZPZV<2»; }; // NOLINT
03358 template<> struct ConwayPolynomial<173, 3> { using ZPZ = aerobus::zpz<173>; using type =
                    POLYV<ZPZV<1>, ZPZV<0>, ZPZV<2>, ZPZV<171»; }; // NOLINT
 03359 template<> struct ConwayPolynomial<173, 4> { using ZPZ = aerobus::zpz<173>; using type =
                    POLYV<ZPZV<1>, ZPZV<0>, ZPZV<1>, ZPZV<102>, ZPZV<2»; }; // NOLINT
 03360 template<> struct ConwayPolynomial<173, 5> { using ZPZ = aerobus::zpz<173>; using type =
                    POLYV<ZPZV<1>, ZPZV<0>, ZPZV<0>, ZPZV<6>, ZPZV<171»; }; // NOLINT
03361 template<> struct ConwayPolynomial<173, 6> { using ZPZ = aerobus::zpz<173>; using type =
                    POLYV<ZPZV<1>, ZPZV<0>, ZPZV<1>, ZPZV<27>, ZPZV<134>, ZPZV<107>, ZPZV<2»; }; // NOLINT
03362 template<> struct ConwayPolynomial<173, 7> { using ZPZ = aerobus::zpz<173>; using type
                     POLYV<ZPZV<1>, ZPZV<0>, ZPZV<0>, ZPZV<0>, ZPZV<0>, ZPZV<0>, ZPZV<5>, ZPZV<5>, ZPZV<171»; };
 03363 template<> struct ConwayPolynomial<173, 8> { using ZPZ = aerobus::zpz<173>; using type =
                     POLYV<ZPZV<1>, ZPZV<0>, ZPZV<0>, ZPZV<0>, ZPZV<2>, ZPZV<125>, ZPZV<158>, ZPZV<27>, ZPZV<2»; }; //
                    NOLINT
03364 template<> struct ConwayPolynomial<173, 9> { using ZPZ = aerobus::zpz<173>; using type =
                     POLYV<ZPZV<1>, ZPZV<0>, ZPZV<0>, ZPZV<0>, ZPZV<0>, ZPZV<0>, ZPZV<0>, ZPZV<0>, ZPZV<0>, ZPZV<56>, ZPZV<56>, ZPZV<171»;
 03365 template<> struct ConwayPolynomial<173, 10> { using ZPZ = aerobus::zpz<173>; using type
POLYV<ZPZV<1>, ZPZV<0>, ZPZV<0>, ZPZV<0>, ZPZV<1>, ZPZV<156>, ZPZV<164>, ZPZV<48>, ZPZV<106>, ZPZV<58>, ZPZV<28; }; // NOLINT

03366 template<> struct ConwayPolynomial<173, 11> { using ZPZ = aerobus::zpz<173>; using type = POLYV<ZPZV<1>, ZPZV<0>, ZPZV<1>; }; // NOLINT
 03367 template<> struct ConwayPolynomial<173, 12> { using ZPZ = aerobus::zpz<173>; using type =
                     POLYV<ZPZV<1>, ZPZV<0>, ZPZV<0>, ZPZV<0>, ZPZV<1>, ZPZV<29>, ZPZV<64>, ZPZV<46>, ZPZV<166>, ZPZV<10>,
                     03368 template<> struct ConwayPolynomial<173, 13> { using ZPZ = aerobus::zpz<173>; using type =
                     POLYV<ZPZV<1>, ZPZV<0>, ZPZV<0
                     ZPZV<0>, ZPZV<0>, ZPZV<6>, ZPZV<171»; };</pre>
                                                                                                                                                                  // NOLINT
03369 template<> struct ConwayPolynomial<173, 17> { using ZPZ = aerobus::zpz<173>; using type =
                     POLYV<ZPZV<1>, ZPZV<0>, ZPZV<0>, ZPZV<0>, ZPZV<0>, ZPZV<0>, ZPZV<0>, ZPZV<0>, ZPZV<0>, ZPZV<0>,
ZPZV<0>, ZPZV<0>, ZPZV<0>, ZPZV<0>, ZPZV<0>, ZPZV<0>, ZPZV<7>, ZPZV<7>, ZPZV<711»; }; // NOLINT
03370 template<> struct ConwayPolynomial<173, 19> { using ZPZ = aerobus::zpz<173>; using type
                     POLYV<ZPZV<1>, ZPZV<0>, ZPZV<0
                     ZPZV<0>, ZPZV<0>, ZPZV<0>, ZPZV<0>, ZPZV<0>, ZPZV<0>, ZPZV<0>, ZPZV<0>, ZPZV<0>, ZPZV<6>, ZPZV<6</pre>, ZPZV<171»; }; //</pre>
 03371 template<> struct ConwayPolynomial<179, 1> { using ZPZ = aerobus::zpz<179>; using type =
                    POLYV<ZPZV<1>, ZPZV<177»; }; // NOLINT
 03372 template<> struct ConwayPolynomial<179, 2> { using ZPZ = aerobus::zpz<179>; using type =
                    POLYV<ZPZV<1>, ZPZV<172>, ZPZV<2»; }; // NOLINT
 03373 template<> struct ConwayPolynomial<179, 3> { using ZPZ = aerobus::zpz<179>; using type =
                    POLYV<ZPZV<1>, ZPZV<0>, ZPZV<4>, ZPZV<177»; }; // NOLINT
 03374 template<> struct ConwayPolynomial<179, 4> { using ZPZ = aerobus::zpz<179>; using type =
POLYV<ZPZV<1>, ZPZV<0>, ZPZV<1>, ZPZV<10>, ZPZV<2»; }; // NOLINT
03375 template<> struct ConwayPolynomial<179, 5> { using ZPZ = aerobus::zpz<179>; using type = POLYV<ZPZV<1>, ZPZV<0>, ZPZV<0>, ZPZV<0>, ZPZV<2>, ZPZV<177»; }; // NOLINT
```

```
03376 template<> struct ConwayPolynomial<179, 6> { using ZPZ = aerobus::zpz<179>; using type =
POLYY<ZPZV<1>, ZPZV<0>, ZPZV<7>, ZPZV<91>, ZPZV<55>, ZPZV<109>, ZPZV<2»; }; // NOLINT 03377 template<> struct ConwayPolynomial<179, 7> { using ZPZ = aerobus::zpz<179>; using type
                      POLYV<ZPZV<1>, ZPZV<0>, ZPZV<0>, ZPZV<0>, ZPZV<0>, ZPZV<0>, ZPZV<6>, ZPZV<6>, ZPZV<177»; };
 03378 template<> struct ConwayPolynomial<179, 8> { using ZPZ = aerobus::zpz<179>; using type
                      POLYV<ZPZV<1>, ZPZV<0>, ZPZV<0>, ZPZV<0>, ZPZV<0>, ZPZV<163>, ZPZV<144>, ZPZV<73>, ZPZV<2»; }; //
 03379 template<> struct ConwayPolynomial<179, 9> { using ZPZ = aerobus::zpz<179>; using type
                      POLYV<ZPZV<1>, ZPZV<0>, ZPZV<0>, ZPZV<0>, ZPZV<0>, ZPZV<0>, ZPZV<0>, ZPZV<0>, ZPZV<40>, ZPZV<40>, ZPZV<46>, ZPZV<177»; };
                       // NOLTNT
03380 template<> struct ConwayPolynomial<179, 10> { using ZPZ = aerobus::zpz<179>; using type =
                      POLYV<ZPZV<1>, ZPZV<0>, ZPZV<0>, ZPZV<0>, ZPZV<0>, ZPZV<0>, ZPZV<15>, ZPZV<115>, ZPZV<71>, ZPZV<150>, ZPZV<49>, ZPZV<87>,
                       ZPZV<2»; }; // NOLINT</pre>
 03381 template<> struct ConwayPolynomial<179, 11> { using ZPZ = aerobus::zpz<179>; using type =
                      POLÝV<ZPZV<1>, ZPZV<0>, ZPZV<0
                       ZPZV<28>, ZPZV<177»; }; // NOLINT</pre>
03382 template<> struct ConwayPolynomial<179, 12> { using ZPZ = aerobus::zpz<179>; using type = POLYV<ZPZV<1>, ZPZV<0>, ZPZV<0>, ZPZV<0>, ZPZV<2>, ZPZV<103>, ZPZV<83>, ZPZV<43>, ZPZV<46>, ZPZV<88>, ZPZV<177>, ZPZV<1>, ZPZV<2»; }; // NOLINT
 03383 template<> struct ConwayPolynomial<179, 13> { using ZPZ = aerobus::zpz<179>; using type
                       POLYV<ZPZV<1>, ZPZV<0>, ZPZV<0
ZPZV<0>, ZPZV<18>, ZPZV<17>»; }; // NOLINT
03384 template<> struct ConwayPolynomial<179, 17> { using ZPZ = aerobus::zpz<179>; using type =
                      POLYV<ZPZV<1>, ZPZV<0>, ZPZV<0
03385 template<> struct ConwayPolynomial<179, 19> { using ZPZ = aerobus::zpz<179>; using type =
                       POLYV<ZPZV<1>, ZPZV<0>, ZPZV<0
                                                                                                                                                                                                                                                                                                                                             ZPZV<0>.
                      ZPZV<0>, ZPZV<0>, ZPZV<0>, ZPZV<0>, ZPZV<0>, ZPZV<0>, ZPZV<177»; }; //</pre>
                      NOLINT
03386 template<> struct ConwayPolynomial<181, 1> { using ZPZ = aerobus::zpz<181>; using type =
                      POLYV<ZPZV<1>, ZPZV<179»; }; // NOLINT
 03387 template<> struct ConwayPolynomial<181, 2> { using ZPZ = aerobus::zpz<181>; using type =
                      POLYV<ZPZV<1>, ZPZV<177>, ZPZV<2»; }; // NOLINT
 03388 template<> struct ConwayPolynomial<181, 3> { using ZPZ = aerobus::zpz<181>; using type =
                      POLYV<ZPZV<1>, ZPZV<0>, ZPZV<6>, ZPZV<179»; }; // NOLINT
03389 template<> struct ConwayPolynomial<181, 4> { using ZPZ = aerobus::zpz<181>; using type = POLYV<ZPZV<1>, ZPZV<0>, ZPZV<6>, ZPZV<105>, ZPZV<2»; }; // NOLINT
03390 template<> struct ConwayPolynomial<181, 5> { using ZPZ = aerobus::zpz<181>; using type =
                      POLYV<ZPZV<1>, ZPZV<0>, ZPZV<0>, ZPZV<2>, ZPZV<21>, ZPZV<179»; }; // NOLINT
 03391 template<> struct ConwayPolynomial<181, 6> { using ZPZ = aerobus::zpz<181>; using type =
                      POLYV<ZPZV<1>, ZPZV<0>, ZPZV<0>, ZPZV<177>, ZPZV<163>, ZPZV<169>, ZPZV<2»; }; // NOLINT
03392 template<> struct ConwayPolynomial<181, 7> { using ZPZ = aerobus::zpz<181>; using type =
                      POLYV<ZPZV<1>, ZPZV<0>, ZPZV<0>, ZPZV<0>, ZPZV<0>, ZPZV<0>, ZPZV<0>, ZPZV<4>, ZPZV<179»; };
03393 template<> struct ConwayPolynomial<181, 8> { using ZPZ = aerobus::zpz<181>; using type
                      POLYV<ZPZV<1>, ZPZV<0>, ZPZV<0>, ZPZV<0>, ZPZV<2>, ZPZV<108>, ZPZV<22>, ZPZV<149>, ZPZV<2»; }; //
                      NOLINT
03394 template<> struct ConwayPolynomial<181, 9> { using ZPZ = aerobus::zpz<181>; using type =
                      POLYV<ZPZV<1>, ZPZV<0>, ZPZV<0>, ZPZV<0>, ZPZV<0>, ZPZV<0>, ZPZV<0>, ZPZV<11>, ZPZV<107>, ZPZV<168>, ZPZV<179»;
                       }; // NOLINT
03395 template<> struct ConwayPolynomial<181, 10> { using ZPZ = aerobus::zpz<181>; using type = POLYV<ZPZV<1>, ZPZV<0>, ZPZV<0>, ZPZV<0>, ZPZV<5>, ZPZV<154>, ZPZV<104>, ZPZV<94>, ZPZV<57>, ZPZV<88>,
                       ZPZV<2»; }; // NOLINT</pre>
03396 template<> struct ConwayPolynomial<181, 11> { using ZPZ = aerobus::zpz<181>; using type
                      POLYV<ZPZV<1>, ZPZV<0>, ZPZV<0
 03397 template<> struct ConwayPolynomial<181, 12> { using ZPZ = aerobus::zpz<181>; using type
                       POLYV<ZPZV<1>, ZPZV<0>, ZPZV<0>, ZPZV<0>, ZPZV<0>, ZPZV<171>, ZPZV<141>, ZPZV<45>, ZPZV<122>,
ZPZV<175>, ZPZV<12>, ZPZV<10>, ZPZV<2»; }; // NOLINT
03398 template<> struct ConwayPolynomial<181, 13> { using ZPZ = aerobus::zpz<181>; using type =
                      POLYV<ZPZV<1>, ZPZV<0>, ZPZV<0
ZPZV<0>, ZPZV<0>, ZPZV<8>, ZPZV<179»; }; // NOLINT
03399 template<> struct ConwayPolynomial<181, 17> { using ZPZ = aerobus::zpz<181>; using type =
POLYV<ZPZV<1>, ZPZV<0>, ZPZV<0>,
ZPZV<0>, ZPZV<0>, ZPZV<0>, ZPZV<0>, ZPZV<0>, ZPZV<0>, ZPZV<0>, ZPZV<9>, ZPZV<179»; }; // NOLINT 03400 template<> struct ConwayPolynomial<181, 19> { using ZPZ = aerobus::zpz<181>; using type :
                      POLYV<ZPZV<1>, ZPZV<0>, ZPZV<36>, ZPZV<36>, ZPZV<36>, ZPZV<36>, ZPZV<37>, ZPZV<37, ZPZV<38, Z
                      NOLINT
03401 template<> struct ConwayPolynomial<191, 1> { using ZPZ = aerobus::zpz<191>; using type =
                      POLYV<ZPZV<1>, ZPZV<172»; }; // NOLINT
 03402 template<> struct ConwayPolynomial<191, 2> { using ZPZ = aerobus::zpz<191>; using type =
POLYV<ZPZV<1>, ZPZV<190>, ZPZV<19s; }; // NOLINT
03403 template<> struct ConwayPolynomial<191, 3> { using ZPZ = aerobus::zpz<191>; using type =
                      POLYV<ZPZV<1>, ZPZV<0>, ZPZV<4>, ZPZV<172»; }; // NOLINT
03404 template<> struct ConwayPolynomial<191, 4> { using ZPZ = aerobus::zpz<191>; using type =
                      POLYV<ZPZV<1>, ZPZV<0>, ZPZV<7>, ZPZV<100>, ZPZV<19»; }; // NOLINT
 03405 template<> struct ConwayPolynomial<191, 5> { using ZPZ = aerobus::zpz<191>; using type =
                      POLYV<ZPZV<1>, ZPZV<0>, ZPZV<0>, ZPZV<0>, ZPZV<1>, ZPZV<172»; }; // NOLINT
 03406 template<> struct ConwayPolynomial<191, 6> { using ZPZ = aerobus::zpz<191>; using type =
                      POLYV<ZPZV<1>, ZPZV<0>, ZPZV<1), ZPZV<110>, ZPZV<10>, ZPZV<10>, ZPZV<10>, ZPZV<19»; ); // NOLINT
03407 template<> struct ConwayPolynomial<191, 7> { using ZPZ = aerobus::zpz<191>; using type = POLYV<ZPZV<1>, ZPZV<0>, ZPZV<0>, ZPZV<0>, ZPZV<0>, ZPZV<0>, ZPZV<14>, ZPZV<172»; }; //
 03408 template<> struct ConwayPolynomial<191, 8> { using ZPZ = aerobus::zpz<191>; using type =
                      POLYV<ZPZV<1>, ZPZV<0>, ZPZV<0>, ZPZV<0>, ZPZV<0>, ZPZV<164>, ZPZV<139>, ZPZV<171>, ZPZV<19»; }; //
 03409 template<> struct ConwayPolynomial<191, 9> { using ZPZ = aerobus::zpz<191>; using type =
                      POLYY<ZPZV<1>, ZPZV<0>, ZPZV<0>, ZPZV<0>, ZPZV<0>, ZPZV<0>, ZPZV<0>, ZPZV<0>, ZPZV<6>, ZPZV<62>, ZPZV<124>, ZPZV<124>, ZPZV<172»;
```

```
}; // NOLINT
 03410 template<> struct ConwayPolynomial<191, 10> { using ZPZ = aerobus::zpz<191>; using type
                           POLYV<ZPZV<1>, ZPZV<0>, ZPZV<0>, ZPZV<0>, ZPZV<0>, ZPZV<113>, ZPZV<47>, ZPZV<173>, ZPZV<74>,
                           ZPZV<156>, ZPZV<19»; }; // NOLINT</pre>
03411 template<> struct ConwayPolynomial<191, 11> { using ZPZ = aerobus::zpz<191>; using type = POLYV<ZPZV<1>, ZPZV<0>, 
 03412 template<> struct ConwayPolynomial<191, 12> { using ZPZ = aerobus::zpz<191>; using type
                           POLYV<ZPZV<1>, ZPZV<0>, ZPZV<0>, ZPZV<0>, ZPZV<0>, ZPZV<7>, ZPZV<168>, ZPZV<25>, ZPZV<49>, ZPZV<90>,
                           \text{ZPZV}<7>, \text{ZPZV}<151>, \text{ZPZV}<19»; }; // NOLINT
 03413 template<> struct ConwayPolynomial<191, 13> { using ZPZ = aerobus::zpz<191>; using type =
                          POLYV<ZPZV<1>, ZPZV<0>, ZPZV<0
                           ZPZV<0>, ZPZV<0>, ZPZV<12>, ZPZV<172»; }; // NOLINT</pre>
 03414 template<> struct ConwayPolynomial<191, 17> { using ZPZ = aerobus::zpz<191>; using type =
                           POLYV<ZPZV<1>, ZPZV<0>, ZPZV<0
ZPZV<0>, ZPZV<0>, ZPZV<0>, ZPZV<0>, ZPZV<0>, ZPZV<0>, ZPZV<0>, ZPZV<0>, ZPZV<0>, ZPZV<1>, ZPZV<1>, ZPZV<1>, ZPZV<1>, ZPZV<1>, ZPZV<12, ZPZV<172»; }; // NOLINT

03415 template<> struct ConwayPolynomial<191, 19> { using ZPZ = aerobus::zpz<191>; using type =
POLYV<ZPZV<1>, ZPZV<0>, ZPZV<190>, ZPZV<2>, ZPZV<172»; }; //
 03416 template<> struct ConwayPolynomial<193, 1> { using ZPZ = aerobus::zpz<193>; using type =
                          POLYV<ZPZV<1>, ZPZV<188»; }; // NOLINT
03417 template<> struct ConwayPolynomial<193, 2> { using ZPZ = aerobus::zpz<193>; using type =
POLYV<ZPZV<1>, ZPZV<192>, ZPZV<5»; }; // NOLINT
03418 template<> struct ConwayPolynomial<193, 3> { using ZPZ = aerobus::zpz<193>; using type =
                          POLYV<ZPZV<1>, ZPZV<0>, ZPZV<1>, ZPZV<188»; }; // NOLINT
 03419 template<> struct ConwayPolynomial<193, 4> { using ZPZ = aerobus::zpz<193>; using type =
POLYV<ZPZV<1>, ZPZV<0>, ZPZV<6>, ZPZV<148>, ZPZV<5»; }; // NOLINT
03420 template<> struct ConwayPolynomial<193, 5> { using ZPZ = aerobus::zpz<193>; using type =
POLYV<ZPZV<1>, ZPZV<0>, ZPZV<0>, ZPZV<0>, ZPZV<7>, ZPZV<188»; }; // NOLINT

03421 template<> struct ConwayPolynomial<193, 6> { using ZPZ = aerobus::zpz<193>; using type =
                          POLYV<ZPZV<1>, ZPZV<0>, ZPZV<0>, ZPZV<149>, ZPZV<8>, ZPZV<172>, ZPZV<5»; }; // NOLINT
 03422 template<> struct ConwayPolynomial<193, 7> { using ZPZ = aerobus::zpz<193>; using type
                          POLYV<ZPZV<1>, ZPZV<0>, ZPZV<0>, ZPZV<0>, ZPZV<0>, ZPZV<0>, ZPZV<0>, ZPZV<0>, ZPZV<8>, ZPZV<188»; };
 03423 template<> struct ConwayPolynomial<193, 8> { using ZPZ = aerobus::zpz<193>; using type =
                           POLYV<ZPZV<1>, ZPZV<0>, ZPZV<0>, ZPZV<0>, ZPZV<10>, ZPZV<145>, ZPZV<34>, ZPZV<154>, ZPZV<15*; }; //
                           NOLINT
03424 template<> struct ConwayPolynomial<193, 9> { using ZPZ = aerobus::zpz<193>; using type =
                           POLYV<ZPZV<1>, ZPZV<0>, ZPZV<0>, ZPZV<0>, ZPZV<0>, ZPZV<0>, ZPZV<8>, ZPZV<168>, ZPZV<168>, ZPZV<27>, ZPZV<188»;
                            }; // NOLINT
03425 template<> struct ConwayPolynomial<193, 10> { using ZPZ = aerobus::zpz<193>; using type =
                          POLYV<ZPZV<1>, ZPZV<0>, ZPZV<0>, ZPZV<0>, ZPZV<20>, ZPZV<51>, ZPZV<77>, ZPZV<79, ZPZV<89>, ZPZV<5»; }; // NOLINT
03426 template<> struct ConwayPolynomial<193, 11> { using ZPZ = aerobus::zpz<193>; using type =
                          POLYV<ZPZV<1>, ZPZV<0>, ZPZV<1>, ZPZV<188»; }; // NOLINT
03427 template<> struct ConwayPolynomial<193, 12> { using ZPZ = aerobus::zpz<193>; using type =
                           POLYV<ZPZV<1>, ZPZV<0>, ZPZV<0>, ZPZV<0>, ZPZV<15>, ZPZV<52>, ZPZV<52>, ZPZV<135>, ZPZV<155>, ZPZV<135>, ZPZV<155>, ZPZV<
ZPZV<90>, ZPZV<46>, ZPZV<28>, ZPZV<59; }; // NOLINT
03428 template<> struct ConwayPolynomial<193, 13> { using ZPZ = aerobus::zpz<193>; using type =
                           POLYV<ZPZV<1>, ZPZV<0>, ZPZV<0
                            ZPZV<0>, ZPZV<0>, ZPZV<39>, ZPZV<188»; }; // NOLINT</pre>
 03429 template<> struct ConwayPolynomial<193, 17> { using ZPZ = aerobus::zpz<193>; using type =
                          POLYV<ZPZV<1>, ZPZV<0>, ZPZV<0
03430 template<> struct ConwayPolynomial<193, 19> { using ZPZ = aerobus::zpz<193>; using type = POLYV<ZPZV<1>, ZPZV<0>, ZPZV<0 , 
                            ZPZV<0>, ZPZV<0>
 03431 template<> struct ConwayPolynomial<197, 1> { using ZPZ = aerobus::zpz<197>; using type =
                          POLYV<ZPZV<1>, ZPZV<195»; }; // NOLINT
 03432 template<> struct ConwayPolynomial<197, 2> { using ZPZ = aerobus::zpz<197>; using type =
 POLYV<ZPZV<1>, ZPZV<192>, ZPZV<2»; }; // NOLINT
03433 template<> struct ConwayPolynomial<197, 3> { using ZPZ = aerobus::zpz<197>; using type =
                          POLYV<ZPZV<1>, ZPZV<0>, ZPZV<3>, ZPZV<195»; }; // NOLINT
 03434 template<> struct ConwayPolynomial<197, 4> { using ZPZ = aerobus::zpz<197>; using type =
POLYV<ZPZV<1>, ZPZV<0>, ZPZV<16>, ZPZV<124>, ZPZV<2»; }; // NOLINT

03435 template<> struct ConwayPolynomial<197, 5> { using ZPZ = aerobus::zpz<197>; using type = POLYV<ZPZV<1>, ZPZV<0>, ZPZV<0>, ZPZV<0>, ZPZV<4>, ZPZV<195»; }; // NOLINT
 03436 template<> struct ConwayPolynomial<197, 6> { using ZPZ = aerobus::zpz<197>; using type
                           POLYV<ZPZV<1>, ZPZV<0>, ZPZV<1>, ZPZV<124>, ZPZV<79>, ZPZV<173>, ZPZV<2»; }; // NOLINT
 03437 template<> struct ConwayPolynomial<197, 7> { using ZPZ = aerobus::zpz<197>; using type =
                          POLYV<ZPZV<1>, ZPZV<0>, ZPZV<0>, ZPZV<0>, ZPZV<0>, ZPZV<0>, ZPZV<0>, ZPZV<6>, ZPZV<195»; };
 03438 template<> struct ConwayPolynomial<197, 8> { using ZPZ = aerobus::zpz<197>; using type =
                           POLYV<ZPZV<1>, ZPZV<0>, ZPZV<0>, ZPZV<0>, ZPZV<0>, ZPZV<176>, ZPZV<96>, ZPZV<29>, ZPZV<29; }; //
 03439 template<> struct ConwayPolynomial<197, 9> { using ZPZ = aerobus::zpz<197>; using type
                           POLÝV<ZPZV<1>, ZPZV<0>, ZPZV<0>, ZPZV<0>, ZPZV<0>, ZPZV<0>, ZPZV<13>, ZPZV<127>, ZPZV<8>, ZPZV<195»;
                            }; // NOLTNT
03440 template<> struct ConwayPolynomial<197, 10> { using ZPZ = aerobus::zpz<197>; using type =
                          POLYV<ZPZV<1>, ZPZV<0>, ZPZV<0>, ZPZV<0>, ZPZV<0>, ZPZV<121>, ZPZV<137>, ZPZV<8>, ZPZV<73>, ZPZV<42>,
                           ZPZV<2»; }; // NOLINT</pre>
 03441 template<> struct ConwayPolynomial<197, 11> { using ZPZ = aerobus::zpz<197>; using type
                           POLYV<ZPZV<1>, ZPZV<0>, ZPZV<0
                           ZPZV<14>, ZPZV<195»; }; // NOLINT</pre>
03442 template<> struct ConwayPolynomial<197, 12> { using ZPZ = aerobus::zpz<197>; using type = POLYV<ZPZV<1>, ZPZV<0>, ZPZV<0>, ZPZV<0>, ZPZV<0>, ZPZV<168>, ZPZV<15>, ZPZV<130>, ZPZV<141>, ZPZV<9>,
```

```
ZPZV<90>, ZPZV<163>, ZPZV<2»; };
03443 template<> struct ConwayPolynomial<197, 13> { using ZPZ = aerobus::zpz<197>; using type =
                                POLYV<ZPZV<1>, ZPZV<0>, ZPZV<0
03444 template<> struct ConwayPolynomial<197, 17> { using ZPZ = aerobus::zpz<197>; using type = POLYV<ZPZV<1>, ZPZV<0>, 
                                 ZPZV<0>, ZPZV<0>, ZPZV<0>, ZPZV<0>, ZPZV<0>, ZPZV<0>, ZPZV<35>, ZPZV<195»; }; // NOLINT
03445 template<> struct ConwayPolynomial<197, 19> { using ZPZ = aerobus::zpz<197>; using type =
                                  \texttt{POLYV} < \texttt{ZPZV} < 1>, \quad \texttt{ZPZV} < 0>, \quad 
                                 ZPZV<0>, ZPZV<0>, ZPZV<0>, ZPZV<0>, ZPZV<0>, ZPZV<0>, ZPZV<0>, ZPZV<0>, ZPZV<6>, ZPZV<6>, ZPZV<6</pre>
                                 NOLINT
03446 template<> struct ConwayPolynomial<199, 1> { using ZPZ = aerobus::zpz<199>; using type =
                                POLYV<ZPZV<1>, ZPZV<196»; }; // NOLINT
 03447 template<> struct ConwayPolynomial<199, 2> { using ZPZ = aerobus::zpz<199>; using type =
                                POLYV<ZPZV<1>, ZPZV<193>, ZPZV<3»; };
                                                                                                                                                                                                                                           // NOLINT
03448 template<> struct ConwayPolynomial<199, 3> { using ZPZ = aerobus::zpz<199>; using type =
POLYV<ZPZV<1>, ZPZV<0>, ZPZV<1>, ZPZV<196»; }; // NOLINT
03449 template<> struct ConwayPolynomial<199, 4> { using ZPZ = aerobus::zpz<199>; using type =
POLYY<ZPZV<1>, ZPZV<0>, ZPZV<7>, ZPZV<162>, ZPZV<3; ; // NOLINT

03450 template<> struct ConwayPolynomial<199, 5> { using ZPZ = aerobus::zpz<199>; using type =
                                POLYV<ZPZV<1>, ZPZV<0>, ZPZV<0>, ZPZV<0>, ZPZV<3>, ZPZV<196»; }; // NOLINT
 03451 template<> struct ConwayPolynomial<199, 6> { using ZPZ = aerobus::zpz<199>; using type =
POLYV<ZPZV<1>, ZPZV<0>, ZPZV<0>, ZPZV<90>, ZPZV<58>, ZPZV<79>, ZPZV<3»; }; // NOLINT 03452 template<> struct ConwayPolynomial<199, 7> { using ZPZ = aerobus::zpz<199>; using type
                                POLYV<ZPZV<1>, ZPZV<0>, ZPZV<0>, ZPZV<0>, ZPZV<0>, ZPZV<0>, ZPZV<3>, ZPZV<196»; };
                                                                                                                                                                                                                                                                                                                                                                                                                                                                                            // NOLINT
 03453 template<> struct ConwayPolynomial<199, 8> { using ZPZ = aerobus::zpz<199>; using type
                                 POLYV<ZPZV<1>, ZPZV<0>, ZPZV<0>, ZPZV<0>, ZPZV<1>, ZPZV<160>, ZPZV<23>, ZPZV<159>, ZPZV<3»; }; //
                                NOT.TNT
03454 template<> struct ConwayPolynomial<199, 9> { using ZPZ = aerobus::zpz<199>; using type :
                                 POLYV<ZPZV<1>, ZPZV<0>, ZPZV<0>, ZPZV<0>, ZPZV<0>, ZPZV<0>, ZPZV<8>, ZPZV<177>, ZPZV<141>, ZPZV<196»;
                                  }; // NOLINT
03455 template<> struct ConwayPolynomial<199, 10> { using ZPZ = aerobus::zpz<199>; using type
                                  POLÝV<ZPZV<1>, ZPZV<0>, ZPZV<0>, ZPZV<0>, ZPZV<0>, ZPZV<0>, ZPZV<171>, ZPZV<158>, ZPZV<31>, ZPZV<54>, ZPZV<9>,
                                 ZPZV<3»; }; // NOLINT</pre>
03456 template<> struct ConwayPolynomial<199, 11> { using ZPZ = aerobus::zpz<199>; using type = POLYV<ZPZV<1>, ZPZV<0>, 
                                 ZPZV<1>, ZPZV<196»; }; // NOLINT</pre>
 03457 template<> struct ConwayPolynomial<199, 12> { using ZPZ = aerobus::zpz<199>; using type
                                 POLYV<ZPZV<1>, ZPZV<0>, ZPZV<0>, ZPZV<0>, ZPZV<0>, ZPZV<3>, ZPZV<192>, ZPZV<197>, ZPZV<138>,
ZPZV<69>, ZPZV<57>, ZPZV<151>, ZPZV<3»; }; // NOLINT
03458 template<> struct ConwayPolynomial<199, 13> { using ZPZ = aerobus::zpz<199>; using type =
POLYY<ZPZY<1>, ZPZY<0>, ZPZV<0>, ZPZV<0
                                 POLYV<ZPZV<1>, ZPZV<0>, ZPZV<0
                                  ZPZV<0>, ZPZV<0>, ZPZV<0>, ZPZV<0>, ZPZV<0>, ZPZV<0>, ZPZV<13>, ZPZV<196»; };</pre>
                                                                                                                                                                                                                                                                                                                                                                                                                                                                // NOLINT
03460 template<> struct ConwayPolynomial<199, 19> { using ZPZ = aerobus::zpz<199>; using type =
                                POLYV<ZPZV<1>, ZPZV<0>, ZPZV<0
03461 template<> struct ConwayPolynomial<211, 1> { using ZPZ = aerobus::zpz<211>; using type =
                                 POLYV<ZPZV<1>, ZPZV<209»; }; // NOLINT
 03462 template<> struct ConwayPolynomial<211, 2> { using ZPZ = aerobus::zpz<211>; using type =
POLYV<ZPZV<1>, ZPZV<207, ZPZV<2»; }; // NOLINT
03463 template<> struct ConwayPolynomial<211, 3> { using ZPZ = aerobus::zpz<211>; using type =
                                POLYV<ZPZV<1>, ZPZV<0>, ZPZV<2>, ZPZV<209»; }; // NOLINT
 03464 template<> struct ConwayPolynomial<211, 4> { using ZPZ = aerobus::zpz<211>; using type =
POLYV<ZPZV<1>, ZPZV<6>, ZPZV<6>, ZPZV<65, ZPZV<2*; }; // NOLINT
03465 template<> struct ConwayPolynomial<211, 5> { using ZPZ = aerobus::zpz<211>; using type =
POLYV<ZPZV<1>, ZPZV<0>, ZPZV<0>, ZPZV<0>, ZPZV<13>, ZPZV<209»; }; // NOLINT
03466 template<> struct ConwayPolynomial<211, 6> { using ZPZ = aerobus::zpz<211>; using type =
                                POLYV<ZPZV<1>, ZPZV<0>, ZPZV<0>, ZPZV<81>, ZPZV<194>, ZPZV<133>, ZPZV<2»; }; // NOLINT
 03467 template<> struct ConwayPolynomial<211, 7> { using ZPZ = aerobus::zpz<211>; using type =
                                 POLYV<ZPZV<1>, ZPZV<0>, ZPZV<0
03468 template<> struct ConwayPolynomial<211, 8> { using ZPZ = aerobus::zpz<211>; using type =
                                 POLYV<ZPZV<1>, ZPZV<0>, ZPZV<0>, ZPZV<0>, ZPZV<2>, ZPZV<200>, ZPZV<87>, ZPZV<29>, ZPZV<29; }; //
                                 NOLINT
03469 template<> struct ConwayPolynomial<211, 9> { using ZPZ = aerobus::zpz<211>; using type =
                                 POLYV<ZPZV<1>, ZPZV<0>, ZPZV<0>, ZPZV<0>, ZPZV<0>, ZPZV<0>, ZPZV<19>, ZPZV<139>, ZPZV<26>, ZPZV<209»;
                                  }; // NOLINT
03470 template<> struct ConwayPolynomial<211, 10> { using ZPZ = aerobus::zpz<211>; using type =
                                 POLYV<ZPZV<1>, ZPZV<0>, ZPZV<0>, ZPZV<0>, ZPZV<1>, ZPZV<30>, ZPZV<61>, ZPZV<148>, ZPZV<87>, ZPZV<125>,
                                 ZPZV<2»; }; // NOLINT</pre>
03471 template<> struct ConwayPolynomial<211, 11> { using ZPZ = aerobus::zpz<211>; using type =
                                POLYV<ZPZV<1>, ZPZV<0>, ZPZV<0
 03472 template<> struct ConwayPolynomial<211, 12> { using ZPZ = aerobus::zpz<211>; using type
                                POLYV<ZPZV<1>, ZPZV<0>, ZPZV<0>, ZPZV<0>, ZPZV<0>, ZPZV<6>, ZPZV<8>, ZPZV<50>, ZPZV<145>, ZPZV<126>, ZPZV<184>, ZPZV<84>, ZPZV<27>, ZPZV<284>, ZPZV<284
03473 template<> struct ConwayPolynomial<211, 13> { using ZPZ = aerobus::zpz<211>; using type =
                                 POLYV<ZPZV<1>, ZPZV<0>, ZPZV<0
 ZPZV<0>, ZPZV<1>, ZPZV<12>, ZPZV<209»; }; // NOLINT
03474 template<> struct ConwayPolynomial<211, 17> { using ZPZ = aerobus::zpz<211>; using type
                                 POLYV<ZPZV<1>, ZPZV<0>, ZPZV<0>, ZPZV<0>, ZPZV<0>, ZPZV<0>, ZPZV<0>, ZPZV<0>, ZPZV<0>, ZPZV<0>, ZPZV<0>,
ZPZV<0>, ZPZV<0>, ZPZV<0>, ZPZV<0>, ZPZV<0>, ZPZV<0>, ZPZV<0>, ZPZV<1>, ZPZV<12>, ZPZV<209»; }; // NOLINT
03475 template<>> struct ConwayPolynomial<211, 19> { using ZPZ = aerobus::zpz<211>; using type =
POLYV<ZPZV<1>, ZPZV<0>, ZPZV<
```

```
ZPZV<0>, ZPZV<0>, ZPZV<0>, ZPZV<0>, ZPZV<0>, ZPZV<0>, ZPZV<0>, ZPZV<0>, ZPZV<0>, ZPZV<17>, ZPZV<209»; }; //</pre>
03476 template<> struct ConwayPolynomial<223, 1> { using ZPZ = aerobus::zpz<223>; using type =
                       POLYV<ZPZV<1>, ZPZV<220»; }; // NOLINT
 03477 template<> struct ConwayPolynomial<223, 2> { using ZPZ = aerobus::zpz<223>; using type =
                       POLYV<ZPZV<1>, ZPZV<221>, ZPZV<3»; }; // NOLINT
 03478 template<> struct ConwayPolynomial<223, 3> { using ZPZ = aerobus::zpz<223>; using type =
                       POLYV<ZPZV<1>, ZPZV<0>, ZPZV<6>, ZPZV<220»; }; // NOLINT
 03479 template<> struct ConwayPolynomial<223, 4> { using ZPZ = aerobus::zpz<223>; using type =
POLYV<ZPZV<1>, ZPZV<0>, ZPZV<6>, ZPZV<163>, ZPZV<3»; }; // NOLINT
03480 template<> struct ConwayPolynomial<223, 5> { using ZPZ = aerobus::zpz<223>; using type =
                       POLYV<ZPZV<1>, ZPZV<0>, ZPZV<0>, ZPZV<0>, ZPZV<1>, ZPZV<220»; }; // NOLINT
03481 template<> struct ConwayPolynomial<223, 6> { using ZPZ = aerobus::zpz<223>; using type =
                       POLYV<ZPZV<1>, ZPZV<0>, ZPZV<2>, ZPZV<68>, ZPZV<24>, ZPZV<196>, ZPZV<3»; }; // NOLINT
 03482 template<> struct ConwayPolynomial<223, 7> { using ZPZ = aerobus::zpz<223>; using type
                       POLYV<ZPZV<1>, ZPZV<0>, ZPZV<0>, ZPZV<0>, ZPZV<0>, ZPZV<0>, ZPZV<6>, ZPZV<220»; };
 03483 template<> struct ConwayPolynomial<223, 8> { using ZPZ = aerobus::zpz<223>; using type =
                       POLYV<ZPZV<1>, ZPZV<0>, ZPZV<0>, ZPZV<0>, ZPZV<2>, ZPZV<139>, ZPZV<98>, ZPZV<138>, ZPZV<3»; }; //
03484 template<> struct ConwayPolynomial<223, 9> { using ZPZ = aerobus::zpz<223>; using type =
                       POLYV<ZPZV<1>, ZPZV<0>, ZPZV<0>, ZPZV<0>, ZPZV<0>, ZPZV<0>, ZPZV<0>, ZPZV<1>, ZPZV<164>, ZPZV<64>, ZPZV<220»;
                         }; // NOLINT
03485 template<> struct ConwayPolynomial<223, 10> { using ZPZ = aerobus::zpz<223>; using type =
                       POLYV<ZPZV<1>, ZPZV<0>, ZPZV<0>, ZPZV<0>, ZPZV<1>, ZPZV<118>, ZPZV<177>, ZPZV<87>, ZPZV<99>, ZPZV<62>,
                        ZPZV<3»; }; // NOLINT</pre>
03486 template<> struct ConwayPolynomial<223, 11> { using ZPZ = aerobus::zpz<223>; using type = POLYV<ZPZV<1>, ZPZV<0>, ZPZV<0 , 
                       ZPZV<8>, ZPZV<220»; }; // NOLINT</pre>
03487 template<> struct ConwayPolynomial<223, 12> { using ZPZ = aerobus::zpz<223>; using type
                       POLYV<ZPZV<1>, ZPZV<0>, ZPZV<0>, ZPZV<0>, ZPZV<1>, ZPZV<64>, ZPZV<94>, ZPZV<11>, ZPZV<105>, ZPZV<64>, ZPZV<151>, ZPZV<213>, ZPZV<3»; }; // NOLINT
03488 template<> struct ConwayPolynomial<223, 13> { using ZPZ = aerobus::zpz<223>; using type
                        POLYV<ZPZV<1>, ZPZV<0>, ZPZV<0
ZPZV<0>, ZPZV<0>, ZPZV<23>, ZPZV<220»; }; // NOLINT
03489 template<> struct ConwayPolynomial<223, 17> { using ZPZ = aerobus::zpz<223>; using type =
                       POLYV<ZPZV<1>, ZPZV<0>, ZPZV<0
ZPZV<0>, ZPZV<0>, ZPZV<0>, ZPZV<0>, ZPZV<0>, ZPZV<0>, ZPZV<0>, ZPZV<0>, ZPZV<2>, ZPZV<2>, ZPZV<2>, ZPZV<2>, ZPZV<220»; }; // NOLINT
03490 template<> struct ConwayPolynomial<223, 19> { using ZPZ = aerobus::zpz<223>; using type =
POLYV<ZPZV<1>, ZPZV<0>, ZPZ
                        ZPZV<0>, ZPZV<0>, ZPZV<0>, ZPZV<0>, ZPZV<0>, ZPZV<0>, ZPZV<0>, ZPZV<0>, ZPZV<7>, ZPZV<7>, ZPZV<20»; }; //</pre>
                       NOLINT
03491 template<> struct ConwayPolynomial<227, 1> { using ZPZ = aerobus::zpz<227>; using type =
                       POLYV<ZPZV<1>, ZPZV<225»; }; // NOLINT
03492 template<> struct ConwayPolynomial<227, 2> { using ZPZ = aerobus::zpz<227>; using type =
POLYV<ZPZV<1>, ZPZV<220, ZPZV<2»; }; // NOLINT
03493 template<> struct ConwayPolynomial<227, 3> { using ZPZ = aerobus::zpz<227>; using type =
                       POLYV<ZPZV<1>, ZPZV<0>, ZPZV<2>, ZPZV<225»; }; // NOLINT
03494 template<> struct ConwayPolynomial<227, 4> { using ZPZ = aerobus::zpz<227>; using type =
POLYV<ZPZV<1>, ZPZV<0>, ZPZV<1>, ZPZV<143>, ZPZV<2»; }; // NOLINT

03495 template<> struct ConwayPolynomial<227, 5> { using ZPZ = aerobus::zpz<227>; using type = POLYV<ZPZV<1>, ZPZV<0>, ZPZV<0>, ZPZV<5>, ZPZV<25>; }; // NOLINT
 03496 template<> struct ConwayPolynomial<227, 6> { using ZPZ = aerobus::zpz<227>; using type =
                       POLYV<ZPZV<1>, ZPZV<0>, ZPZV<1>, ZPZV<174>, ZPZV<24>, ZPZV<135>, ZPZV<2»; }; // NOLINT
 03497 template<> struct ConwayPolynomial<227, 7> { using ZPZ = aerobus::zpz<227>; using type =
                       POLYV<ZPZV<1>, ZPZV<0>, ZPZV<0>, ZPZV<0>, ZPZV<0>, ZPZV<0>, ZPZV<2>, ZPZV<25»; }; // NOLINT
 03498 template<> struct ConwayPolynomial<227, 8> { using ZPZ = aerobus::zpz<227>; using type
                       POLYV<ZPZV<1>, ZPZV<0>, ZPZV<0>, ZPZV<0>, ZPZV<2>, ZPZV<151>, ZPZV<176>, ZPZV<106>, ZPZV<2»; }; //
 03499 template<> struct ConwayPolynomial<227, 9> { using ZPZ = aerobus::zpz<227>; using type =
                       POLYV<ZPZV<1>, ZPZV<0>, ZPZV<0>, ZPZV<0>, ZPZV<0>, ZPZV<0>, ZPZV<3>, ZPZV<24>, ZPZV<183>, ZPZV<225»;
                        }; // NOLINT
03500 template<> struct ConwayPolynomial<227, 10> { using ZPZ = aerobus::zpz<227>; using type = POLYV<ZPZV<1>, ZPZV<0>, ZPZV<0>, ZPZV<0>, ZPZV<0>, ZPZV<2>, ZPZV<199>, ZPZV<12>, ZPZV<193>, ZPZV<7>,
                       ZPZV<2»; }; // NOLINT</pre>
 03501 template<> struct ConwayPolynomial<227, 11> { using ZPZ = aerobus::zpz<227>; using type =
                       POLYV<ZPZV<1>, ZPZV<0>, ZPZV<0
                       ZPZV<2>, ZPZV<225»; }; // NOLINT</pre>
03502 template<> struct ConwayPolynomial<227, 12> { using ZPZ = aerobus::zpz<227>; using type =
                       POLYV<ZPZV<1>, ZPZV<0>, ZPZV<0>, ZPZV<0>, ZPZV<0>, ZPZV<123>, ZPZV<99>, ZPZV<160>, ZPZV<96>,
ZPZV<127>, ZPZV<142>, ZPZV<94>, ZPZV<2x; }; // NOLINT
03503 template<> struct ConwayPolynomial<227, 13> { using ZPZ = aerobus::zpz<227>; using type =
                       POLYV<ZPZV<1>, ZPZV<0>, ZPZV<0
ZPZV<0>, ZPZV<0>, ZPZV<2>, ZPZV<225»; }; // NOLINT 03504 template<> struct ConwayPolynomial<227, 17> { using ZPZ = aerobus::zpz<227>; using type =
                       POLYV<ZPZV<1>, ZPZV<0>, ZPZV<0
                        ZPZV<0>, ZPZV<0>, ZPZV<0>, ZPZV<0>, ZPZV<0>, ZPZV<0>, ZPZV<8>, ZPZV<225»; }; // NOLINT</pre>
 03505 template<> struct ConwayPolynomial<227, 19> { using ZPZ = aerobus::zpz<227>; using type
                       POLÝV<ZPZV<1>, ZPZV<0>, ZPZV<0
                       ZPZV<0>, ZPZV<0>, ZPZV<0>, ZPZV<0>, ZPZV<0>, ZPZV<0>, ZPZV<0>, ZPZV<0>, ZPZV<0>, ZPZV<34>, ZPZV<225»; }; //</pre>
                       NOLINT
 03506 template<> struct ConwayPolynomial<229, 1> { using ZPZ = aerobus::zpz<229>; using type =
                       POLYV<ZPZV<1>, ZPZV<223»; }; // NOLINT
 03507 template<> struct ConwayPolynomial<229, 2> { using ZPZ = aerobus::zpz<229>; using type =
                       POLYV<ZPZV<1>, ZPZV<228>, ZPZV<6»; };
                                                                                                                                                                         // NOLINT
 03508 template<> struct ConwayPolynomial<229, 3> { using ZPZ = aerobus::zpz<229>; using type =
POLYV<ZPZV<1>, ZPZV<0>, ZPZV<1>, ZPZV<223»; }; // NOLINT
03509 template<> struct ConwayPolynomial<229, 4> { using ZPZ = aerobus::zpz<229>; using type =
```

```
POLYV<ZPZV<1>, ZPZV<0>, ZPZV<7>, ZPZV<162>, ZPZV<6»; }; // NOLINT 03510 template<> struct ConwayPolynomial<229, 5> { using ZPZ = aerobus::zpz<229>; using type =
                      POLYV<ZPZV<1>, ZPZV<0>, ZPZV<0>, ZPZV<0>, ZPZV<8>, ZPZV<223»; }; // NOLINT
 03511 template<> struct ConwayPolynomial<229, 6> { using ZPZ = aerobus::zpz<229>; using type =
POLYY<ZPZV<1>, ZPZV<0>, ZPZV<0>, ZPZV<24>, ZPZV<160>, ZPZV<186>, ZPZV<6»; }; // NOLINT 03512 template<> struct ConwayPolynomial<229, 7> { using ZPZ = aerobus::zpz<229>; using type =
                      POLYV<ZPZV<1>, ZPZV<0>, ZPZV<0>, ZPZV<0>, ZPZV<0>, ZPZV<0>, ZPZV<7>, ZPZV<23»; }; // NOLINT
 03513 template<> struct ConwayPolynomial<229, 8> { using ZPZ = aerobus::zpz<229>; using type
                       POLYV<ZPZV<1>, ZPZV<0>, ZPZV<0>, ZPZV<0>, ZPZV<8>, ZPZV<193>, ZPZV<62>, ZPZV<205>, ZPZV<6»; }; //
                       NOLINT
03514 template<> struct ConwayPolynomial<229, 9> { using ZPZ = aerobus::zpz<229>; using type =
                      POLYV<ZPZV<1>, ZPZV<0>, ZPZV<0>, ZPZV<0>, ZPZV<0>, ZPZV<0>, ZPZV<0>, ZPZV<15>, ZPZV<117>, ZPZV<50>, ZPZV<223»;
                        }: // NOLINT
 03515 template<> struct ConwayPolynomial<229, 10> { using ZPZ = aerobus::zpz<229>; using type =
                       POLYV<ZPZV<1>, ZPZV<0>, ZPZV<0>, ZPZV<0>, ZPZV<1>, ZPZV<185>, ZPZV<135>, ZPZV<158>, ZPZV<167>,
                        ZPZV<98>, ZPZV<6»; }; // NOLINT</pre>
03516 template<> struct ConwayPolynomial<229, 11> { using ZPZ = aerobus::zpz<229>; using type =
                      POLYV<ZPZV<1>, ZPZV<0>, ZPZV<0
 03517 template<> struct ConwayPolynomial<229, 12> { using ZPZ = aerobus::zpz<229>; using type
                      POLYV<ZPZV<1>, ZPZV<0>, ZPZV<0>, ZPZV<0>, ZPZV<131>, ZPZV<140>, ZPZV<25>, ZPZV<6>, ZPZV<172>, ZPZV<9>, ZPZV<145>, ZPZV<6»; }; // NOLINT
03518 template<> struct ConwayPolynomial<229, 13> { using ZPZ = aerobus::zpz<229>; using type =
                       POLYV<ZPZV<1>, ZPZV<0>, ZPZV<0
                        ZPZV<0>, ZPZV<0>, ZPZV<47>, ZPZV<223»; };</pre>
                                                                                                                                                                                      // NOLINT
03519 template<> struct ConwayPolynomial<229, 17> { using ZPZ = aerobus::zpz<229>; using type
                       POLYV<ZPZV<1>, ZPZV<0>, ZPZV<0
ZPZV<0>, ZPZV<0>, ZPZV<0>, ZPZV<0>, ZPZV<0>, ZPZV<0>, ZPZV<2>, ZPZV<2>, ZPZV<2>, ZPZV<2>, ZPZV<2>, ZPZV<2>, ZPZV<2
                       POLYV<ZPZV<1>, ZPZV<0>, ZPZV<0
                       ZPZV<0>, ZPZV<0>, ZPZV<0>, ZPZV<0>, ZPZV<0>, ZPZV<0>, ZPZV<0>, ZPZV<228>, ZPZV<15>, ZPZV<223»; }; //</pre>
                       NOLINT
 03521 template<> struct ConwayPolynomial<233, 1> { using ZPZ = aerobus::zpz<233>; using type =
                       POLYV<ZPZV<1>, ZPZV<230»; }; // NOLINT
 03522 template<> struct ConwayPolynomial<233, 2> { using ZPZ = aerobus::zpz<233>; using type =
                      POLYV<ZPZV<1>, ZPZV<232>, ZPZV<3»; }; // NOLINT
03523 template<> struct ConwayPolynomial<233, 3> { using ZPZ = aerobus::zpz<233>; using type =
                      POLYV<ZPZV<1>, ZPZV<0>, ZPZV<1>, ZPZV<230»; }; // NOLINT
 03524 template<> struct ConwayPolynomial<233, 4> { using ZPZ = aerobus::zpz<233>; using type =
POLYV<ZPZV<1>, ZPZV<0>, ZPZV<4>, ZPZV<158>, ZPZV<3»; }; // NOLINT
03525 template<> struct ConwayPolynomial<233, 5> { using ZPZ = aerobus::zpz<233>; using type =
POLYV<ZPZV<1>, ZPZV<0>, ZPZV<0>, ZPZV<0>, ZPZV<23, ZPZV<230»; }; // NOLINT

03526 template<> struct ConwayPolynomial<233, 6> { using ZPZ = aerobus::zpz<233>; using type =
                       POLYV<ZPZV<1>, ZPZV<0>, ZPZV<3>, ZPZV<122>, ZPZV<215>, ZPZV<32>, ZPZV<3»; }; // NOLINT
 03527 template<> struct ConwayPolynomial<233, 7> { using ZPZ = aerobus::zpz<233>; using type
                      POLYV<ZPZV<1>, ZPZV<0>, ZPZV<0>, ZPZV<0>, ZPZV<0>, ZPZV<4>, ZPZV<4>, ZPZV<230»; };
03528 template<> struct ConwayPolynomial<233, 8> { using ZPZ = aerobus::zpz<233>; using type =
                      POLYV<ZPZV<1>, ZPZV<0>, ZPZV<0>, ZPZV<0>, ZPZV<12>, ZPZV<202>, ZPZV<135>, ZPZV<181>, ZPZV<3»; }; //
                      NOLINT
03529 template<> struct ConwayPolynomial<233, 9> { using ZPZ = aerobus::zpz<233>; using type =
                       POLYV<ZPZV<1>, ZPZV<0>, ZPZV<0>, ZPZV<0>, ZPZV<0>, ZPZV<0>, ZPZV<0>, ZPZV<5>, ZPZV<56>, ZPZV<146>, ZPZV<230»;
                        }; // NOLINT
 03530 template<> struct ConwayPolynomial<233, 10> { using ZPZ = aerobus::zpz<233>; using type =
                      POLYV<ZPZV<1>, ZPZV<0>, ZPZV<0>, ZPZV<0>, ZPZV<3>, ZPZV<28>, ZPZV<71>, ZPZV<102>, ZPZV<3>, ZPZV<48>, ZPZV<3»; }; // NOLINT
 03531 template<> struct ConwayPolynomial<233, 11> { using ZPZ = aerobus::zpz<233>; using type
                       POLYV<ZPZV<1>, ZPZV<0>, ZPZV<0
 03532 template<> struct ConwayPolynomial<233, 12> { using ZPZ = aerobus::zpz<233>; using type =
                       POLYV<ZPZV<1>, ZPZV<0>, ZPZV<0>, ZPZV<0>, ZPZV<0>, ZPZV<0>, ZPZV<21>, ZPZV<114>, ZPZV<114>, ZPZV<31>, ZPZV<19>,
ZPZV<216, ZPZV<20>, ZPZV<3»; }; // NOLINT

03533 template<> struct ConwayPolynomial<233, 13> { using ZPZ = aerobus::zpz<233>; using type = POLYV<ZPZV<1>, ZPZV<0>, ZP
                        ZPZV<0>, ZPZV<0>, ZPZV<14>, ZPZV<230»; }; // NOLINT</pre>
 03534 template<> struct ConwayPolynomial<233, 17> { using ZPZ = aerobus::zpz<233>; using type =
                       POLYV<ZPZV<1>, ZPZV<0>, ZPZV<0
ZPZV<0>, ZPZV<0>, ZPZV<0>, ZPZV<0>, ZPZV<0>, ZPZV<0>, ZPZV<4>, ZPZV<4>, ZPZV<230_{\odot}; }; // NOLINT 03535 template<> struct ConwayPolynomial<233, 19> { using ZPZ = aerobus::zpz<233>; using type =
                       POLYV<ZPZV<1>, ZPZV<0>, ZPZV<0
                        ZPZV<0>, ZPZV<0>, ZPZV<0>, ZPZV<0>, ZPZV<0>, ZPZV<0>, ZPZV<0>, ZPZV<0>, ZPZV<0>, ZPZV<25>, ZPZV<23o»; }; //</pre>
03536 template<> struct ConwayPolynomial<239, 1> { using ZPZ = aerobus::zpz<239>; using type =
                       POLYV<ZPZV<1>, ZPZV<232»; }; // NOLINT
 03537 template<> struct ConwayPolynomial<239, 2> { using ZPZ = aerobus::zpz<239>; using type =
                      POLYV<ZPZV<1>, ZPZV<237>, ZPZV<7»; };
                                                                                                                                                                      // NOLINT
 03538 template<> struct ConwayPolynomial<239, 3> { using ZPZ = aerobus::zpz<239>; using type =
                       POLYV<ZPZV<1>, ZPZV<0>, ZPZV<11>, ZPZV<232»; }; // NOLINT
 03539 template<> struct ConwayPolynomial<239, 4> { using ZPZ = aerobus::zpz<239>; using type =
POLYV<ZPZV<1>, ZPZV<0>, ZPZV<11>, ZPZV<132>, ZPZV<7»; }; // NOLINT
03540 template<> struct ConwayPolynomial<239, 5> { using ZPZ = aerobus::zpz<239>; using type =
                      POLYV<ZPZV<1>, ZPZV<0>, ZPZV<0>, ZPZV<0>, ZPZV<2>, ZPZV<232»; }; // NOLINT
 03541 template<> struct ConwayPolynomial<239, 6> { using ZPZ = aerobus::zpz<239>; using type =
                       POLYV<ZPZV<1>, ZPZV<0>, ZPZV<1>, ZPZV<237>, ZPZV<60>, ZPZV<200>, ZPZV<7»; }; // NOLINT
 03542 template<> struct ConwayPolynomial<239, 7> { using ZPZ = aerobus::zpz<239>; using type =
                      POLYV<ZPZV<1>, ZPZV<0>, ZPZV<0>, ZPZV<0>, ZPZV<0>, ZPZV<0>, ZPZV<1>, ZPZV<232»; }; // NOLINT
 03543 template<> struct ConwayPolynomial<239, 8> { using ZPZ = aerobus::zpz<239>; using type = POLYV<ZPZV<1>, ZPZV<0>, ZPZV<0>, ZPZV<0>, ZPZV<1>, ZPZV<201>, ZPZV<202>, ZPZV<204>, ZPZV<54>, ZPZV<7»; }; //
```

```
03544 template<> struct ConwayPolynomial<239, 9> { using ZPZ = aerobus::zpz<239>; using type =
                             POLYV<ZPZV<1>, ZPZV<0>, ZPZV<0>, ZPZV<0>, ZPZV<0>, ZPZV<0>, ZPZV<3>, ZPZV<3>, ZPZV<2>, ZPZV<88>, ZPZV<3232»; };
                             // NOLINT
 03545 template<> struct ConwayPolynomial<239, 10> { using ZPZ = aerobus::zpz<239>; using type = POLYV<ZPZV<1>, ZPZV<0>, ZPZV<0>, ZPZV<0>, ZPZV<0>, ZPZV<57>, ZPZV<68>, ZPZV<226>, ZPZV<127>, ZPZV<108>, ZPZV<7»; }; // NOLINT
  03546 template<> struct ConwayPolynomial<239, 11> { using ZPZ = aerobus::zpz<239>; using type
                            POLYV<ZPZV<1>, ZPZV<0>, ZPZV<0
  03547 template<> struct ConwayPolynomial<239, 12> { using ZPZ = aerobus::zpz<239>; using type =
  POLYV<2PZV<1>, ZPZV<0>, ZPZV<0>, ZPZV<1>, ZPZV<235>, ZPZV<14>, ZPZV<113>, ZPZV<182>, ZPZV<101>, ZPZV<81>, ZPZV<216>, ZPZV<7»; }; // NOLINT

03548 template<> struct ConwayPolynomial<239, 13> { using ZPZ = aerobus::zpz<239>; using type =
                             POLÝV<ZPZV<1>, ZPZV<0>, ZPZV<0
ZPZV<0>, ZPZV<0>, ZPZV<2>, ZPZV<232»; }; // NOLINT
03549 template<> struct ConwayPolynomial<239, 17> { using ZPZ = aerobus::zpz<239>; using type =
                             POLYV<ZPZV<1>, ZPZV<0>, ZPZV<0
                             ZPZV<0>, ZPZV<0>, ZPZV<0>, ZPZV<0>, ZPZV<0>, ZPZV<0>, ZPZV<0>, ZPZV<232»; }; // NOLINT</pre>
  03550 template<> struct ConwayPolynomial<239, 19> { using ZPZ = aerobus::zpz<239>; using type
                             POLYV<ZPZV<1>, ZPZV<0>, ZPZV<0
                             ZPZV<0>, ZPZV<0>, ZPZV<0>, ZPZV<0>, ZPZV<0>, ZPZV<0>, ZPZV<0>, ZPZV<0>, ZPZV<0>, ZPZV<24>, ZPZV<24>, ZPZV<232»; }; //</pre>
                             NOLINT
 03551 template<> struct ConwayPolynomial<241, 1> { using ZPZ = aerobus::zpz<241>; using type =
                             POLYV<ZPZV<1>, ZPZV<234»; }; // NOLINT
  03552 template<> struct ConwayPolynomial<241, 2> { using ZPZ = aerobus::zpz<241>; using type =
                             POLYV<ZPZV<1>, ZPZV<238>, ZPZV<7»; }; // NOLINT
O3553 template<> struct ConwayPolynomial<241, 3> { using Zr2 = aerobus..rr = POLYV<ZPZV<1>, ZPZV<0>, ZPZV<1>, ZPZV<234»; }; // NOLINT

O3554 template<> struct ConwayPolynomial<241, 4> { using ZPZ = aerobus::zpz<241>; using type = POLYV<ZPZV<1>, ZPZV<0>, ZPZV<14>, ZPZV<152>, ZPZV<7»; }; // NOLINT

O3554 template<> struct ConwayPolynomial<241, 5> { using ZPZ = aerobus::zpz<241>; using type = Aerobus::zpz<241>; us
                             POLYV<ZPZV<1>, ZPZV<0>, ZPZV<0>, ZPZV<0>, ZPZV<7>, ZPZV<234»; }; // NOLINT
  03556 template<> struct ConwayPolynomial<241, 6> { using ZPZ = aerobus::zpz<241>; using type =
 POLYV<ZPZV<1>, ZPZV<0>, ZPZV<0>, ZPZV<83>, ZPZV<6>, ZPZV<5>, ZPZV<7»; }; // NOLINT 03557 template<> struct ConwayPolynomial<241, 7> { using ZPZ = aerobus::zpz<241>; using type =
                            POLYV<ZPZV<1>, ZPZV<0>, ZPZV<0>, ZPZV<0>, ZPZV<0>, ZPZV<0>, ZPZV<0>, ZPZV<2>, ZPZV<234»; };
                                                                                                                                                                                                                                                                                                                                                                                                                      // NOLINT
  03558 template<> struct ConwayPolynomial<241, 8> { using ZPZ = aerobus::zpz<241>; using type =
                             POLYV<ZPZV<1>, ZPZV<0>, ZPZV<0>, ZPZV<0>, ZPZV<1>, ZPZV<113>, ZPZV<212>, ZPZV<153>, ZPZV<13; }; //
 03559 template<> struct ConwayPolynomial<241, 9> { using ZPZ = aerobus::zpz<241>; using type =
                            POLYV<ZPZV<1>, ZPZV<0>, ZPZV<0>, ZPZV<0>, ZPZV<0>, ZPZV<0>, ZPZV<0>, ZPZV<236>, ZPZV<125>, ZPZV<234»;
                             }: // NOLINT
 03560 template<> struct ConwayPolynomial<241, 10> { using ZPZ = aerobus::zpz<241>; using type
                             POLYV<ZPZV<1>, ZPZV<0>, ZPZV<0>, ZPZV<0>, ZPZV<1>, ZPZV<2>, ZPZV<27>, ZPZV<145>, ZPZV<208>, ZPZV<55>,
                             ZPZV<7»; }; // NOLINT</pre>
 03561 template<> struct ConwayPolynomial<241, 11> { using ZPZ = aerobus::zpz<241>; using type =
                             POLYV<ZPZV<1>, ZPZV<0>, ZPZV<0
                             ZPZV<3>, ZPZV<234»: }: // NOLINT</pre>
03562 template<> struct ConwayPolynomial<241, 12> { using ZPZ = aerobus::zpz<241>; using type = POLYV<ZPZV<1>, ZPZV<0>, ZPZV<0>, ZPZV<0>, ZPZV<42>, ZPZV<10>, ZPZV<109>, ZPZV<168>, ZPZV<22>,
                             ZPZV<197>, ZPZV<17>, ZPZV<7»; }; // NOLINT</pre>
  03563 template<> struct ConwayPolynomial<241, 13> { using ZPZ = aerobus::zpz<241>; using type
POLYV<ZPZV<1>, ZPZV<0>, ZPZV<0 , ZPZV<0
                             ZPZV<0>, ZPZV<0>, ZPZV<0>, ZPZV<0>, ZPZV<0>, ZPZV<0>, ZPZV<0>, ZPZV<14>, ZPZV<234»; }; // NOLINT</pre>
  03565 template<> struct ConwayPolynomial<241, 19> { using ZPZ = aerobus::zpz<241>; using type =
                             POLYV<ZPZV<1>, ZPZV<0>, ZPZV<0
                              \texttt{ZPZV} < \texttt{0>, ZPZV} < \texttt{1>, ZPZV} < \texttt{234} *; }; // \texttt{200} 
                             NOLINT
 03566 template<> struct ConwayPolynomial<251, 1> { using ZPZ = aerobus::zpz<251>; using type =
                             POLYV<ZPZV<1>, ZPZV<245»; }; // NOLINT
  03567 template<> struct ConwayPolynomial<251, 2> { using ZPZ = aerobus::zpz<251>; using type =
                            POLYV<ZPZV<1>, ZPZV<242>, ZPZV<6»; }; // NOLINT
  03568 template<> struct ConwayPolynomial<251, 3> { using ZPZ = aerobus::zpz<251>; using type =
                            POLYV<ZPZV<1>, ZPZV<0>, ZPZV<3>, ZPZV<245»; }; // NOLINT
  03569 template<> struct ConwayPolynomial<251, 4> { using ZPZ = aerobus::zpz<251>; using type =
  POLYV<ZPZV<1>, ZPZV<3>, ZPZV<30>, ZPZV<200>, ZPZV<6»; }; // NOLINT
03570 template<> struct ConwayPolynomial<251, 5> { using ZPZ = aerobus::zpz<251>; using type =
                            POLYV<ZPZV<1>, ZPZV<0>, ZPZV<0>, ZPZV<0>, ZPZV<3>, ZPZV<3>, ZPZV<245»; }; // NOLINT
03571 template<> struct ConwayPolynomial<251, 6> { using ZPZ = aerobus::zpz<251>; using type = POLYV<ZPZV<1>, ZPZV<0>, ZPZV<1>, ZPZV<247>, ZPZV<151>, ZPZV<179>, ZPZV<6»; }; // NOLINT 03572 template<> struct ConwayPolynomial<251, 7> { using ZPZ = aerobus::zpz<251>; using type =
                            POLYV<ZPZV<1>, ZPZV<0>, ZPZV<0>, ZPZV<0>, ZPZV<0>, ZPZV<0>, ZPZV<8>, ZPZV<245»; }; // NOLINT
  03573 template<> struct ConwayPolynomial<251, 8> { using ZPZ = aerobus::zpz<251>; using type =
                             POLYV<ZPZV<1>, ZPZV<0>, ZPZV<0>, ZPZV<0>, ZPZV<7>, ZPZV<142>, ZPZV<215>, ZPZV<173>, ZPZV<6»; }; //
                            NOLINT
 03574 template<> struct ConwayPolynomial<251, 9> { using ZPZ = aerobus::zpz<251>; using type =
                            POLYV<ZPZV<1>, ZPZV<0>, ZPZV<0>, ZPZV<0>, ZPZV<0>, ZPZV<0>, ZPZV<0>, ZPZV<4>, ZPZV<187>, ZPZV<106>, ZPZV<24*;
                             }; // NOLINT
  03575 template<> struct ConwayPolynomial<251, 10> { using ZPZ = aerobus::zpz<251>; using type
                            POLYV<2PZV<1>, ZPZV<0>, ZPZV<0>, ZPZV<0>, ZPZV<1>, ZPZV<118>, ZPZV<110>, ZPZV<45>, ZPZV<34>,
                             {\tt ZPZV<149>}, {\tt ZPZV<6}; }; // NOLINT
 03576 template<> struct ConwayPolynomial<251, 11> { using ZPZ = aerobus::zpz<251>; using type = POLYV<ZPZV<1>, ZPZV<0>, ZPZV<0>,
```

```
ZPZV<26>, ZPZV<245»; };</pre>
 03577 template<> struct ConwayPolynomial<251, 12> { using ZPZ = aerobus::zpz<251>; using type =
                       POLYV<ZPZV<1>, ZPZV<0>, ZPZV<0>, ZPZV<0>, ZPZV<1>, ZPZV<192>, ZPZV<53>, ZPZV<20>, ZPZV<20>, ZPZV<15>,
ZPZV<201>, ZPZV<232>, ZPZV<6»; }; // NOLINT

03578 template<> struct ConwayPolynomial<251, 13> { using ZPZ = aerobus::zpz<251>; using type = POLYV<ZPZV<1>, ZPZV<0>, 
 03579 template<> struct ConwayPolynomial<251, 17> { using ZPZ = aerobus::zpz<251>; using type
                       POLYV<ZPZV<1>, ZPZV<0>, ZPZV<0
ZPZV<0>, ZPZV<0>, ZPZV<0>, ZPZV<0>, ZPZV<0>, ZPZV<0>, ZPZV<0>, ZPZV<0>, ZPZV<245>; // NOLINT 03580 template<> struct ConwayPolynomial<251, 19> { using ZPZ = aerobus::zpz<251>; using type =
                       POLYV<ZPZV<1>, ZPZV<0>, ZPZV<0
                        ZPZV<0>, ZPZV<2</pre>
 03581 template<> struct ConwayPolynomial<257, 1> { using ZPZ = aerobus::zpz<257>; using type =
                      POLYV<ZPZV<1>, ZPZV<254»; }; // NOLINT
03582 template<> struct ConwayPolynomial<257, 2> { using ZPZ = aerobus::zpz<257>; using type =
POLYV<ZPZV<1>, ZPZV<251>, ZPZV<251>, ZPZV<3»; }; /NOLINT

03583 template<> struct ConwayPolynomial<257, 3> { using ZPZ = aerobus::zpz<257>; using type =
                      POLYV<ZPZV<1>, ZPZV<0>, ZPZV<6>, ZPZV<254»; }; // NOLINT
 03584 template<> struct ConwayPolynomial<257, 4> { using ZPZ = aerobus::zpz<257>; using type =
POLYV<ZPZV<1>, ZPZV<0>, ZPZV<16>, ZPZV<187>, ZPZV<3»; }; // NOLINT
03585 template<> struct ConwayPolynomial<257, 5> { using ZPZ = aerobus::zpz<257>; using type =
POLYY<ZPZY<1>, ZPZV<0>, ZPZV<0>, ZPZV<0>, ZPZV<4>, ZPZV<254»; }; // NOLINT
03586 template<> struct ConwayPolynomial<257, 6> { using ZPZ = aerobus::zpz<257>; using type =
                      POLYV<ZPZV<1>, ZPZV<0>, ZPZV<3>, ZPZV<62>, ZPZV<18>, ZPZV<18>, ZPZV<138>, ZPZV<3»; ); // NOLINT
 03587 template<> struct ConwayPolynomial<257, 7> { using ZPZ = aerobus::zpz<257>; using type
                      POLYV<ZPZV<1>, ZPZV<0>, ZPZV<0>, ZPZV<0>, ZPZV<0>, ZPZV<0>, ZPZV<0>, ZPZV<31>, ZPZV<254»; }; // NOLINT
 03588 template<> struct ConwayPolynomial<257, 8> { using ZPZ = aerobus::zpz<257>; using type =
                       POLYV<ZPZV<1>, ZPZV<0>, ZPZV<0>, ZPZV<0>, ZPZV<0>, ZPZV<179>, ZPZV<140>, ZPZV<162>, ZPZV<3»; }; //
                      NOLINT
03589 template<> struct ConwayPolynomial<257, 9> { using ZPZ = aerobus::zpz<257>; using type
                       POLYV<ZPZV<1>, ZPZV<0>, ZPZV<0>, ZPZV<0>, ZPZV<0>, ZPZV<0>, ZPZV<1>, ZPZV<201>, ZPZV<50>, ZPZV<254»;
                        }; // NOLINT
03590 template<> struct ConwayPolynomial<257, 10> { using ZPZ = aerobus::zpz<257>; using type = POLYV<ZPZV<1>, ZPZV<0>, ZPZV<0>, ZPZV<0>, ZPZV<0>, ZPZV<20>, ZPZV<21>, ZPZV<225>, ZPZV<180>, ZPZV<20>,
                       ZPZV<3»; }; // NOLINT</pre>
03591 template<> struct ConwayPolynomial<257, 11> { using ZPZ = aerobus::zpz<257>; using type = POLYV<ZPZV<1>, ZPZV<0>, ZPZV<0 , 
                        ZPZV<40>, ZPZV<254»; }; // NOLINT</pre>
03592 template<> struct ConwayPolynomial<257, 12> { using ZPZ = aerobus::zpz<257>; using type =
POLYY<ZPZV<1>, ZPZV<0>, ZPZV<0>, ZPZV<0>, ZPZV<2>, ZPZV<13>, ZPZV<225>, ZPZV<215>, ZPZV<2173>, ZPZV<249>, ZPZV<148>, ZPZV<20>, ZPZV<3»; }; // NOLINT

03593 template<> struct ConwayPolynomial<257, 13> { using ZPZ = aerobus::zpz<257>; using type =
                      POLYV<ZPZV<1>, ZPZV<0>, ZPZV<0
03594 template<> struct ConwayPolynomial<257, 17> { using ZPZ = aerobus::zpz<257>; using type =
POLYV<ZPZV<1>, ZPZV<0>, ZPZV<0 , ZPZV<0
                        ZPZV<0>, ZPZV<10>, ZPZV<10>, ZPZV<254»; }; //</pre>
                       NOLINT
03596 template<> struct ConwayPolynomial<263, 1> { using ZPZ = aerobus::zpz<263>; using type =
                      POLYV<ZPZV<1>, ZPZV<258»; }; // NOLINT
 03597 template<> struct ConwayPolynomial<263, 2> { using ZPZ = aerobus::zpz<263>; using type =
                       POLYV<ZPZV<1>, ZPZV<261>, ZPZV<5»; }; // NOLINT
 03598 template<> struct ConwayPolynomial<263, 3> { using ZPZ = aerobus::zpz<263>; using type =
                       POLYV<ZPZV<1>, ZPZV<0>, ZPZV<14>, ZPZV<258»; }; // NOLINT
03599 template<> struct ConwayPolynomial<263, 4> { using ZPZ = aerobus::zpz<263>; using type = POLYV<ZPZV<1>, ZPZV<0>, ZPZV<5>, ZPZV<171>, ZPZV<5»; }; // NOLINT
03600 template<> struct ConwayPolynomial<263, 5> { using ZPZ = aerobus::zpz<263>; using type =
                      POLYV<ZPZV<1>, ZPZV<0>, ZPZV<0>, ZPZV<0>, ZPZV<13>, ZPZV<258»; }; // NOLINT
 03601 template<> struct ConwayPolynomial<263, 6> { using ZPZ = aerobus::zpz<263>; using type =
                      POLYV<ZPZV<1>, ZPZV<0>, ZPZV<1>, ZPZV<22>, ZPZV<25>, ZPZV<25>, ZPZV<5»; }; // NOLINT
 03602 template<> struct ConwayPolynomial<263, 7> { using ZPZ = aerobus::zpz<263>; using type =
                      POLYV<ZPZV<1>, ZPZV<0>, ZPZV<0>, ZPZV<0>, ZPZV<0>, ZPZV<0>, ZPZV<1>, ZPZV<2>, ZPZV<2>, ZPZV<1>, ZPZV<2>, ZPZV<2>, ZPZV<2>, ZPZV<3>, ZPZV<3
03603 template<> struct ConwayPolynomial<263, 8> { using ZPZ = aerobus::zpz<263>; using type =
                       POLYV<ZPZV<1>, ZPZV<0>, ZPZV<0>, ZPZV<0>, ZPZV<3>, ZPZV<227>, ZPZV<170>, ZPZV<7>, ZPZV<5»; }; //
                       NOLINT
03604 template<> struct ConwayPolynomial<263, 9> { using ZPZ = aerobus::zpz<263>; using type =
                       POLYV<ZPZV<1>, ZPZV<0>, ZPZV<0>, ZPZV<0>, ZPZV<0>, ZPZV<0>, ZPZV<6>, ZPZV<6>, ZPZV<6>, ZPZV<261>, ZPZV<29>, ZPZV<258»;
                        }; // NOLINT
03605 template<> struct ConwayPolynomial<263, 10> { using ZPZ = aerobus::zpz<263>; using type =
                      POLYV<ZPZV<1>, ZPZV<0>, ZPZV<0>, ZPZV<0>, ZPZV<245>, ZPZV<231>, ZPZV<198>, ZPZV<145>, ZPZV<119>, ZPZV<5»; }; // NOLINT
 03606 template<> struct ConwayPolynomial<263, 11> { using ZPZ = aerobus::zpz<263>; using type
                      POLYV<ZPZV<1>, ZPZV<0>, ZPZV<2>, ZPZV<258»; }; // NOLINT
03607 template<> struct ConwayPolynomial<263, 12> { using ZPZ = aerobus::zpz<263>; using type =
                       POLYV<ZPZV<1>, ZPZV<0>, ZPZV<0>, ZPZV<0>, ZPZV<0>, ZPZV<172>, ZPZV<174>, ZPZV<162>, ZPZV<252>,
                       ZPZV<47>, ZPZV<45>, ZPZV<180>, ZPZV<5»; }; // NOLINT</pre>
 03608 template<> struct ConwayPolynomial<269, 1> { using ZPZ = aerobus::zpz<269>; using type =
                      POLYV<ZPZV<1>, ZPZV<267»; }; // NOLINT
 03609 template<> struct ConwayPolynomial<269, 2> { using ZPZ = aerobus::zpz<269>; using type =
 POLYV<ZPZV<1>, ZPZV<268>, ZPZV<2»; }; // NOLINT
03610 template<> struct ConwayPolynomial<269, 3> { using ZPZ = aerobus::zpz<269>; using type =
```

```
POLYV<ZPZV<1>, ZPZV<0>, ZPZV<9>, ZPZV<267»; };
03611 template<> struct ConwayPolynomial<269, 4> { using ZPZ = aerobus::zpz<269>; using type = POLYV<ZPZV<1>, ZPZV<0>, ZPZV<8>, ZPZV<262>, ZPZV<2»; }; // NOLINT
03612 template<> struct ConwayPolynomial<269, 5> { using ZPZ = aerobus::zpz<269>; using type =
POLYV<ZPZV<1>, ZPZV<0>, ZPZV<0>, ZPZV<0>, ZPZV<12>, ZPZV<267»; }; // NOLINT
03613 template<> struct ConwayPolynomial<269, 6> { using ZPZ = aerobus::zpz<269>; using type =
               POLYV<ZPZV<1>, ZPZV<0>, ZPZV<1>, ZPZV<120>, ZPZV<101>, ZPZV<206>, ZPZV<2»; }; // NOLINT
03614 template<> struct ConwayPolynomial<269, 7> { using ZPZ = aerobus::zpz<269>;
                                                                                                                                                                                                                  using type
                POLYV<ZPZV<1>, ZPZV<0>, ZPZV<0>, ZPZV<0>, ZPZV<0>, ZPZV<0>, ZPZV<6>, ZPZV<6 , ZPZV<6
03615 template<> struct ConwayPolynomial<269, 8> { using ZPZ = aerobus::zpz<269>; using type =
                POLYV<ZPZV<1>, ZPZV<0>, ZPZV<0>, ZPZV<0>, ZPZV<4>, ZPZV<220>, ZPZV<131>, ZPZV<232>, ZPZV<2»; }; //
                NOLINT
03616 template<> struct ConwayPolynomial<269, 9> { using ZPZ = aerobus::zpz<269>; using type
                POLYV<ZPZV<1>, ZPZV<0>, ZPZV<0>, ZPZV<0>, ZPZV<0>, ZPZV<0>, ZPZV<2), ZPZV<214>, ZPZV<267>, ZPZV<267»;
                }; // NOLINT
03617 template<> struct ConwayPolynomial<269, 10> { using ZPZ = aerobus::zpz<269>; using type =
               POLYV<ZPZV<1>, ZPZV<0>, ZPZV<0>, ZPZV<0>, ZPZV<2>, ZPZV<264>, ZPZV<243>, ZPZV<186>, ZPZV<61>, ZPZV<10>, ZPZV<20>; }; // NOLINT
03618 template<> struct ConwayPolynomial<269, 11> { using ZPZ = aerobus::zpz<269>; using type = POLYV<ZPZV<1>, ZPZV<0>, ZPZV<0 , 
                ZPZV<20>, ZPZV<267»; }; // NOLINT</pre>
03619 template<> struct ConwayPolynomial<269, 12> { using ZPZ = aerobus::zpz<269>; using type =
POLYV<ZPZV<1>, ZPZV<0>, ZPZV<0>, ZPZV<0>, ZPZV<0>, ZPZV<126>, ZPZV<165>, ZPZV<165>, ZPZV<63>, ZPZV<215>, ZPZV<132>, ZPZV<180>, ZPZV<150>, ZPZV<2»; }; // NOLINT
03620 template<> struct ConwayPolynomial<271, 1> { using ZPZ = aerobus::zpz<271>; using type =
               POLYV<ZPZV<1>, ZPZV<265»; }; // NOLINT
03621 template<> struct ConwayPolynomial<271, 2> { using ZPZ = aerobus::zpz<271>; using type =
                POLYV<ZPZV<1>, ZPZV<269>, ZPZV<6»; }; // NOLINT
03622 template<> struct ConwayPolynomial<271, 3> { using ZPZ = aerobus::zpz<271>; using type =
POLYV<ZPZV<1>, ZPZV<0>, ZPZV<2>, ZPZV<265»; }; // NOLINT
03623 template<> struct ConwayPolynomial<271, 4> { using ZPZ = aerobus::zpz<271>; using type =
POLYV<ZPZV<1>, ZPZV<3>, ZPZV<3>, ZPZV<205>, ZPZV<6»; }; // NOLINT
03624 template<> struct ConwayPolynomial<271, 5> { using ZPZ = aerobus::zpz<271>; using type =
               POLYV<ZPZV<1>, ZPZV<0>, ZPZV<0>, ZPZV<0>, ZPZV<2>, ZPZV<265»; }; // NOLINT
03625 template<> struct ConwayPolynomial<271, 6> { using ZPZ = aerobus::zpz<271>; using type =
               POLYV<ZPZV<1>, ZPZV<0>, ZPZV<0>, ZPZV<207>, ZPZV<207>, ZPZV<81>, ZPZV<6»; }; // NOLINT
03626 template<> struct ConwayPolynomial<271, 7> { using ZPZ = aerobus::zpz<271>; using type =
               POLYV<ZPZV<1>, ZPZV<0>, ZPZV<0>, ZPZV<0>, ZPZV<0>, ZPZV<0>, ZPZV<0>, ZPZV<2>, ZPZV<265»; }; // NOLINT
03627 template<> struct ConwayPolynomial<271, 8> { using ZPZ = aerobus::zpz<271>; using type
                POLYV<ZPZV<1>, ZPZV<0>, ZPZV<0>, ZPZV<0>, ZPZV<0>, ZPZV<1>, ZPZV<199>, ZPZV<114>, ZPZV<69>, ZPZV<69»; }; //
                NOT.TNT
03628 template<> struct ConwayPolynomial<271, 9> { using ZPZ = aerobus::zpz<271>; using type =
               POLYV<ZPZV<1>, ZPZV<0>, ZPZV<0>, ZPZV<0>, ZPZV<0>, ZPZV<26>, ZPZV<266>, ZPZV<186>, ZPZV<265»;
                }; // NOLINT
03629 template<> struct ConwayPolynomial<271, 10> { using ZPZ = aerobus::zpz<271>; using type = POLYV<ZPZV<1>, ZPZV<0>, ZPZV<0>, ZPZV<0>, ZPZV<1>, ZPZV<133>, ZPZV<10>, ZPZV<256>, ZPZV<74>,
                ZPZV<126>, ZPZV<6>; }; // NOLINT
03630 template<> struct ConwayPolynomial<271, 11> { using ZPZ = aerobus::zpz<271>; using type = POLYV<ZPZV<1>, ZPZV<0>, 
03631 template<> struct ConwayPolynomial<271, 12> { using ZPZ = aerobus::zpz<271>; using type
                POLYV<ZPZV<1>, ZPZV<0>, ZPZV<0>, ZPZV<0>, ZPZV<0>, ZPZV<0>, ZPZV<162>, ZPZV<210>, ZPZV<116>, ZPZV<205>,
                ZPZV<237>, ZPZV<256>, ZPZV<130>, ZPZV<6»; }; // NOLINT</pre>
03632 template<> struct ConwayPolynomial<277, 1> { using ZPZ = aerobus::zpz<277>; using type =
               POLYV<ZPZV<1>, ZPZV<272»; }; // NOLINT
03633 template<> struct ConwayPolynomial<277, 2> { using ZPZ = aerobus::zpz<277>; using type =
                POLYV<ZPZV<1>, ZPZV<274>, ZPZV<5»; }; // NOLINT
03634 template<> struct ConwayPolynomial<277, 3> { using ZPZ = aerobus::zpz<277>; using type =
                POLYV<ZPZV<1>, ZPZV<0>, ZPZV<3>, ZPZV<272»; }; // NOLINT
03635 template<> struct ConwayPolynomial<277, 4> { using ZPZ = aerobus::zpz<277>; using type =
POLYV<ZPZV<1>, ZPZV<0>, ZPZV<1>, ZPZV<222>, ZPZV<5»; }; // NOLINT
03636 template<> struct ConwayPolynomial<277, 5> { using ZPZ = aerobus::zpz<277>; using type =
               POLYV<ZPZV<1>, ZPZV<0>, ZPZV<0>, ZPZV<0>, ZPZV<2>, ZPZV<272»; }; // NOLINT
03637 template<> struct ConwayPolynomial<277, 6> { using ZPZ = aerobus::zpz<277>; using type
               POLYV<ZPZV<1>, ZPZV<0>, ZPZV<1>, ZPZV<33>, ZPZV<9>, ZPZV<118>, ZPZV<5»; }; // NOLINT
03638 template<> struct ConwayPolynomial<277, 7> { using ZPZ = aerobus::zpz<277>; using type :
               POLYV<ZPZV<1>, ZPZV<0>, ZPZV<0>, ZPZV<0>, ZPZV<0>, ZPZV<0>, ZPZV<0>, ZPZV<9>, ZPZV<272»; };
03639 template<> struct ConwayPolynomial<277, 8> { using ZPZ = aerobus::2pz<277>; using type =
                POLYV<ZPZV<1>, ZPZV<0>, ZPZV<0>, ZPZV<0>, ZPZV<4>, ZPZV<187>, ZPZV<159>, ZPZV<176>, ZPZV<5»; }; //
                NOLINT
03640 template<> struct ConwayPolynomial<277, 9> { using ZPZ = aerobus::zpz<277>; using type =
                POLYV<ZPZV<1>, ZPZV<0>, ZPZV<0>, ZPZV<0>, ZPZV<0>, ZPZV<0>, ZPZV<4>, ZPZV<177>, ZPZV<110>, ZPZV<272»;
                }; // NOLINT
03641 template<> struct ConwayPolynomial<277, 10> { using ZPZ = aerobus::zpz<277>; using type
               POLYV<ZPZV<1>, ZPZV<0>, ZPZV<0>, ZPZV<0>, ZPZV<241>, ZPZV<253>, ZPZV<237>, ZPZV<241>, ZPZV<260>, ZPZV<5»; }; // NOLINT
03642 template<> struct ConwayPolynomial<277, 11> { using ZPZ = aerobus::zpz<277>; using type
               POLYV<ZPZV<1>, ZPZV<0>, ZPZV<0
03643 template<> struct ConwayPolynomial<277, 12> { using ZPZ = aerobus::zpz<277>; using type =
               POLYV<ZPZV<1>, ZPZV<0>, ZPZV<0>, ZPZV<0>, ZPZV<4>, ZPZV<183>, ZPZV<218>, ZPZV<240>, ZPZV<40>, ZPZV<40>, ZPZV<180>, ZPZV<218>, ZPZV<202>, ZPZV<5»; }; // NOLINT
03644 template<> struct ConwayPolynomial<281, 1> { using ZPZ = aerobus::zpz<281>; using type =
               POLYV<ZPZV<1>, ZPZV<278»; }; // NOLINT
03645 template<> struct ConwayPolynomial<281, 2> { using ZPZ = aerobus::zpz<281>; using type =
POLYV<ZPZV<1>, ZPZV<280>, ZPZV<3»; }; // NOLINT
03646 template<> struct ConwayPolynomial<281, 3> { using ZPZ = aerobus::zpz<281>; using type =
```

```
POLYV<ZPZV<1>, ZPZV<0>, ZPZV<1>, ZPZV<278»; };
03647 template<> struct ConwayPolynomial<281, 4> { using ZPZ = aerobus::zpz<281>; using type = POLYV<ZPZV<1>, ZPZV<0>, ZPZV<7>, ZPZV<176>, ZPZV<3»; }; // NOLINT
03648 template<> struct ConwayPolynomial<281, 5> { using ZPZ = aerobus::zpz<281>; using type =
               POLYV<ZPZV<1>, ZPZV<0>, ZPZV<0>, ZPZV<0>, ZPZV<5>, ZPZV<278»; }; // NOLINT
03649 template<> struct ConwayPolynomial<281, 6> { using ZPZ = aerobus::zpz<281>; using type =
               POLYV<ZPZV<1>, ZPZV<0>, ZPZV<1>, ZPZV<151>, ZPZV<13>, ZPZV<27>, ZPZV<3»; }; // NOLINT
03650 template<> struct ConwayPolynomial<281, 7> { using ZPZ = aerobus::zpz<281>; using type
               POLYV<ZPZV<1>, ZPZV<0>, ZPZV<0>, ZPZV<0>, ZPZV<0>, ZPZV<0>, ZPZV<2>, ZPZV<2>, ZPZV<2+, ZPZV<2
03651 template<> struct ConwayPolynomial<281, 8> { using ZPZ = aerobus::zpz<281>; using type =
               POLYV<ZPZV<1>, ZPZV<0>, ZPZV<0>, ZPZV<0>, ZPZV<4>, ZPZV<195>, ZPZV<279>, ZPZV<140>, ZPZV<3»; }; //
               NOLINT
03652 template<> struct ConwayPolynomial<281, 9> { using ZPZ = aerobus::zpz<281>; using type =
                POLYV<ZPZV<1>, ZPZV<0>, ZPZV<0>, ZPZV<0>, ZPZV<0>, ZPZV<0>, ZPZV<0>, ZPZV<6>, ZPZV<148>, ZPZV<70>, ZPZV<278»;
                }; // NOLINT
03653 template<> struct ConwayPolynomial<281, 10> { using ZPZ = aerobus::zpz<281>; using type
               POLYV<ZPZV<1>, ZPZV<0>, ZPZV<0>, ZPZV<0>, ZPZV<2>, ZPZV<258>, ZPZV<145>, ZPZV<13>, ZPZV<138>, ZPZV<191>, ZPZV<3»; }; // NOLINT
03654 template<> struct ConwayPolynomial<281, 11> { using ZPZ = aerobus::zpz<281>; using type = POLYV<ZPZV<1>, ZPZV<0>, 
                ZPZV<36>, ZPZV<278»; }; // NOLINT</pre>
03655 template<> struct ConwayPolynomial<281, 12> { using ZPZ = aerobus::zpz<281>; using type =
POLYV<ZPZV<1>, ZPZV<0>, ZPZV<0>, ZPZV<0>, ZPZV<0>, ZPZV<20>, ZPZV<202>, ZPZV<68>, ZPZV<103>, ZPZV<116>, ZPZV<58>, ZPZV<28>, ZPZV<191>, ZPZV<3»; }; // NOLINT
03656 template<> struct ConwayPolynomial<283, 1> { using ZPZ = aerobus::zpz<283>; using type =
               POLYV<ZPZV<1>, ZPZV<280»; }; // NOLINT
03657 template<> struct ConwayPolynomial<283, 2> { using ZPZ = aerobus::zpz<283>; using type =
               POLYV<ZPZV<1>, ZPZV<282>, ZPZV<3»; }; // NOLINT
03658 template<> struct ConwayPolynomial<283, 3> { using ZPZ = aerobus::zpz<283>; using type =
POLYV<ZPZV<1>, ZPZV<0>, ZPZV<3>, ZPZV<280»; }; // NOLINT
03659 template<> struct ConwayPolynomial<283, 4> { using ZPZ = aerobus::zpz<283>; using type =
POLYV<ZPZV<1>, ZPZV<5>, ZPZV<238>, ZPZV<3»; }; // NOLINT
03660 template<> struct ConwayPolynomial<283, 5> { using ZPZ = aerobus::zpz<283>; using type =
               POLYV<ZPZV<1>, ZPZV<0>, ZPZV<0>, ZPZV<0>, ZPZV<3>, ZPZV<280»; }; // NOLINT
03661 template<> struct ConwayPolynomial<283, 6> { using ZPZ = aerobus::zpz<283>; using type =
               POLYV<ZPZV<1>, ZPZV<0>, ZPZV<0>, ZPZV<199>, ZPZV<68>, ZPZV<73>, ZPZV<3»; }; // NOLINT
03662 template<> struct ConwayPolynomial<283, 7> { using ZPZ = aerobus::zpz<283>; using type
               POLYV<ZPZV<1>, ZPZV<0>, ZPZV<0>, ZPZV<0>, ZPZV<0>, ZPZV<0>, ZPZV<0>, ZPZV<8>, ZPZV<8>, ZPZV<280»; };
03663 template<> struct ConwayPolynomial<283, 8> { using ZPZ = aerobus::zpz<283>; using type
               POLYV<ZPZV<1>, ZPZV<0>, ZPZV<0>, ZPZV<0>, ZPZV<11>, ZPZV<179>, ZPZV<32>, ZPZV<232>, ZPZV<23»; }; //
               NOT.TNT
03664 template<> struct ConwayPolynomial<283, 9> { using ZPZ = aerobus::zpz<283>; using type =
               POLYV<ZPZV<1>, ZPZV<0>, ZPZV<0>, ZPZV<0>, ZPZV<0>, ZPZV<0>, ZPZV<0>, ZPZV<136>, ZPZV<65>, ZPZV<280»;
               }; // NOLINT
03665 template<> struct ConwayPolynomial<283, 10> { using ZPZ = aerobus::zpz<283>; using type = POLYV<ZPZV<1>, ZPZV<0>, ZPZV<0>, ZPZV<0>, ZPZV<3>, ZPZV<271>, ZPZV<185>, ZPZV<185>, ZPZV<100>,
               ZPZV<219>, ZPZV<3»; }; // NOLINT</pre>
03666 template<> struct ConwayPolynomial<283, 11> { using ZPZ = aerobus::zpz<283>; using type = POLYV<ZPZV<1>, ZPZV<0>, 
03667 template<> struct ConwayPolynomial<283, 12> { using ZPZ = aerobus::zpz<283>; using type =
               POLYV<ZPZV<1>, ZPZV<0>, ZPZV<0>, ZPZV<0>, ZPZV<0>, ZPZV<20>, ZPZV<20>, ZPZV<8>, ZPZV<96>, ZPZV<229>, ZPZV<49>, ZPZV<14>, ZPZV<56>, ZPZV<3»; }; // NOLINT
03668 template<> struct ConwayPolynomial<293, 1> { using ZPZ = aerobus::zpz<293>; using type =
               POLYV<ZPZV<1>, ZPZV<291»; }; // NOLINT
03669 template<> struct ConwayPolynomial<293, 2> { using ZPZ = aerobus::zpz<293>; using type =
               POLYV<ZPZV<1>, ZPZV<292>, ZPZV<2»; }; // NOLINT
03670 template<> struct ConwayPolynomial<293, 3> { using ZPZ = aerobus::zpz<293>; using type =
               POLYV<ZPZV<1>, ZPZV<0>, ZPZV<2>, ZPZV<291»; }; // NOLINT
03671 template<> struct ConwayPolynomial<293, 4> { using ZPZ = aerobus::zpz<293>; using type =
POLYV<ZPZV<1>, ZPZV<3>, ZPZV<166>, ZPZV<2»; }; // NOLINT

03672 template<> struct ConwayPolynomial<293, 5> { using ZPZ = aerobus::zpz<293>; using type =
               POLYV<ZPZV<1>, ZPZV<0>, ZPZV<0>, ZPZV<0>, ZPZV<2>, ZPZV<291»; }; // NOLINT
03673 template<> struct ConwayPolynomial<293, 6> { using ZPZ = aerobus::zpz<293>; using type =
               POLYV<ZPZV<1>, ZPZV<0>, ZPZV<1>, ZPZV<128>, ZPZV<210>, ZPZV<260>, ZPZV<2»; }; // NOLINT
03674 template<> struct ConwayPolynomial<293, 7> { using ZPZ = aerobus::zpz<293>; using type =
               POLYV<ZPZV<1>, ZPZV<0>, ZPZV<0>, ZPZV<0>, ZPZV<0>, ZPZV<0>, ZPZV<0>, ZPZV<8>, ZPZV<291»; };
03675 template<> struct ConwayPolynomial<293, 8> { using ZPZ = aerobus::zpz<293>; using type =
               POLYV<ZPZV<1>, ZPZV<0>, ZPZV<0>, ZPZV<0>, ZPZV<29>, ZPZV<175>, ZPZV<195>, ZPZV<239>, ZPZV<239>, ZPZV<29>; //
               NOLINT
03676 template<> struct ConwayPolynomial<293, 9> { using ZPZ = aerobus::zpz<293>; using type =
               POLYV<ZPZV<1>, ZPZV<0>, ZPZV<0>, ZPZV<0>, ZPZV<0>, ZPZV<0>, ZPZV<0>, ZPZV<0>, ZPZV<208>, ZPZV<190>, ZPZV<291»;
                }; // NOLINT
03677 template<> struct ConwayPolynomial<293, 10> { using ZPZ = aerobus::zpz<293>; using type
               POLYV<ZPZV<1>, ZPZV<0>, ZPZV<0>, ZPZV<0>, ZPZV<28>, ZPZV<46>, ZPZV<184>, ZPZV<24>, ZPZV<24>, ZPZV<28; }; // NOLINT
03678 template<> struct ConwayPolynomial<293, 11> { using ZPZ = aerobus::zpz<293>; using type
               POLYV<ZPZV<1>, ZPZV<0>, ZPZV<0
03679 template<> struct ConwayPolynomial<293, 12> { using ZPZ = aerobus::zpz<293>; using type =
               POLYV<ZPZV<1>, ZPZV<0>, ZPZV<0>, ZPZV<0>, ZPZV<0>, ZPZV<210>, ZPZV<125>, ZPZV<212>,
               ZPZV<167>, ZPZV<144>, ZPZV<157>, ZPZV<2»; }; // NOLINT</pre>
03680 template<> struct ConwayPolynomial<307, 1> { using ZPZ = aerobus::zpz<307>; using type =
               POLYV<ZPZV<1>, ZPZV<302»; }; // NOLINT
03681 template<> struct ConwayPolynomial<307, 2> { using ZPZ = aerobus::zpz<307>; using type =
POLYV<ZPZV<1>, ZPZV<306>, ZPZV<5»; }; // NOLINT
03682 template<> struct ConwayPolynomial<307, 3> { using ZPZ = aerobus::zpz<307>; using type =
```

```
POLYV<ZPZV<1>, ZPZV<0>, ZPZV<7>, ZPZV<302»; };
03683 template<> struct ConwayPolynomial<307, 4> { using ZPZ = aerobus::zpz<307>; using type = POLYV<ZPZV<1>, ZPZV<0>, ZPZV<2>, ZPZV<239>, ZPZV<5>; }; // NOLINT
03684 template<> struct ConwayPolynomial<307, 5> { using ZPZ = aerobus::zpz<307>; using type =
      POLYV<ZPZV<1>, ZPZV<0>, ZPZV<0>, ZPZV<0>, ZPZV<5>, ZPZV<302»; }; // NOLINT
03685 template<> struct ConwayPolynomial<307, 6> { using ZPZ = aerobus::zpz<307>; using type =
      POLYV<ZPZV<1>, ZPZV<0>, ZPZV<0>, ZPZV<213>, ZPZV<172>, ZPZV<61>, ZPZV<5»; }; // NOLINT
03686 template<> struct ConwayPolynomial<307, 7> { using ZPZ = aerobus::zpz<307>; using type
       POLYV<ZPZV<1>, ZPZV<0>, ZPZV<0>, ZPZV<0>, ZPZV<0>, ZPZV<0>, ZPZV<6>, ZPZV<6>, ZPZV<302*; };
03687 template<> struct ConwayPolynomial<307, 8> { using ZPZ = aerobus::zpz<307>; using type =
       POLYV<ZPZV<1>, ZPZV<0>, ZPZV<0>, ZPZV<0>, ZPZV<0>, ZPZV<2>, ZPZV<232>, ZPZV<231>, ZPZV<131>, ZPZV<5»; }; //
       NOLINT
03688 template<> struct ConwayPolynomial<307, 9> { using ZPZ = aerobus::zpz<307>; using type =
       POLYV<ZPZV<1>, ZPZV<0>, ZPZV<0>, ZPZV<0>, ZPZV<0>, ZPZV<0>, ZPZV<1>, ZPZV<165>, ZPZV<70>, ZPZV<302»;
       }; // NOLINT
03689 template<> struct ConwayPolynomial<311, 1> { using ZPZ = aerobus::zpz<311>; using type =
      POLYV<ZPZV<1>, ZPZV<294»; }; // NOLINT
03690 template<> struct ConwayPolynomial<311, 2> { using ZPZ = aerobus::zpz<311>; using type =
POLYY<ZPZY<1>, ZPZY<310>, ZPZY<17»; }; // NOLINT

03691 template<> struct ConwayPolynomial<311, 3> { using ZPZ = aerobus::zpz<311>; using type =
       POLYV<ZPZV<1>, ZPZV<0>, ZPZV<3>, ZPZV<294»; }; // NOLINT
03692 template<> struct ConwayPolynomial<311, 4> { using ZPZ = aerobus::zpz<311>; using type =
POLYV<ZPZV<1>, ZPZV<0>, ZPZV<3>, ZPZV<163>, ZPZV<17»; }; // NOLINT
03693 template<> struct ConwayPolynomial<311, 5> { using ZPZ = aerobus::zpz<311>; using type =
      POLYV<ZPZV<1>, ZPZV<0>, ZPZV<0>, ZPZV<0>, ZPZV<1>, ZPZV<294»; }; // NOLINT
03694 template<> struct ConwayPolynomial<311, 6> { using ZPZ = aerobus::zpz<311>; using type =
       POLYV<ZPZV<1>, ZPZV<0>, ZPZV<1>, ZPZV<27>, ZPZV<167>, ZPZV<152>, ZPZV<17»; }; // NOLINT
03695 template<> struct ConwayPolynomial<311, 7> { using ZPZ = aerobus::zpz<311>; using type =
      POLYV<ZPZV<1>, ZPZV<0>, ZPZV<0>, ZPZV<0>, ZPZV<0>, ZPZV<0>, ZPZV<0>, ZPZV<0>, ZPZV<10>, ZPZV<294»; };
03696 template<> struct ConwayPolynomial<311, 8> { using ZPZ = aerobus::zpz<311>; using type =
       POLYV<ZPZV<1>, ZPZV<0>, ZPZV<0>, ZPZV<0>, ZPZV<19>, ZPZV<162>, ZPZV<118>, ZPZV<2>, ZPZV<17»; }; //
       NOLINT
03697 template<> struct ConwayPolynomial<311, 9> { using ZPZ = aerobus::zpz<311>; using type =
       POLYV<ZPZV<1>, ZPZV<0>, ZPZV<0>, ZPZV<0>, ZPZV<0>, ZPZV<0>, ZPZV<294»;
       }; // NOLINT
03698 template<> struct ConwayPolynomial<313, 1> { using ZPZ = aerobus::zpz<313>; using type =
      POLYV<ZPZV<1>, ZPZV<303»; }; // NOLINT
03699 template<> struct ConwayPolynomial<313, 2> { using ZPZ = aerobus::zpz<313>; using type =
      POLYV<ZPZV<1>, ZPZV<310>, ZPZV<10»; }; // NOLINT
03700 template<> struct ConwayPolynomial<313, 3> { using ZPZ = aerobus::zpz<313>; using type =
      POLYV<ZPZV<1>, ZPZV<0>, ZPZV<2>, ZPZV<303»; }; // NOLINT
03701 template<> struct ConwayPolynomial<313, 4> { using ZPZ = aerobus::zpz<313>; using type =
POLYV<ZPZV<1>, ZPZV<0>, ZPZV<8>, ZPZV<239>, ZPZV<10»; }; // NOLINT
03702 template<> struct ConwayPolynomial<313, 5> { using ZPZ = aerobus::zpz<313>; using type =
      POLYV<ZPZV<1>, ZPZV<0>, ZPZV<0>, ZPZV<0>, ZPZV<7>, ZPZV<303»; }; // NOLINT
03703 template<> struct ConwayPolynomial<313, 6> { using ZPZ = aerobus::zpz<313>; using type =
      POLYV<ZPZV<1>, ZPZV<0>, ZPZV<0>, ZPZV<196>, ZPZV<213>, ZPZV<253>, ZPZV<10»; }; // NOLINT
03704 template<> struct ConwayPolynomial<313, 7> { using ZPZ = aerobus::zpz<313>; using type =
      POLYV<ZPZV<1>, ZPZV<0>, ZPZV<0>, ZPZV<0>, ZPZV<0>, ZPZV<0>, ZPZV<4>, ZPZV<303»; };
                                                                                                      // NOLINT
03705 template<> struct ConwayPolynomial<313, 8> { using ZPZ = aerobus::zpz<313>; using type =
       POLYV<ZPZV<1>, ZPZV<0>, ZPZV<0>, ZPZV<0>, ZPZV<8>, ZPZV<306>, ZPZV<99>, ZPZV<106>, ZPZV<10»; }; //
03706 template<> struct ConwayPolynomial<313, 9> { using ZPZ = aerobus::zpz<313>; using type =
       POLYV<ZPZV<1>, ZPZV<0>, ZPZV<0>, ZPZV<0>, ZPZV<0>, ZPZV<0>, ZPZV<8>, ZPZV<267>, ZPZV<300>, ZPZV<303»;
       ); // NOLINT
03707 template<> struct ConwayPolynomial<317, 1> { using ZPZ = aerobus::zpz<317>; using type =
       POLYV<ZPZV<1>, ZPZV<315»; }; // NOLINT
03708 template<> struct ConwayPolynomial<317, 2> { using ZPZ = aerobus::zpz<317>; using type =
POLYV<ZPZV<1>, ZPZV<313>, ZPZV<2»; }; // NOLINT
03709 template<> struct ConwayPolynomial<317, 3> { using ZPZ = aerobus::zpz<317>; using type =
POLYV<ZPZV<1>, ZPZV<0>, ZPZV<7>, ZPZV<315»; }; // NOLINT

03710 template<> struct ConwayPolynomial<317, 4> { using ZPZ = aerobus::zpz<317>; using type =
POLYV<ZPZV<1>, ZPZV<0>, ZPZV<6>, ZPZV<178>, ZPZV<2»; }; // NOLINT
03711 template<> struct ConwayPolynomial<317, 5> { using ZPZ = aerobus::zpz<317>; using type =
       POLYV<ZPZV<1>, ZPZV<0>, ZPZV<0>, ZPZV<0>, ZPZV<2>, ZPZV<315»; // NOLINT
03712 template<> struct ConwayPolynomial<317, 6> { using ZPZ = aerobus::zpz<317>; using type =
      POLYV<ZPZV<1>, ZPZV<0>, ZPZV<2>, ZPZV<195>, ZPZV<156>, ZPZV<4>, ZPZV<2»; }; // NOLINT
03713 template<> struct ConwayPolynomial<317, 7> { using ZPZ = aerobus: zpz<317>; using type = POLYV<ZPZV<1>, ZPZV<0>, ZPZV<0>, ZPZV<0>, ZPZV<0>, ZPZV<0>, ZPZV<7>, ZPZV<315»; }; // NOLINT
03714 template<> struct ConwayPolynomial<317, 8> { using ZPZ = aerobus::zpz<317>; using type
       POLYV<ZPZV<1>, ZPZV<0>, ZPZV<0>, ZPZV<0>, ZPZV<1>, ZPZV<207>, ZPZV<85>, ZPZV<31>, ZPZV<2»; }; //
       NOLINT
03715 template<> struct ConwayPolynomial<317, 9> { using ZPZ = aerobus::zpz<317>; using type = POLYV<ZPZV<1>, ZPZV<0>, ZPZV<0>, ZPZV<0>, ZPZV<0>, ZPZV<0>, ZPZV<2>, ZPZV<284>, ZPZV<296>, ZPZV<315»;
       }; // NOLINT
03716 template<> struct ConwayPolynomial<331, 1> { using ZPZ = aerobus::zpz<331>; using type =
       POLYV<ZPZV<1>, ZPZV<328»; }; // NOLINT
03717 template<> struct ConwayPolynomial<331, 2> { using ZPZ = aerobus::zpz<331>; using type =
POLYV<ZPZV<1>, ZPZV<326, ZPZV<3»; }; // NOLINT
03718 template<> struct ConwayPolynomial<331, 3> { using ZPZ = aerobus::zpz<331>; using type =
      POLYV<ZPZV<1>, ZPZV<0>, ZPZV<1>, ZPZV<328»; }; // NOLINT
03719 template<> struct ConwayPolynomial<331, 4> { using ZPZ = aerobus::zpz<331>; using type =
POLYV<ZPZV<1>, ZPZV<3>, ZPZV<3>, ZPZV<290>, ZPZV<3»; }; // NOLINT
03720 template<> struct ConwayPolynomial<331, 5> { using ZPZ = aerobus::zpz<331>; using type =
      03721 template<> struct ConwayPolynomial<331, 6> { using ZPZ = aerobus::zpz<331>; using type = POLYV<ZPZV<1>, ZPZV<0>, ZPZV<1>, ZPZV<283>, ZPZV<205>, ZPZV<159>, ZPZV<3»; }; // NOLINT
```

```
03722 template<> struct ConwayPolynomial<331, 7> { using ZPZ = aerobus::zpz<331>; using type
       POLYV<ZPZV<1>, ZPZV<0>, ZPZV<0>, ZPZV<0>, ZPZV<0>, ZPZV<0>, ZPZV<5>, ZPZV<328»; };
03723 template<> struct ConwayPolynomial<331, 8> { using ZPZ = aerobus::zpz<331>; using type =
       POLYV<ZPZV<1>, ZPZV<0>, ZPZV<0>, ZPZV<0>, ZPZV<0>, ZPZV<4>, ZPZV<249>, ZPZV<308>, ZPZV<78>, ZPZV<3»; }; //
       NOLINT
03724 template<> struct ConwayPolynomial<331, 9> { using ZPZ = aerobus::zpz<331>; using type =
       POLYV<ZPZV<1>, ZPZV<0>, ZPZV<0>, ZPZV<0>, ZPZV<0>, ZPZV<0>, ZPZV<0>, ZPZV<0>, ZPZV<194>, ZPZV<210>, ZPZV<328»;
        }; // NOLINT
03725 template<> struct ConwayPolynomial<337, 1> { using ZPZ = aerobus::zpz<337>; using type =
       POLYV<ZPZV<1>, ZPZV<327»; }; // NOLINT
03726 template<> struct ConwayPolynomial<337, 2> { using ZPZ = aerobus::zpz<337>; using type =
POLYV<ZPZV<1>, ZPZV<332>, ZPZV<10»; }; // NOLINT
03727 template<> struct ConwayPolynomial<337, 3> { using ZPZ = aerobus::zpz<337>; using type =
       POLYV<ZPZV<1>, ZPZV<0>, ZPZV<2>, ZPZV<327»; }; // NOLINT
03728 template<> struct ConwayPolynomial<337, 4> { using ZPZ = aerobus::zpz<337>; using type =
POLYV<ZPZV<1>, ZPZV<0>, ZPZV<25>, ZPZV<24>, ZPZV<10»; }; // NOLINT
03729 template<> struct ConwayPolynomial<337, 5> { using ZPZ = aerobus::zpz<337>; using type =
POLYV<ZPZV<1>, ZPZV<0>, ZPZV<0>, ZPZV<0>, ZPZV<2>, ZPZV<2>, ZPZV<327»; }; // NOLINT
03730 template<> struct ConwayPolynomial<337, 6> { using ZPZ = aerobus::zpz<337>; using type =
       POLYV<ZPZV<1>, ZPZV<0>, ZPZV<0>, ZPZV<216>, ZPZV<127>, ZPZV<109>, ZPZV<10»; }; //
03731 template<> struct ConwayPolynomial<337, 7> { using ZPZ = aerobus::zpz<337>; using type =
       POLYV<ZPZV<1>, ZPZV<0>, ZPZV<0>, ZPZV<0>, ZPZV<0>, ZPZV<0>, ZPZV<0>, ZPZV<5>, ZPZV<327»; };
03732 template<> struct ConwayPolynomial<337, 8> { using ZPZ = aerobus::zpz<337>; using type =
       POLYV<ZPZV<1>, ZPZV<0>, ZPZV<0>, ZPZV<0>, ZPZV<0>, ZPZV<331>, ZPZV<246>, ZPZV<251>, ZPZV<10»; }; //
       NOLINT
03733 template<> struct ConwayPolynomial<337, 9> { using ZPZ = aerobus::zpz<337>; using type =
       POLYV<ZPZV<1>, ZPZV<0>, ZPZV<0>, ZPZV<0>, ZPZV<0>, ZPZV<0>, ZPZV<1>, ZPZV<148>, ZPZV<148>, ZPZV<98>, ZPZV<327»;
        }; // NOLINT
03734 template<> struct ConwayPolynomial<347, 1> { using ZPZ = aerobus::zpz<347>; using type =
       POLYV<ZPZV<1>, ZPZV<345»; }; // NOLINT
03735 template<> struct ConwayPolynomial<347, 2> { using ZPZ = aerobus::zpz<347>; using type =
POLYV<ZPZV<1>, ZPZV<343>, ZPZV<2»; }; // NOLINT
03736 template<> struct ConwayPolynomial<347, 3> { using ZPZ = aerobus::zpz<347>; using type =
       POLYV<ZPZV<1>, ZPZV<0>, ZPZV<9>, ZPZV<345»; }; // NOLINT
03737 template<> struct ConwayPolynomial<347, 4> { using ZPZ = aerobus::zpz<347>; using type =
POLYV<ZPZV<1>, ZPZV<0>, ZPZV<13>, ZPZV<295>, ZPZV<2»; }; // NOLINT
03738 template<> struct ConwayPolynomial<347, 5> { using ZPZ = aerobus::zpz<347>; using type =
       POLYV<ZPZV<1>, ZPZV<0>, ZPZV<0>, ZPZV<3>, ZPZV<345»; }; // NOLINT
03739 template<> struct ConwayPolynomial<347, 6> { using ZPZ = aerobus::zpz<347>; using type =
       POLYV<ZPZV<1>, ZPZV<0>, ZPZV<3>, ZPZV<343>, ZPZV<26>, ZPZV<56>, ZPZV<2»; }; // NOLINT
03740 template<> struct ConwayPolynomial<347, 7> { using ZPZ = aerobus::zpz<347>; using type =
       POLYV<ZPZV<1>, ZPZV<0>, ZPZV<0>, ZPZV<0>, ZPZV<0>, ZPZV<0>, ZPZV<4>, ZPZV<345»; };
03741 template<> struct ConwayPolynomial<347, 8> { using ZPZ = aerobus::zpz<347>; using type = POLYV<ZPZV<1>, ZPZV<0>, ZPZV<0>, ZPZV<0>, ZPZV<1>, ZPZV<187>, ZPZV<213>, ZPZV<2117>, ZPZV<2»; }; //
03742 template<> struct ConwayPolynomial<347, 9> { using ZPZ = aerobus::zpz<347>; using type =
       POLYV<ZPZV<1>, ZPZV<0>, ZPZV<0>, ZPZV<0>, ZPZV<0>, ZPZV<0>, ZPZV<2>, ZPZV<235>, ZPZV<252>, ZPZV<345»;
       }; // NOLINT
03743 template<> struct ConwayPolynomial<349, 1> { using ZPZ = aerobus::zpz<349>; using type =
       POLYV<ZPZV<1>, ZPZV<347»; }; // NOLINT
03744 template<> struct ConwayPolynomial<349, 2> { using ZPZ = aerobus::zpz<349>; using type =
       POLYV<ZPZV<1>, ZPZV<348>, ZPZV<2»; }; // NOLINT
03745 template<> struct ConwayPolynomial<349, 3> { using ZPZ = aerobus::zpz<349>; using type =
POLYY<ZPZV<1>, ZPZV<4>, ZPZV<4>, ZPZV<347»; }; // NOLINT

03746 template<> struct ConwayPolynomial<349, 4> { using ZPZ = aerobus::zpz<349>; using type = POLYV<ZPZV<1>, ZPZV<0>, ZPZV<3>, ZPZV<279>, ZPZV<2»; }; // NOLINT

03747 template<> struct ConwayPolynomial<349, 5> { using ZPZ = aerobus::zpz<349>; using type =
       POLYV<ZPZV<1>, ZPZV<0>, ZPZV<0>, ZPZV<0>, ZPZV<2>, ZPZV<347»; }; // NOLINT
03748 template<> struct ConwayPolynomial<349, 6> { using ZPZ = aerobus::zpz<349>; using type =
POLYV<ZPZV<1>, ZPZV<0>, ZPZV<0>, ZPZV<135>, ZPZV<177>, ZPZV<316>, ZPZV<2»; }; // NOLINT 03749 template<> struct ConwayPolynomial<349, 7> { using ZPZ = aerobus::zpz<349>; using type =
       POLYV<ZPZV<1>, ZPZV<0>, ZPZV<0>, ZPZV<0>, ZPZV<0>, ZPZV<0>, ZPZV<10>, ZPZV<347»; }; // NOLINT
03750 template<> struct ConwayPolynomial<349, 8> { using ZPZ = aerobus::zpz<349>; using type =
       POLYV<ZPZV<1>, ZPZV<0>, ZPZV<0>, ZPZV<0>, ZPZV<0>, ZPZV<308>, ZPZV<328>, ZPZV<268>, ZPZV<268), ZPZV<20; }; //
       NOT.TNT
03751 template<> struct ConwayPolynomial<349, 9> { using ZPZ = aerobus::zpz<349>; using type =
       POLYV<ZPZV<1>, ZPZV<0>, ZPZV<0>, ZPZV<0>, ZPZV<34>, ZPZV<36>, ZPZV<290>, ZPZV<130>, ZPZV<347»;
       }; // NOLINT
03752 template<> struct ConwayPolynomial<353, 1> { using ZPZ = aerobus::zpz<353>; using type =
       POLYV<ZPZV<1>, ZPZV<350»; }; // NOLINT
03753 template<> struct ConwayPolynomial<353, 2> { using ZPZ = aerobus::zpz<353>; using type =
       POLYV<ZPZV<1>, ZPZV<348>, ZPZV<3»; }; // NOLINT
03754 template<> struct ConwayPolynomial<353, 3> { using ZPZ = aerobus::zpz<353>; using type =
       POLYV<ZPZV<1>, ZPZV<0>, ZPZV<3>, ZPZV<350»; }; // NOLINT
03755 template<> struct ConwayPolynomial<353, 4> { using ZPZ = aerobus::zpz<353>; using type = POLYV<ZPZV<1>, ZPZV<0>, ZPZV<0>, ZPZV<199>, ZPZV<3»; }; // NOLINT
03756 template<> struct ConwayPolynomial<353, 5> { using ZPZ = aerobus::zpz<353>; using type =
       POLYV<ZPZV<1>, ZPZV<0>, ZPZV<0>, ZPZV<0>, ZPZV<1>, ZPZV<350»; }; // NOLINT
03757 template<> struct ConwayPolynomial<353, 6> { using ZPZ = aerobus::zpz<353>; using type =
POLYV<ZPZV<1>, ZPZV<0>, ZPZV<1>, ZPZV<2>, ZPZV<226>, ZPZV<226>, ZPZV<295>, ZPZV<3»; }; // NOLINT

03758 template<> struct ConwayPolynomial<353, 7> { using ZPZ = aerobus::zpz<353>; using type = POLYV<ZPZV<1>, ZPZV<0>, ZPZV<0>, ZPZV<0>, ZPZV<0>, ZPZV<0>, ZPZV<6>, ZPZV<35»; }; // NOLINT
03759 template<> struct ConwayPolynomial<353, 8> { using ZPZ = aerobus::zpz<353>; using type =
       POLYV<ZPZV<1>, ZPZV<0>, ZPZV<0>, ZPZV<0>, ZPZV<1>, ZPZV<182>, ZPZV<26>, ZPZV<37>, ZPZV<37>, ZPZV<39; }; //
03760 template<> struct ConwayPolynomial<353, 9> { using ZPZ = aerobus::zpz<353>; using type = POLYV<ZPZV<1>, ZPZV<0>, ZPZV<0>, ZPZV<0>, ZPZV<0>, ZPZV<0>, ZPZV<2>, ZPZV<319>, ZPZV<49>, ZPZV<350»;
```

```
}; // NOLINT
03761 template<> struct ConwayPolynomial<359, 1> { using ZPZ = aerobus::zpz<359>; using type =
       POLYV<ZPZV<1>, ZPZV<352»; }; // NOLINT
03762 template<> struct ConwayPolynomial<359, 2> { using ZPZ = aerobus::zpz<359>; using type =
POLYV<ZPZV<1>, ZPZV<358>, ZPZV<7»; }; // NOLINT
03763 template<> struct ConwayPolynomial<359, 3> { using ZPZ = aerobus::zpz<359>; using type =
       POLYV<ZPZV<1>, ZPZV<0>, ZPZV<3>, ZPZV<352»; }; // NOLINT
03764 template<> struct ConwayPolynomial<359, 4> { using ZPZ = aerobus::zpz<359>; using type =
POLYV<ZPZV<1>, ZPZV<0>, ZPZV<2>, ZPZV<229>, ZPZV<7»; }; // NOLINT
03765 template<> struct ConwayPolynomial<359, 5> { using ZPZ = aerobus::zpz<359>; using type =
       POLYV<ZPZV<1>, ZPZV<0>, ZPZV<0>, ZPZV<0>, ZPZV<1>, ZPZV<352»; }; // NOLINT
03766 template<> struct ConwayPolynomial<359, 6> { using ZPZ = aerobus::zpz<359>; using type =
POLYV<ZPZV<1>, ZPZV<4>, ZPZV<309>, ZPZV<327>, ZPZV<37>; ; // NOLINT 03767 template<> struct ConwayPolynomial<359, 7> { using ZPZ = aerobus::zpz<359>; using type =
       POLYV<ZPZV<1>, ZPZV<0>, ZPZV<0>, ZPZV<0>, ZPZV<0>, ZPZV<0>, ZPZV<0>, ZPZV<1>, ZPZV<352»; );
03768 template<> struct ConwayPolynomial<359, 8> { using ZPZ = aerobus::zpz<359>; using type =
       POLYV<ZPZV<1>, ZPZV<0>, ZPZV<0>, ZPZV<0>, ZPZV<0>, ZPZV<301>, ZPZV<143>, ZPZV<271>, ZPZV<7»; }; //
       NOLINT
03769 template<> struct ConwayPolynomial<359, 9> { using ZPZ = aerobus::zpz<359>; using type =
       POLYV<ZPZV<1>, ZPZV<0>, ZPZV<0>, ZPZV<0>, ZPZV<0>, ZPZV<0>, ZPZV<0>, ZPZV<0>, ZPZV<0>, ZPZV<356>, ZPZV<165>, ZPZV<352»;
       }; // NOLINT
03770 template<> struct ConwayPolynomial<367, 1> { using ZPZ = aerobus::zpz<367>; using type =
       POLYV<ZPZV<1>, ZPZV<361»; }; // NOLINT
03771 template<> struct ConwayPolynomial<367, 2> { using ZPZ = aerobus::zpz<367>; using type =
POLYV<ZPZV<1>, ZPZV<366>, ZPZV<6»; }; // NOLINT
03772 template<> struct ConwayPolynomial<367, 3> { using ZPZ = aerobus::zpz<367>; using type =
       POLYV<ZPZV<1>, ZPZV<0>, ZPZV<10>, ZPZV<361»; }; // NOLINT
03773 template<> struct ConwayPolynomial<367, 4> { using ZPZ = aerobus::zpz<367>; using type =
POLYV<ZPZV<1>, ZPZV<0>, ZPZV<3>, ZPZV<295>, ZPZV<6»; }; // NOLINT

03774 template<> struct ConwayPolynomial<367, 5> { using ZPZ = aerobus::zpz<367>; using type = POLYV<ZPZV<1>, ZPZV<0>, ZPZV<0>, ZPZV<0>, ZPZV<3>, ZPZV<36h; }; // NOLINT
03775 template<> struct ConwayPolynomial<367, 6> { using ZPZ = aerobus::zpz<367>; using type
       POLYV<ZPZV<1>, ZPZV<0>, ZPZV<0>, ZPZV<222>, ZPZV<321>, ZPZV<324>, ZPZV<6»; }; // NOLINT
03776 template<> struct ConwayPolynomial<367, 7> { using ZPZ = aerobus::zpz<367>; using type
       POLYV<ZPZV<1>, ZPZV<0>, ZPZV<0>, ZPZV<0>, ZPZV<0>, ZPZV<0>, ZPZV<0>, ZPZV<13>, ZPZV<361»; };
03777 template<> struct ConwayPolynomial<367, 8> { using ZPZ = aerobus::zpz<367>; using type :
       POLYV<ZPZV<1>, ZPZV<0>, ZPZV<0>, ZPZV<0>, ZPZV<2>, ZPZV<335>, ZPZV<282>, ZPZV<50>, ZPZV<6»; }; //
03778 template<> struct ConwayPolynomial<367, 9> { using ZPZ = aerobus::zpz<367>; using type
       POLYV<ZPZV<1>, ZPZV<0>, ZPZV<0>, ZPZV<0>, ZPZV<0>, ZPZV<0>, ZPZV<0>, ZPZV<15>, ZPZV<213>, ZPZV<268>, ZPZV<361»;
       }; // NOLINT
03779 template<> struct ConwayPolynomial<373, 1> { using ZPZ = aerobus::zpz<373>; using type =
       POLYV<ZPZV<1>, ZPZV<371»; }; // NOLINT
03780 template<> struct ConwayPolynomial<373, 2> { using ZPZ = aerobus::zpz<373>; using type =
       POLYV<ZPZV<1>, ZPZV<369>, ZPZV<2»; }; // NOLINT
03781 template<> struct ConwayPolynomial<373, 3> { using ZPZ = aerobus::zpz<373>; using type =
       POLYV<ZPZV<1>, ZPZV<0>, ZPZV<5>, ZPZV<371»; }; // NOLINT
03782 template<> struct ConwayPolynomial<373, 4> { using ZPZ = aerobus::zpz<373>; using type =
POLYV<ZPZV<1>, ZPZV<0>, ZPZV<15>, ZPZV<304>, ZPZV<2»; }; // NOLINT

03783 template<> struct ConwayPolynomial<373, 5> { using ZPZ = aerobus::zpz<373>; using type = POLYV<ZPZV<1>, ZPZV<0>, ZPZV<0>, ZPZV<0>, ZPZV<2>, ZPZV<371»; }; // NOLINT
03784 template<> struct ConwayPolynomial<373, 6> { using ZPZ = aerobus::zpz<373>; using type =
       POLYV<ZPZV<1>, ZPZV<0>, ZPZV<0>, ZPZV<126>, ZPZV<83>, ZPZV<108>, ZPZV<2»; }; // NOLINT
03785 template<> struct ConwayPolynomial<373, 7> { using ZPZ = aerobus::zpz<373>; using type =
       POLYV<ZPZV<1>, ZPZV<0>, ZPZV<0>, ZPZV<0>, ZPZV<0>, ZPZV<0>, ZPZV<0>, ZPZV<7>, ZPZV<371»; };
03786 template<> struct ConwayPolynomial<373, 8> { using ZPZ = aerobus::zpz<373>; using type
       POLYV<ZPZV<1>, ZPZV<0>, ZPZV<0>, ZPZV<0>, ZPZV<0>, ZPZV<2), ZPZV<203>, ZPZV<219>, ZPZV<66>, ZPZV<2»; }; //
03787 template<> struct ConwayPolynomial<373, 9> { using ZPZ = aerobus::zpz<373>; using type =
       POLYV<ZPZV<1>, ZPZV<0>, ZPZV<0>, ZPZV<0>, ZPZV<0>, ZPZV<0>, ZPZV<14>, ZPZV<238>, ZPZV<370>, ZPZV<371»;
       }; // NOLINT
03788 template<> struct ConwayPolynomial<379, 1> { using ZPZ = aerobus::zpz<379>; using type =
       POLYV<ZPZV<1>, ZPZV<377»; }; // NOLINT
03789 template<> struct ConwayPolynomial<379, 2> { using ZPZ = aerobus::zpz<379>; using type =
       POLYV<ZPZV<1>, ZPZV<374>, ZPZV<2»; }; // NOLINT
03790 template<> struct ConwayPolynomial<379, 3> { using ZPZ = aerobus::zpz<379>; using type =
POLYY<ZPZV<1>, ZPZV<0>, ZPZV<5>, ZPZV<377»; }; // NOLINT

03791 template<> struct ConwayPolynomial<379, 4> { using ZPZ = aerobus::zpz<379>; using type = POLYV<ZPZV<1>, ZPZV<0>, ZPZV<2>, ZPZV<327>, ZPZV<2»; }; // NOLINT

03792 template<> struct ConwayPolynomial<379, 5> { using ZPZ = aerobus::zpz<379>; using type =
       POLYV<ZPZV<1>, ZPZV<0>, ZPZV<0>, ZPZV<0>, ZPZV<4>, ZPZV<377»; }; // NOLINT
03793 template<> struct ConwayPolynomial<379, 6> { using ZPZ = aerobus::zpz<379>; using type =
POLYV<ZPZV<1>, ZPZV<0>, ZPZV<0>, ZPZV<374>, ZPZV<364>, ZPZV<246, ZPZV<2*; }; // NOLINT 03794 template<> struct ConwayPolynomial<379, 7> { using ZPZ = aerobus::zpz<379>; using type =
       POLYV<ZPZV<1>, ZPZV<0>, ZPZV<0>, ZPZV<0>, ZPZV<0>, ZPZV<0>, ZPZV<0>, ZPZV<14>, ZPZV<377»; }; // NOLINT
03795 template<> struct ConwayPolynomial<379, 8> { using ZPZ = aerobus::zpz<379>; using type =
       POLYV<ZPZV<1>, ZPZV<0>, ZPZV<0>, ZPZV<0>, ZPZV<13>, ZPZV<210>, ZPZV<194>, ZPZV<173>, ZPZV<2»; }; //
03796 template<> struct ConwayPolynomial<379, 9> { using ZPZ = aerobus::zpz<379>; using type = POLYV<ZPZV<1>, ZPZV<0>, ZPZV<0>, ZPZV<0>, ZPZV<0>, ZPZV<1>, ZPZV<362>, ZPZV<369>, ZPZV<377»;
       }; // NOLINT
03797 template<> struct ConwayPolynomial<383, 1> { using ZPZ = aerobus::zpz<383>; using type =
       POLYV<ZPZV<1>, ZPZV<378»; }; // NOLINT
03798 template<> struct ConwayPolynomial<383, 2> { using ZPZ = aerobus::zpz<383>; using type =
POLYV<ZPZV<1>, ZPZV<382>, ZPZV<5»; }; // NOLINT
03799 template<> struct ConwayPolynomial<383, 3> { using ZPZ = aerobus::zpz<383>; using type = POLYV<ZPZV<1>, ZPZV<0>, ZPZV<1>, ZPZV<378»; }; // NOLINT
```

```
03800 template<> struct ConwayPolynomial<383, 4> { using ZPZ = aerobus::zpz<383>; using type =
POLYY<ZPZV<1>, ZPZV<0>, ZPZV<309, ZPZV<309, ZPZV<5»; }; // NOLINT
03801 template<> struct ConwayPolynomial<383, 5> { using ZPZ = aerobus::zpz<383>; using type =
      POLYV<ZPZV<1>, ZPZV<0>, ZPZV<0>, ZPZV<0>, ZPZV<1>, ZPZV<378»; }; // NOLINT
03802 template<> struct ConwayPolynomial<383, 6> { using ZPZ = aerobus::zpz<383>; using type = POLYV<ZPZV<1>, ZPZV<0>, ZPZV<1>, ZPZV<69>, ZPZV<8>, ZPZV<158>, ZPZV<5»; }; // NOLINT
03803 template<> struct ConwayPolynomial<383, 7> { using ZPZ = aerobus::zpz<383>; using type
      POLYV<ZPZV<1>, ZPZV<0>, ZPZV<0>, ZPZV<0>, ZPZV<0>, ZPZV<0>, ZPZV<0>, ZPZV<6>, ZPZV<6>, ZPZV<378»; };
03804 template<> struct ConwayPolynomial<383, 8> { using ZPZ = aerobus::zpz<383>; using type =
      POLYV<ZPZV<1>, ZPZV<0>, ZPZV<0>, ZPZV<0>, ZPZV<10>, ZPZV<281>, ZPZV<332>, ZPZV<296>, ZPZV<5»; }; //
      NOLINT
03805 template<> struct ConwayPolynomial<383, 9> { using ZPZ = aerobus::zpz<383>; using type = POLYV<ZPZV<1>, ZPZV<0>, ZPZV<0>, ZPZV<0>, ZPZV<0>, ZPZV<2>, ZPZV<137>, ZPZV<76>, ZPZV<378»;
       }; // NOLINT
03806 template<> struct ConwayPolynomial<389, 1> { using ZPZ = aerobus::zpz<389>; using type =
      POLYV<ZPZV<1>, ZPZV<387»; }; // NOLINT
03807 template<> struct ConwayPolynomial<389, 2> { using ZPZ = aerobus::zpz<389>; using type =
POLYV<ZPZV<1>, ZPZV<379>, ZPZV<2»; }; // NOLINT

03808 template<> struct ConwayPolynomial<389, 3> { using ZPZ = aerobus::zpz<389>; using type =
      POLYV<ZPZV<1>, ZPZV<0>, ZPZV<2>, ZPZV<387»; }; // NOLINT
POLYV<ZPZV<1>, ZPZV<0>, ZPZV<0>, ZPZV<0>, ZPZV<4>, ZPZV<387»; }; // NOLINT
03811 template<> struct ConwayPolynomial<389, 6> { using ZPZ = aerobus::zpz<389>; using type =
      POLYV<ZPZV<1>, ZPZV<0>, ZPZV<1>, ZPZV<218>, ZPZV<339>, ZPZV<255>, ZPZV<2»; }; // NOLINT
03812 template<> struct ConwayPolynomial<389, 7> { using ZPZ = aerobus::zpz<389>;
                                                                                         using type
      POLYV<ZPZV<1>, ZPZV<0>, ZPZV<0>, ZPZV<0>, ZPZV<0>, ZPZV<0>, ZPZV<24>, ZPZV<387»; }; // NOLINT
03813 template<> struct ConwayPolynomial<389, 8> { using ZPZ = aerobus::zpz<389>; using type =
      POLYV<ZPZV<1>, ZPZV<0>, ZPZV<0>, ZPZV<0>, ZPZV<2>, ZPZV<351>, ZPZV<19>, ZPZV<290>, ZPZV<2»; }; //
      NOLINT
03814 template<> struct ConwayPolynomial<389, 9> { using ZPZ = aerobus::zpz<389>; using type
      POLYV<ZPZV<1>, ZPZV<0>, ZPZV<0>, ZPZV<0>, ZPZV<0>, ZPZV<0>, ZPZV<2>, ZPZV<258>, ZPZV<308>, ZPZV<387»;
       }; // NOLINT
03815 template<> struct ConwayPolynomial<397, 1> { using ZPZ = aerobus::zpz<397>; using type =
      POLYV<ZPZV<1>, ZPZV<392»; }; // NOLINT
03816 template<> struct ConwayPolynomial<397, 2> { using ZPZ = aerobus::zpz<397>; using type =
      POLYV<ZPZV<1>, ZPZV<392>, ZPZV<5»; }; // NOLINT
03817 template<> struct ConwayPolynomial<397, 3> { using ZPZ = aerobus::zpz<397>; using type =
      POLYV<ZPZV<1>, ZPZV<0>, ZPZV<2>, ZPZV<392»; }; // NOLINT
03818 template<> struct ConwayPolynomial<397, 4> { using ZPZ = aerobus::zpz<397>; using type =
POLYV<ZPZV<1>, ZPZV<0>, ZPZV<12>, ZPZV<363>, ZPZV<5»; }; // NOLINT
03819 template<> struct ConwayPolynomial<397, 5> { using ZPZ = aerobus::zpz<397>; using type =
      POLYV<ZPZV<1>, ZPZV<0>, ZPZV<0>, ZPZV<7>, ZPZV<392»; }; // NOLINT
03820 template<> struct ConwayPolynomial<397, 6> { using ZPZ = aerobus::zpz<397>; using type =
      POLYV<ZPZV<1>, ZPZV<0>, ZPZV<0>, ZPZV<382>, ZPZV<274>, ZPZV<287>, ZPZV<5»; };
03821 template<> struct ConwayPolynomial<397, 7> { using ZPZ = aerobus::zpz<397>; using type =
      POLYV<ZPZV<1>, ZPZV<0>, ZPZV<0>, ZPZV<0>, ZPZV<0>, ZPZV<0>, ZPZV<1>, ZPZV<392»; }; // NOLINT
03822 template<> struct ComwayPolynomial<397, 8> { using ZPZ = aerobus::zpz<397>; using type = POLYV<ZPZV<1>, ZPZV<0>, ZPZV<0>, ZPZV<0>, ZPZV<1>, ZPZV<25>, ZPZV<25>, ZPZV<203>, ZPZV<5»; }; //
03823 template<> struct ConwayPolynomial<397, 9> { using ZPZ = aerobus::zpz<397>; using type =
      POLYV<ZPZV<1>, ZPZV<0>, ZPZV<0>, ZPZV<0>, ZPZV<0>, ZPZV<0>, ZPZV<6>, ZPZV<166>, ZPZV<166>, ZPZV<252>, ZPZV<392»;
       }; // NOLINT
03824 template<> struct ConwayPolynomial<401, 1> { using ZPZ = aerobus::zpz<401>; using type =
      POLYV<ZPZV<1>, ZPZV<398»; }; // NOLINT
03825 template<> struct ConwayPolynomial<401, 2> { using ZPZ = aerobus::zpz<401>; using type =
      POLYV<ZPZV<1>, ZPZV<396>, ZPZV<3»; }; // NOLINT
03826 template<> struct ConwayPolynomial<401, 3> { using ZPZ = aerobus::zpz<401>; using type =
      POLYV<ZPZV<1>, ZPZV<0>, ZPZV<3>, ZPZV<398»; }; // NOLINT
POLYV<ZPZV<1>, ZPZV<0>, ZPZV<0>, ZPZV<0>, ZPZV<4>, ZPZV<398»; }; // NOLINT
03829 template<> struct ConwayPolynomial<401, 6> { using ZPZ = aerobus::zpz<401>; using type =
       \texttt{POLYV} < \texttt{ZPZV} < 1>, \ \texttt{ZPZV} < 0>, \ \texttt{ZPZV} < 4>, \ \texttt{ZPZV} < 115>, \ \texttt{ZPZV} < 81>, \ \texttt{ZPZV} < 51>, \ \texttt{ZPZV} < 3»; \ \}; \ \ // \ \texttt{NOLINT} 
03830 template<> struct ConwayPolynomial<401, 7> { using ZPZ = aerobus::zpz<401>; using type
      POLYV<ZPZV<1>, ZPZV<0>, ZPZV<0>, ZPZV<0>, ZPZV<0>, ZPZV<0>, ZPZV<5>, ZPZV<5>, ZPZV<398»; };
                                                                                                  // NOLINT
03831 template<> struct ConwayPolynomial<401, 8> { using ZPZ = aerobus::zpz<401>; using type =
      POLYV<ZPZV<1>, ZPZV<0>, ZPZV<0>, ZPZV<0>, ZPZV<0>, ZPZV<3», ZPZV<164>, ZPZV<164>, ZPZV<3»; }; //
03832 template<> struct ConwayPolynomial<401, 9> { using ZPZ = aerobus::zpz<401>; using type =
      POLYV<ZPZV<1>, ZPZV<0>, ZPZV<0>, ZPZV<0>, ZPZV<0>, ZPZV<0>, ZPZV<1>, ZPZV<1>, ZPZV<199>, ZPZV<158>, ZPZV<398»;
       }; // NOLINT
03833 template<> struct ConwayPolynomial<409, 1> { using ZPZ = aerobus::zpz<409>; using type =
      POLYV<ZPZV<1>, ZPZV<388»; }; // NOLINT
03834 template<> struct ConwayPolynomial<409, 2> { using ZPZ = aerobus::zpz<409>; using type =
POLYV<ZPZV<1>, ZPZV<404, ZPZV<21»; }; // NOLINT
03835 template<> struct ConwayPolynomial<409, 3> { using ZPZ = aerobus::zpz<409>; using type =
      POLYV<ZPZV<1>, ZPZV<0>, ZPZV<3>, ZPZV<388»; }; // NOLINT
03836 template<> struct ConwayPolynomial<409, 4> { using ZPZ = aerobus::zpz<409>; using type =
      POLYV<ZPZV<1>, ZPZV<0>, ZPZV<12>, ZPZV<407>, ZPZV<21»; }; // NOLINT
03837 template<> struct ConwayPolynomial<409, 5> { using ZPZ = aerobus::zpz<409>; using type =
      POLYV<ZPZV<1>, ZPZV<0>, ZPZV<0>, ZPZV<0>, ZPZV<5>, ZPZV<388»; }; // NOLINT
03838 template<> struct ConwayPolynomial<409, 6> { using ZPZ = aerobus::zpz<409>; using type =
POLYV<ZPZV<1>, ZPZV<0>, ZPZV<0>, ZPZV<372>, ZPZV<53>, ZPZV<364>, ZPZV<21»; }; // NOLINT 03839 template<> struct ConwayPolynomial<409, 7> { using ZPZ = aerobus::zpz<409>; using type =
```

```
POLYV<ZPZV<1>, ZPZV<0>, ZPZV<0>, ZPZV<0>, ZPZV<0>, ZPZV<7>, ZPZV<7>, ZPZV<388»; };
03840 template<> struct ConwayPolynomial<409, 8> { using ZPZ = aerobus::zpz<409>; using type
           POLYV<ZPZV<1>, ZPZV<0>, ZPZV<0>, ZPZV<0>, ZPZV<3>, ZPZV<256>, ZPZV<69>, ZPZV<396>, ZPZV<31»; }; //
           NOLINT
03841 template<> struct ConwayPolynomial<409, 9> { using ZPZ = aerobus::zpz<409>; using type = POLYV<ZPZV<1>, ZPZV<0>, ZPZV<0>, ZPZV<0>, ZPZV<0>, ZPZV<0>, ZPZV<0>, ZPZV<318>, ZPZV<318*, ZPZV<318*, ZPZV<318*, ZPZV<318*, ZPZV<318*, Z
03842 template<> struct ConwayPolynomial<419, 1> { using ZPZ = aerobus::zpz<419>; using type =
           POLYV<ZPZV<1>, ZPZV<417»; }; // NOLINT
03843 template<> struct ConwayPolynomial<419, 2> { using ZPZ = aerobus::zpz<419>; using type =
POLYV<ZPZV<1>, ZPZV<418>, ZPZV<2»; }; // NOLINT

03844 template<> struct ConwayPolynomial<419, 3> { using ZPZ = aerobus::zpz<419>; using type =
           POLYV<ZPZV<1>, ZPZV<0>, ZPZV<11>, ZPZV<417»; }; // NOLINT
03845 template<> struct ConwayPolynomial<419, 4> { using ZPZ = aerobus::zpz<419>; using type =
POLYV<ZPZV<1>, ZPZV<0>, ZPZV<4>, ZPZV<373>, ZPZV<2»; }; // NOLINT

03846 template<> struct ConwayPolynomial<419, 5> { using ZPZ = aerobus::zpz<419>; using type = POLYV<ZPZV<1>, ZPZV<0>, ZPZV<0>, ZPZV<0>, ZPZV<4>, ZPZV<417»; }; // NOLINT
03847 template<> struct ConwayPolynomial<419, 6> { using ZPZ = aerobus::zpz<419>; using type =
           POLYV<ZPZV<1>, ZPZV<0>, ZPZV<2>, ZPZV<411>, ZPZV<33>, ZPZV<257>, ZPZV<2»; }; // NOLINT
03848 template<> struct ConwayPolynomial<419, 7> { using ZPZ = aerobus::zpz<419>; using type
           POLYV<ZPZV<1>, ZPZV<0>, ZPZV<0>, ZPZV<0>, ZPZV<0>, ZPZV<0>, ZPZV<0>, ZPZV<4>, ZPZV<417»; };
03849 template<> struct ConwayPolynomial<419, 8> { using ZPZ = aerobus::zpz<419>; using type =
           POLYV<ZPZV<1>, ZPZV<0>, ZPZV<0>, ZPZV<0>, ZPZV<8>, ZPZV<234>, ZPZV<388>, ZPZV<151>, ZPZV<2»; }; //
           NOLINT
03850 template<> struct ConwayPolynomial<419, 9> { using ZPZ = aerobus::zpz<419>; using type =
           POLYV<ZPZV<1>, ZPZV<0>, ZPZV<0>, ZPZV<0>, ZPZV<0>, ZPZV<0>, ZPZV<0>, ZPZV<0>, ZPZV<0>, ZPZV<0>, ZPZV<3, ZPZV<3
            }; // NOLINT
03851 template<> struct ConwayPolynomial<421, 1> { using ZPZ = aerobus::zpz<421>; using type =
           POLYV<ZPZV<1>, ZPZV<419»; }; // NOLINT
03852 template<> struct ConwayPolynomial<421, 2> { using ZPZ = aerobus::zpz<421>; using type = POLYV<ZPZV<1>, ZPZV<417>, ZPZV<2»; }; // NOLINT
03853 template<> struct ConwayPolynomial<421, 3> { using ZPZ = aerobus::zpz<421>; using type =
           POLYV<ZPZV<1>, ZPZV<0>, ZPZV<2>, ZPZV<419»; }; // NOLINT
03854 template<> struct ConwayPolynomial<421, 4> { using ZPZ = aerobus::zpz<421>; using type =
POLYV<ZPZV<1>, ZPZV<0>, ZPZV<10>, ZPZV<257>, ZPZV<28; }; // NOLINT
03855 template<> struct ConwayPolynomial<421, 5> { using ZPZ = aerobus::zpz<421>; using type =
           POLYV<ZPZV<1>, ZPZV<0>, ZPZV<0>, ZPZV<0>, ZPZV<15>, ZPZV<419»; }; // NOLINT
03856 template<> struct ConwayPolynomial<421, 6> { using ZPZ = aerobus::zpz<421>; using type =
           POLYV<ZPZV<1>, ZPZV<0>, ZPZV<0>, ZPZV<111>, ZPZV<342>, ZPZV<41>, ZPZV<2»; }; // NOLINT
03857 template<> struct ConwayPolynomial<421, 7> { using ZPZ = aerobus::zpz<421>; using type =
           POLYV<ZPZV<1>, ZPZV<0>, ZPZV<0>, ZPZV<0>, ZPZV<0>, ZPZV<0>, ZPZV<21>, ZPZV<419»; };
03858 template<> struct ConwayPolynomial<421, 8> { using ZPZ = aerobus::zpz<421>; using type =
           POLYV<ZPZV<1>, ZPZV<0>, ZPZV<0>, ZPZV<0>, ZPZV<5>, ZPZV<389>, ZPZV<312>, ZPZV<77>, ZPZV<2»; }; //
           NOLINT
03859 template<> struct ConwayPolynomial<421, 9> { using ZPZ = aerobus::zpz<421>; using type
           POLYV<ZPZV<1>, ZPZV<0>, ZPZV<0>, ZPZV<0>, ZPZV<0>, ZPZV<0>, ZPZV<18>, ZPZV<394>, ZPZV<145>, ZPZV<419»;
            }; // NOLINT
03860 template<> struct ConwayPolynomial<431, 1> { using ZPZ = aerobus::zpz<431>; using type =
           POLYV<ZPZV<1>, ZPZV<424»; }; // NOLINT
03861 template<> struct ConwayPolynomial<431, 2> { using ZPZ = aerobus::zpz<431>; using type =
           POLYV<ZPZV<1>, ZPZV<430>, ZPZV<7»; }; // NOLINT
03862 template<> struct ConwayPolynomial<431, 3> { using ZPZ = aerobus::zpz<431>; using type =
           POLYV<ZPZV<1>, ZPZV<0>, ZPZV<2>, ZPZV<424»; }; // NOLINT
03863 template<> struct ConwayPolynomial<431, 4> { using ZPZ = aerobus::zpz<431>; using type = POLYV<ZPZV<1>, ZPZV<0>, ZPZV<2>, ZPZV<323>, ZPZV<7»; }; // NOLINT
03864 template<> struct ConwayPolynomial<431, 5> { using ZPZ = aerobus::zpz<431>; using type =
           POLYV<ZPZV<1>, ZPZV<0>, ZPZV<0>, ZPZV<7>, ZPZV<424»; }; // NOLINT
03865 template<> struct ConwayPolynomial<431, 6> { using ZPZ = aerobus::zpz<431>; using type =
           POLYV<ZPZV<1>, ZPZV<0>, ZPZV<4>, ZPZV<161>, ZPZV<202>, ZPZV<182>, ZPZV<7»; }; // NOLINT
03866 template<> struct ConwayPolynomial<431, 7> { using ZPZ = aerobus::zpz<431>; using type =
           POLYV<ZPZV<1>, ZPZV<0>, ZPZV<0>, ZPZV<0>, ZPZV<0>, ZPZV<0>, ZPZV<0>, ZPZV<424»; };
                                                                                                                                                                      // NOLINT
03867 template<> struct ConwayPolynomial<431, 8> { using ZPZ = aerobus::zpz<431>; using type = POLYV<ZPZV<1>, ZPZV<0>, ZPZV<0>, ZPZV<5>, ZPZV<243>, ZPZV<286>, ZPZV<115>, ZPZV<7»; }; //
           NOLINT
03868 template<> struct ConwayPolynomial<431, 9> { using ZPZ = aerobus::zpz<431>; using type =
           POLYV<ZPZV<1>, ZPZV<0>, ZPZV<0>, ZPZV<0>, ZPZV<0>, ZPZV<0>, ZPZV<2>, ZPZV<71>, ZPZV<329>, ZPZV<424*;
            }; // NOLINT
03869 template<> struct ConwayPolynomial<433, 1> { using ZPZ = aerobus::zpz<433>; using type =
           POLYV<ZPZV<1>, ZPZV<428»; }; // NOLINT
03870 template<> struct ConwayPolynomial<433, 2> { using ZPZ = aerobus::zpz<433>; using type =
           POLYV<ZPZV<1>, ZPZV<432>, ZPZV<5»; }; // NOLINT
03871 template<> struct ConwayPolynomial<433, 3> { using ZPZ = aerobus::zpz<433>; using type =
           POLYV<ZPZV<1>, ZPZV<0>, ZPZV<1>, ZPZV<428»; }; // NOLINT
03872 template<> struct ConwayPolynomial<433, 4> { using ZPZ = aerobus::zpz<433>; using type = POLYV<ZPZV<1>, ZPZV<0>, ZPZV<6>, ZPZV<402>, ZPZV<5>; }; // NOLINT
03873 template<> struct ConwayPolynomial<433, 5> { using ZPZ = aerobus::zpz<433>; using type =
           POLYV<ZPZV<1>, ZPZV<0>, ZPZV<0>, ZPZV<5>, ZPZV<428»; }; // NOLINT
03874 template<> struct ConwayPolynomial<433, 6> { using ZPZ = aerobus::zpz<433>; using type =
POLYV<ZPZV<1>, ZPZV<0>, ZPZV<0>, ZPZV<244>, ZPZV<353>, ZPZV<360>, ZPZV<5»; }; // NOLINT 03875 template<> struct ConwayPolynomial<433, 7> { using ZPZ = aerobus::zpz<433>; using type =
           POLYV<ZPZV<1>, ZPZV<0>, ZPZV<0>, ZPZV<0>, ZPZV<0>, ZPZV<0>, ZPZV<6>, ZPZV<6>, ZPZV<428»; }; // NOLINT
03876 template<> struct ConwayPolynomial<433, 8> { using ZPZ = aerobus::zpz<433>; using type
           POLYV<ZPZV<1>, ZPZV<0>, ZPZV<0>, ZPZV<0>, ZPZV<10>, ZPZV<347>, ZPZV<32>, ZPZV<39>, ZPZV<5»; }; //
03877 template<> struct ConwayPolynomial<433, 9> { using ZPZ = aerobus::zpz<433>; using type =
           POLYV<ZPZV<1>, ZPZV<0>, ZPZV<0>, ZPZV<0>, ZPZV<0>, ZPZV<0>, ZPZV<27>, ZPZV<232>, ZPZV<45>, ZPZV<428»;
           }; // NOLINT
```

```
03878 template<> struct ConwayPolynomial<439, 1> { using ZPZ = aerobus::zpz<439>; using type =
      POLYV<ZPZV<1>, ZPZV<424»; }; // NOLINT
03879 template<> struct ConwayPolynomial<439, 2> { using ZPZ = aerobus::zpz<439>; using type =
POLYV<ZPZV<1>, ZPZV<436>, ZPZV<15»; }; // NOLINT

03880 template<> struct ConwayPolynomial<439, 3> { using ZPZ = aerobus::zpz<439>; using type =
      POLYV<ZPZV<1>, ZPZV<0>, ZPZV<3>, ZPZV<424»; }; // NOLINT
03881 template<> struct ConwayPolynomial<439, 4> { using ZPZ = aerobus::zpz<439>; using type =
      POLYV<ZPZV<1>, ZPZV<0>, ZPZV<0>, ZPZV<323>, ZPZV<15»; }; // NOLINT
03882 template<> struct ConwayPolynomial<439, 5> { using ZPZ = aerobus::zpz<439>; using type =
      POLYV<ZPZV<1>, ZPZV<0>, ZPZV<0>, ZPZV<0>, ZPZV<1>, ZPZV<424»; }; // NOLINT
03883 template<> struct ConwayPolynomial<439, 6> { using ZPZ = aerobus::zpz<439>; using type =
      POLYV<ZPZV<1>, ZPZV<0>, ZPZV<0>, ZPZV<1>, ZPZV<324>, ZPZV<190>, ZPZV<15»; }; // NOLINT
03884 template<> struct ConwayPolynomial<439, 7> { using ZPZ = aerobus::zpz<439>; using type
      POLYV<ZPZV<1>, ZPZV<0>, ZPZV<0>, ZPZV<0>, ZPZV<0>, ZPZV<0>, ZPZV<0>, ZPZV<1>, ZPZV<424»; };
03885 template<> struct ConwayPolynomial<439, 8> { using ZPZ = aerobus::zpz<439>, using type =
      POLYV<ZPZV<1>, ZPZV<0>, ZPZV<0>, ZPZV<0>, ZPZV<35>, ZPZV<296>, ZPZV<266>, ZPZV<15»; }; //
      NOLTNT
03886 template<> struct ConwayPolynomial<439, 9> { using ZPZ = aerobus::zpz<439>; using type = POLYV<ZPZV<1>, ZPZV<0>, ZPZV<0>, ZPZV<0>, ZPZV<0>, ZPZV<16>, ZPZV<342>, ZPZV<342>, ZPZV<254>, ZPZV<424»;
03887 template<> struct ConwayPolynomial<443, 1> { using ZPZ = aerobus::zpz<443>; using type =
      POLYV<ZPZV<1>, ZPZV<441»; }; // NOLINT
03888 template<> struct ConwayPolynomial<443, 2> { using ZPZ = aerobus::zpz<443>; using type =
POLYV<ZPZV<1>, ZPZV<437>, ZPZV<2»; }; // NOLINT
03889 template<> struct ConwayPolynomial<443, 3> { using ZPZ = aerobus::zpz<443>; using type =
      POLYV<ZPZV<1>, ZPZV<0>, ZPZV<4>, ZPZV<441»; }; // NOLINT
03890 template<> struct ConwayPolynomial<443, 4> { using ZPZ = aerobus::zpz<443>; using type =
POLYV<ZPZV<1>, ZPZV<0>, ZPZV<4>, ZPZV<383>, ZPZV<2»; }; // NOLINT
03891 template<> struct ConwayPolynomial<443, 5> { using ZPZ = aerobus::zpz<443>; using type =
      POLYV<ZPZV<1>, ZPZV<0>, ZPZV<0>, ZPZV<4+, ZPZV<441»; }; // NOLINT
03892 template<> struct ConwayPolynomial<443, 6> { using ZPZ = aerobus::zpz<443>; using type =
      POLYV<ZPZV<1>, ZPZV<0>, ZPZV<1>, ZPZV<298>, ZPZV<218>, ZPZV<41>, ZPZV<2»; }; // NOLINT
03893 template<> struct ConwayPolynomial<443, 7> { using ZPZ = aerobus::zpz<443>; using type =
      POLYV<ZPZV<1>, ZPZV<0>, ZPZV<0>, ZPZV<0>, ZPZV<0>, ZPZV<0>, ZPZV<6>, ZPZV<6>, ZPZV<441»; };
03894 template<> struct ConwayPolynomial<443, 8> { using ZPZ = aerobus::zpz<443>; using type =
      POLYV<ZPZV<1>, ZPZV<0>, ZPZV<0>, ZPZV<0>, ZPZV<2>, ZPZV<437>, ZPZV<217>, ZPZV<290>, ZPZV<2»; }; //
      NOLINT
03895 template<> struct ConwayPolynomial<443, 9> { using ZPZ = aerobus::zpz<443>; using type =
       POLYV<ZPZV<1>, ZPZV<0>, ZPZV<0>, ZPZV<0>, ZPZV<0>, ZPZV<0>, ZPZV<0>, ZPZV<3>, ZPZV<109>, ZPZV<441»;
       }; // NOLINT
03896 template<> struct ConwayPolynomial<449, 1> { using ZPZ = aerobus::zpz<449>; using type =
      POLYV<ZPZV<1>, ZPZV<446»; }; // NOLINT
03897 template<> struct ConwayPolynomial<449, 2> { using ZPZ = aerobus::zpz<449>; using type =
      POLYV<ZPZV<1>, ZPZV<4444>, ZPZV<3»; }; // NOLINT
03898 template<> struct ConwayPolynomial<449, 3> { using ZPZ = aerobus::zpz<449>; using type =
                                                           // NOLINT
      POLYV<ZPZV<1>, ZPZV<0>, ZPZV<1>, ZPZV<446»; };
03899 template<> struct ConwayPolynomial<449, 4> { using ZPZ = aerobus::zpz<449>; using type = POLYV<ZPZV<1>, ZPZV<0>, ZPZV<2>, ZPZV<249>, ZPZV<3»; }; // NOLINT
03900 template<> struct ConwayPolynomial<449, 5> { using ZPZ = aerobus::zpz<449>; using type =
      POLYV<ZPZV<1>, ZPZV<0>, ZPZV<0>, ZPZV<0>, ZPZV<9>, ZPZV<446»; }; // NOLINT
03901 template<> struct ConwayPolynomial<449, 6> { using ZPZ = aerobus::zpz<449>; using type =
      POLYV<ZPZV<1>, ZPZV<0>, ZPZV<2>, ZPZV<437>, ZPZV<293>, ZPZV<69>, ZPZV<3»; }; // NOLINT
03902 template<> struct ConwayPolynomial<449, 7> { using ZPZ = aerobus::zpz<449>; using type =
      POLYV<ZPZV<1>, ZPZV<0>, ZPZV<0>, ZPZV<0>, ZPZV<0>, ZPZV<0>, ZPZV<0>, ZPZV<28>, ZPZV<446»; };
03903 template<> struct ConwayPolynomial<449, 8> { using ZPZ = aerobus::zpz<449>; using type =
      POLYV<ZPZV<1>, ZPZV<0>, ZPZV<0>, ZPZV<0>, ZPZV<0>, ZPZV<34>, ZPZV<348>, ZPZV<124>, ZPZV<12*, ZPZV<3*; }; //
03904 template<> struct ConwayPolynomial<449, 9> { using ZPZ = aerobus::zpz<449>; using type =
      POLYV<ZPZV<1>, ZPZV<0>, ZPZV<0>, ZPZV<0>, ZPZV<0>, ZPZV<0>, ZPZV<6>, ZPZV<226>, ZPZV<9>, ZPZV<4446»; };
       // NOLINT
03905 template<> struct ConwayPolynomial<457, 1> { using ZPZ = aerobus::zpz<457>; using type =
      POLYV<ZPZV<1>, ZPZV<444»; }; // NOLINT
03906 template<> struct ConwayPolynomial<457, 2> { using ZPZ = aerobus::zpz<457>; using type =
POLYV<ZPZV<1>, ZPZV<454>, ZPZV<13s; }; // NOLINT
03907 template<> struct ConwayPolynomial<457, 3> { using ZPZ = aerobus::zpz<457>; using type =
      POLYV<ZPZV<1>, ZPZV<0>, ZPZV<1>, ZPZV<444»; }; // NOLINT
03908 template<> struct ConwayPolynomial<457, 4> { using ZPZ = aerobus::zpz<457>; using type = POLYV<ZPZV<1>, ZPZV<0>, ZPZV<8>, ZPZV<407>, ZPZV<13»; }; // NOLINT
03909 template<> struct ConwayPolynomial<457, 5> { using ZPZ = aerobus::zpz<457>; using type =
      POLYV<ZPZV<1>, ZPZV<0>, ZPZV<0>, ZPZV<0>, ZPZV<4>, ZPZV<44*, }; // NOLINT
03910 template<> struct ConwayPolynomial<457, 6> { using ZPZ = aerobus::zpz<457>; using type =
      POLYV<ZPZV<1>, ZPZV<0>, ZPZV<0>, ZPZV<205>, ZPZV<389>, ZPZV<266>, ZPZV<13»; }; // NOLINT
03911 template<> struct ConwayPolynomial<457, 7> { using ZPZ = aerobus::zpz<457>; using type =
      POLYV<ZPZV<1>, ZPZV<0>, ZPZV<0>, ZPZV<0>, ZPZV<0>, ZPZV<0>, ZPZV<14>, ZPZV<444»; }; // NOLINT
03912 template<> struct ConwayPolynomial<457, 8> { using ZPZ = aerobus::zpz<457>; using type
      POLYV<ZPZV<1>, ZPZV<0>, ZPZV<0>, ZPZV<0>, ZPZV<9>, ZPZV<365>, ZPZV<296>, ZPZV<412>, ZPZV<13»; }; //
      NOLINT
03913 template<> struct ConwayPolynomial<457, 9> { using ZPZ = aerobus::zpz<457>; using type =
      POLYV<ZPZV<1>, ZPZV<0>, ZPZV<0>, ZPZV<0>, ZPZV<0>, ZPZV<0>, ZPZV<9>, ZPZV<354>, ZPZV<84>, ZPZV<444*;
       ): // NOLINT
03914 template<> struct ConwayPolynomial<461, 1> { using ZPZ = aerobus::zpz<461>; using type =
      POLYV<ZPZV<1>, ZPZV<459»; }; // NOLINT
03915 template<> struct ConwayPolynomial<461, 2> { using ZPZ = aerobus::zpz<461>; using type =
      POLYV<ZPZV<1>, ZPZV<460>, ZPZV<2»; }; // NOLINT
03916 template<> struct ConwayPolynomial<461, 3> { using ZPZ = aerobus::zpz<461>; using type =
POLYV<ZPZV<1>, ZPZV<0>, ZPZV<3>, ZPZV<459»; }; // NOLINT
03917 template<> struct ConwayPolynomial<461, 4> { using ZPZ = aerobus::zpz<461>; using type =
```

```
POLYV<ZPZV<1>, ZPZV<0>, ZPZV<3>, ZPZV<393>, ZPZV<2»; }; // NOLINT
03918 template<> struct ConwayPolynomial<461, 5> { using ZPZ = aerobus::zpz<461>; using type =
      POLYV<ZPZV<1>, ZPZV<0>, ZPZV<0>, ZPZV<0>, ZPZV<459»; }; // NOLINT
03919 template<> struct ConwayPolynomial<461, 6> { using ZPZ = aerobus::zpz<461>; using type =
POLYV<ZPZV<1>, ZPZV<0>, ZPZV<1>, ZPZV<439>, ZPZV<432>, ZPZV<329>, ZPZV<2»; }; // NOLINT 03920 template<> struct ConwayPolynomial<461, 7> { using ZPZ = aerobus::zpz<461>; using type =
      POLYV<ZPZV<1>, ZPZV<0>, ZPZV<0>, ZPZV<0>, ZPZV<0>, ZPZV<0>, ZPZV<5>, ZPZV<459»; }; // NOLINT
03921 template<> struct ConwayPolynomial<461, 8> { using ZPZ = aerobus::zpz<461>; using type
       POLYV<ZPZV<1>, ZPZV<0>, ZPZV<0>, ZPZV<0>, ZPZV<15>, ZPZV<388>, ZPZV<449>, ZPZV<321>, ZPZV<32»; }; //
       NOLINT
03922 template<> struct ConwayPolynomial<461, 9> { using ZPZ = aerobus::zpz<461>; using type = POLYV<ZPZV<1>, ZPZV<0>, ZPZV<0>, ZPZV<0>, ZPZV<0>, ZPZV<0>, ZPZV<5>, ZPZV<5>, ZPZV<210>, ZPZV<276>, ZPZV<459»;
       }; // NOLINT
03923 template<> struct ConwayPolynomial<463, 1> { using ZPZ = aerobus::zpz<463>; using type =
       POLYV<ZPZV<1>, ZPZV<460»; }; // NOLINT
03924 template<> struct ConwayPolynomial<463, 2> { using ZPZ = aerobus::zpz<463>; using type =
POLYV<ZPZV<1>, ZPZV<461>, ZPZV<3»; }; // NOLINT
03925 template<> struct ConwayPolynomial<463, 3> { using ZPZ = aerobus::zpz<463>; using type =
       POLYV<ZPZV<1>, ZPZV<0>, ZPZV<10>, ZPZV<460»; }; // NOLINT
03926 template<> struct ConwayPolynomial<463, 4> { using ZPZ = aerobus::zpz<463>; using type =
       POLYV<ZPZV<1>, ZPZV<0>, ZPZV<17>, ZPZV<262>, ZPZV<3»; }; // NOLINT
03927 template<> struct ConwayPolynomial<463, 5> { using ZPZ = aerobus::zpz<463>; using type =
      03928 template<> struct ConwayPolynomial<463, 6> { using ZPZ = aerobus::zpz<463>; using type = POLYV<ZPZV<1>, ZPZV<0>, ZPZV<0>, ZPZV<462>, ZPZV<51>, ZPZV<110>, ZPZV<3»; }; // NOLINT
03929 template<> struct ConwayPolynomial<463, 7> { using ZPZ = aerobus::zpz<463>; using type
       POLYV<ZPZV<1>, ZPZV<0>, ZPZV<0>, ZPZV<0>, ZPZV<0>, ZPZV<0>, ZPZV<0>, ZPZV<13>, ZPZV<460»; };
03930 template<> struct ConwayPolynomial<463, 8> { using ZPZ = aerobus::zpz<463>; using type =
       POLYV<ZPZV<1>, ZPZV<0>, ZPZV<0>, ZPZV<0>, ZPZV<0>, ZPZV<234>, ZPZV<414>, ZPZV<396>, ZPZV<3»; }; //
       NOLINT
03931 template<> struct ConwayPolynomial<463, 9> { using ZPZ = aerobus::zpz<463>; using type = POLYV<ZPZV<1>, ZPZV<0>, ZPZV<0>, ZPZV<0>, ZPZV<0>, ZPZV<0>, ZPZV<3>, ZPZV<433>, ZPZV<433>, ZPZV<227>, ZPZV<460»;
03932 template<> struct ConwayPolynomial<467, 1> { using ZPZ = aerobus::zpz<467>; using type =
       POLYV<ZPZV<1>, ZPZV<465»; }; // NOLINT
03933 template<> struct ConwayPolynomial<467, 2> { using ZPZ = aerobus::zpz<467>; using type =
POLYV<ZPZV<1>, ZPZV<463>, ZPZV<2»; }; // NOLINT
03934 template<> struct ConwayPolynomial<467, 3> { using ZPZ = aerobus::zpz<467>; using type =
       POLYV<ZPZV<1>, ZPZV<0>, ZPZV<2>, ZPZV<465»; }; // NOLINT
03935 template<> struct ConwayPolynomial<467, 4> { using ZPZ = aerobus::zpz<467>; using type =
POLYV<ZPZV<1>, ZPZV<0>, ZPZV<14>, ZPZV<353>, ZPZV<2»; }; // NOLINT
03936 template<> struct ConwayPolynomial<467, 5> { using ZPZ = aerobus::zpz<467>; using type =
      POLYV<ZPZV<1>, ZPZV<0>, ZPZV<0>, ZPZV<2>, ZPZV<465»; }; // NOLINT
03937 template<> struct ConwayPolynomial<467, 6> { using ZPZ = aerobus::zpz<467>; using type =
      POLYV<ZPZV<1>, ZPZV<0>, ZPZV<1>, ZPZV<123>, ZPZV<62>, ZPZV<237>, ZPZV<2»; }; // NOLINT
03938 template<> struct ConwayPolynomial<467, 7> { using ZPZ = aerobus::zpz<467>;
      POLYV<ZPZV<1>, ZPZV<0>, ZPZV<0>, ZPZV<0>, ZPZV<0>, ZPZV<0>, ZPZV<7>, ZPZV<7>, ZPZV<465»; };
03939 template<> struct ConwayPolynomial<467, 8> { using ZPZ = aerobus::zpz<467>; using type =
       POLYV<ZPZV<1>, ZPZV<0>, ZPZV<0>, ZPZV<0>, ZPZV<5>, ZPZV<318>, ZPZV<413>, ZPZV<289>, ZPZV<2»; }; //
       NOLINT
03940 template<> struct ConwayPolynomial<467, 9> { using ZPZ = aerobus::zpz<467>; using type
       POLYV<ZPZV<1>, ZPZV<0>, ZPZV<0>, ZPZV<0>, ZPZV<0>, ZPZV<0>, ZPZV<0>, ZPZV<397>, ZPZV<3447>, ZPZV<465»;
       }; // NOLINT
03941 template<> struct ConwayPolynomial<479, 1> { using ZPZ = aerobus::zpz<479>; using type =
      POLYV<ZPZV<1>, ZPZV<466»; }; // NOLINT
03942 template<> struct ConwayPolynomial<479, 2> { using ZPZ = aerobus::zpz<479>; using type =
       POLYV<ZPZV<1>, ZPZV<474>, ZPZV<13»; }; // NOLINT
03943 template<> struct ConwayPolynomial<479, 3> { using ZPZ = aerobus::zpz<479>; using type =
       POLYV<ZPZV<1>, ZPZV<0>, ZPZV<4>, ZPZV<466»; }; // NOLINT
03944 template<> struct ConwayPolynomial<479, 4> { using ZPZ = aerobus::zpz<479>; using type =
POLYV<ZPZV<1>, ZPZV<0>, ZPZV<6>, ZPZV<386>, ZPZV<13»; }; // NOLINT
03945 template<> struct ConwayPolynomial<479, 5> { using ZPZ = aerobus::zpz<479>; using type =
      POLYV<ZPZV<1>, ZPZV<0>, ZPZV<0>, ZPZV<0>, ZPZV<466»; }; // NOLINT
03946 template<> struct ConwayPolynomial<479, 6> { using ZPZ = aerobus::zpz<479>; using type =
      POLYV<ZPZV<1>, ZPZV<0>, ZPZV<1>, ZPZV<243>, ZPZV<287>, ZPZV<334>, ZPZV<13»; }; // NOLINT
03947 template<> struct ConwayPolynomial<479, 7> { using ZPZ = aerobus::zpz<479>; using type =
      POLYV<ZPZV<1>, ZPZV<0>, ZPZV<0>, ZPZV<0>, ZPZV<0>, ZPZV<0>, ZPZV<4>, ZPZV<46»; };
03948 template<> struct ConwayPolynomial<479, 8> { using ZPZ = aerobus::zpz<479>; using type =
       POLYV<ZPZV<1>, ZPZV<0>, ZPZV<0>, ZPZV<0>, ZPZV<7>, ZPZV<247>, ZPZV<440>, ZPZV<17>, ZPZV<13»; }; //
       NOLINT
03949 template<> struct ConwayPolynomial<479, 9> { using ZPZ = aerobus::zpz<479>; using type =
       POLYV<ZPZV<1>, ZPZV<0>, ZPZV<0>, ZPZV<0>, ZPZV<0>, ZPZV<0>, ZPZV<2>, ZPZV<3>, ZPZV<185>, ZPZV<466»; };
       // NOLINT
03950 template<> struct ConwayPolynomial<487, 1> { using ZPZ = aerobus::zpz<487>; using type =
      POLYV<ZPZV<1>, ZPZV<484»; }; // NOLINT
03951 template<> struct ConwayPolynomial<487, 2> { using ZPZ = aerobus::zpz<487>; using type =
                                                  // NOLINT
       POLYV<ZPZV<1>, ZPZV<485>, ZPZV<3»; };
03952 template<> struct ConwayPolynomial<487, 3> { using ZPZ = aerobus::zpz<487>; using type =
      POLYV<ZPZV<1>, ZPZV<0>, ZPZV<4>, ZPZV<484»; }; // NOLINT
03953 template<> struct ConwayPolynomial<487, 4> { using ZPZ = aerobus::zpz<487>; using type =
POLYV<ZPZV<1>, ZPZV<0>, ZPZV<4>, ZPZV<48, ZPZV<483>, ZPZV<3»; }; // NOLINT
03954 template<> struct ConwayPolynomial<487, 5> { using ZPZ = aerobus::zpz<487>; using type =
      POLYV<ZPZV<1>, ZPZV<0>, ZPZV<0>, ZPZV<0>, ZPZV<1>, ZPZV<484»; }; // NOLINT
03955 template<> struct ConwayPolynomial<487, 6> { using ZPZ = aerobus::zpz<487>; using type =
POLYV<ZPZV<1>, ZPZV<0>, ZPZV<0>, ZPZV<450>, ZPZV<427>, ZPZV<185>, ZPZV<3»; }; // NOLINT 03956 template<> struct ConwayPolynomial<487, 7> { using ZPZ = aerobus::zpz<487>; using type = POLYV<ZPZV<1>, ZPZV<0>, ZPZV<0>, ZPZV<0>, ZPZV<0>, ZPZV<0>, ZPZV<7>, ZPZV<7>, ZPZV<7>, ZPZV<484»; }; // NO
```

```
03957 template<> struct ConwayPolynomial<487, 8> { using ZPZ = aerobus::zpz<487>; using type
       POLYV<ZPZV<1>, ZPZV<0>, ZPZV<0>, ZPZV<0>, ZPZV<0>, ZPZV<1>, ZPZV<283>, ZPZV<249>, ZPZV<137>, ZPZV<3»; }; //
       NOLTNT
03958 template<> struct ConwayPolynomial<487, 9> { using ZPZ = aerobus::zpz<487>; using type =
       POLYV<ZPZV<1>, ZPZV<0>, ZPZV<0>, ZPZV<0>, ZPZV<0>, ZPZV<0>, ZPZV<10>, ZPZV<10>, ZPZV<271>, ZPZV<4447>, ZPZV<484%;
       }; // NOLINT
03959 template<> struct ConwayPolynomial<491, 1> { using ZPZ = aerobus::zpz<491>; using type =
       POLYV<ZPZV<1>, ZPZV<489»; }; // NOLINT
03960 template<> struct ConwayPolynomial<491, 2> { using ZPZ = aerobus::zpz<491>; using type =
POLYV<ZPZV<1>, ZPZV<487, ZPZV<2»; }; // NOLINT
03961 template<> struct ConwayPolynomial<491, 3> { using ZPZ = aerobus::zpz<491>; using type =
       POLYV<ZPZV<1>, ZPZV<0>, ZPZV<2>, ZPZV<489»; }; // NOLINT
03962 template<> struct ConwayPolynomial<br/>491, 4> { using ZPZ = aerobus::zpz<491>; using type = POLYV<ZPZV<1>, ZPZV<0>, ZPZV<7>, ZPZV<360>, ZPZV<2»; }; // NOLINT<br/>03963 template<> struct ConwayPolynomial<br/>491, 5> { using ZPZ = aerobus::zpz<491>; using type =
POLYV<ZPZV<1>, ZPZV<0>, ZPZV<0>, ZPZV<0>, ZPZV<5>, ZPZV<489»; }; // NOLINT
03964 template<> struct ConwayPolynomial<491, 6> { using ZPZ = aerobus::zpz<491>; using type =
POLYV<ZPZV<1>, ZPZV<0>, ZPZV<1>, ZPZV<369>, ZPZV<402>, ZPZV<125>, ZPZV<29; }; // NOLINT 03965 template<> struct ConwayPolynomial<491, 7> { using ZPZ = aerobus::zpz<491>; using type =
       POLYV<ZPZV<1>, ZPZV<0>, ZPZV<0>, ZPZV<0>, ZPZV<0>, ZPZV<0>, ZPZV<5>, ZPZV<489»; };
03966 template<> struct ConwayPolynomial<491, 8> { using ZPZ = aerobus::zpz<491>; using type =
       POLYV<ZPZV<1>, ZPZV<0>, ZPZV<0>, ZPZV<0>, ZPZV<1>, ZPZV<378>, ZPZV<372>, ZPZV<216>, ZPZV<2»; }; //
       NOLINT
03967 template<> struct ConwayPolynomial<491, 9> { using ZPZ = aerobus::zpz<491>; using type = POLYV<ZPZV<1>, ZPZV<0>, ZPZV<0>, ZPZV<0>, ZPZV<0>, ZPZV<0>, ZPZV<453>, ZPZV<4453>, ZPZV<489»;
       }; // NOLINT
03968 template<> struct ConwayPolynomial<499, 1> { using ZPZ = aerobus::zpz<499>; using type =
       POLYV<ZPZV<1>, ZPZV<492»; }; // NOLINT
03969 template<> struct ConwayPolynomial<499, 2> { using ZPZ = aerobus::zpz<499>; using type =
POLYV<ZPZV<1>, ZPZV<493>, ZPZV<7»; }; // NOLINT

03970 template<> struct ConwayPolynomial<499, 3> { using ZPZ = aerobus::zpz<499>; using type =
       POLYV<ZPZV<1>, ZPZV<0>, ZPZV<2>, ZPZV<492»; }; // NOLINT
03971 template<> struct ConwayPolynomial<499, 4> { using ZPZ = aerobus::zpz<499>; using type =
POLYV<ZPZV<1>, ZPZV<0>, ZPZV<4>, ZPZV<495>, ZPZV<7»; }; // NOLINT
03972 template<> struct ConwayPolynomial<499, 5> { using ZPZ = aerobus::zpz<499>; using type =
POLYY<ZPZY<1>, ZPZY<0>, ZPZY<0>, ZPZY<0>, ZPZY<0>, ZPZY<0>, ZPZY<1>, ZPZY<492»; }; // NOLINT
03973 template<> struct ConwayPolynomial<499, 6> { using ZPZ = aerobus::zpz<499>; using type =
       POLYV<ZPZV<1>, ZPZV<0>, ZPZV<0>, ZPZV<407>, ZPZV<191>, ZPZV<78>, ZPZV<7»; }; // NOLINT
03974 template<> struct ConwayPolynomial<499, 7> { using ZPZ = aerobus::zpz<499>; using type
       POLYV<ZPZV<1>, ZPZV<0>, ZPZV<0>, ZPZV<0>, ZPZV<0>, ZPZV<0>, ZPZV<0>, ZPZV<0>, ZPZV<492»; };
03975 template<> struct ConwayPolynomial<499, 8> { using ZPZ = aerobus::zpz<499>; using type =
       POLYV<ZPZV<1>, ZPZV<0>, ZPZV<0>, ZPZV<0>, ZPZV<1>, ZPZV<288>, ZPZV<309>, ZPZV<200>, ZPZV<7»; }; //
       NOLINT
03976 template<> struct ConwayPolynomial<499, 9> { using ZPZ = aerobus::zpz<499>; using type =
       POLYV<ZPZV<1>, ZPZV<0>, ZPZV<0>, ZPZV<0>, ZPZV<0>, ZPZV<0>, ZPZV<5>, ZPZV<491>, ZPZV<222>, ZPZV<492»;
       }; // NOLINT
03977 template<> struct ConwayPolynomial<503, 1> { using ZPZ = aerobus::zpz<503>; using type =
       POLYV<ZPZV<1>, ZPZV<498»; }; // NOLINT
03978 template<> struct ConwayPolynomial<503, 2> { using ZPZ = aerobus::zpz<503>; using type =
POLYV<ZPZV<1>, ZPZV<498>, ZPZV<5»; }; // NOLINT
03979 template<>> struct ConwayPolynomial<503, 3> { using ZPZ = aerobus::zpz<503>; using type =
       POLYV<ZPZV<1>, ZPZV<0>, ZPZV<2>, ZPZV<498»; }; // NOLINT
03980 template<> struct ConwayPolynomial<503, 4> { using ZPZ = aerobus::zpz<503>; using type =
POLYV<ZPZV<1>, ZPZV<0>, ZPZV<6>, ZPZV<325>, ZPZV<5»; }; // NOLINT
03981 template<> struct ConwayPolynomial<503, 5> { using ZPZ = aerobus::zpz<503>; using type =
       POLYV<ZPZV<1>, ZPZV<0>, ZPZV<0>, ZPZV<0>, ZPZV<15>, ZPZV<498»; }; // NOLINT
03982 template<> struct ConwayPolynomial<503, 6> { using ZPZ = aerobus::zpz<503>; using type =
       POLYV<ZPZV<1>, ZPZV<0>, ZPZV<1>, ZPZV<380>, ZPZV<292>, ZPZV<255>, ZPZV<5»; }; // NOLINT
03983 template<> struct ConwayPolynomial<503, 7> { using ZPZ = aerobus::zpz<503>; using type =
       POLYV<ZPZV<1>, ZPZV<0>, ZPZV<0>, ZPZV<0>, ZPZV<0>, ZPZV<0>, ZPZV<0>, ZPZV<11>, ZPZV<498»; };
03984 template<> struct ConwayPolynomial<503, 8> { using ZPZ = aerobus::zpz<503>; using type =
       POLYV<ZPZV<1>, ZPZV<0>, ZPZV<0>, ZPZV<0>, ZPZV<8>, ZPZV<441>, ZPZV<203>, ZPZV<316>, ZPZV<5»; }; //
       NOLINT
03985 template<> struct ConwayPolynomial<503, 9> { using ZPZ = aerobus::zpz<503>; using type
       POLYV<ZPZV<1>, ZPZV<0>, ZPZV<0>, ZPZV<0>, ZPZV<0>, ZPZV<0>, ZPZV<3>, ZPZV<158>, ZPZV<337>, ZPZV<498»;
       }; // NOLINT
03986 template<> struct ConwayPolynomial<509, 1> { using ZPZ = aerobus::zpz<509>; using type =
       POLYV<ZPZV<1>, ZPZV<507»; }; // NOLINT
03987 template<> struct ConwayPolynomial<509, 2> { using ZPZ = aerobus::zpz<509>; using type =
       POLYV<ZPZV<1>, ZPZV<508>, ZPZV<2»; };
                                                   // NOLINT
03988 template<> struct ConwayPolynomial<509, 3> { using ZPZ = aerobus::zpz<509>; using type =
       POLYV<ZPZV<1>, ZPZV<0>, ZPZV<3>, ZPZV<507»; }; // NOLINT
03989 template<> struct ConwayPolynomial<509, 4> { using ZPZ = aerobus::zpz<509>; using type =
POLYV<ZPZV<1>, ZPZV<0>, ZPZV<4>, ZPZV<408>, ZPZV<2»; }; // NOLINT
03990 template<> struct ConwayPolynomial<509, 5> { using ZPZ = aerobus::zpz<509>; using type =
       POLYV<ZPZV<1>, ZPZV<0>, ZPZV<0>, ZPZV<3>, ZPZV<507»; }; // NOLINT
03991 template<> struct ConwayPolynomial<509, 6> { using ZPZ = aerobus::zpz<509>; using type
       POLYV<ZPZV<1>, ZPZV<0>, ZPZV<1>, ZPZV<350>, ZPZV<232>, ZPZV<41>, ZPZV<2»; }; // NOLINT
03992 template<> struct ConwayPolynomial<509, 7> { using ZPZ = aerobus::zpz<509>; using type =
       POLYV<ZPZV<1>, ZPZV<0>, ZPZV<0>, ZPZV<0>, ZPZV<0>, ZPZV<6>, ZPZV<507»; };
                                                                                                        // NOLINT
03993 template<> struct ConwayPolynomial<509, 8> { using ZPZ = aerobus::zpz<509>; using type
       POLYV<ZPZV<1>, ZPZV<0>, ZPZV<0>, ZPZV<0>, ZPZV<2>, ZPZV<420, ZPZV<473>, ZPZV<382>, ZPZV<2»; }; //
03994 template<> struct ConwayPolynomial<509, 9> { using ZPZ = aerobus::zpz<509>; using type =
       POLYV<ZPZV<1>, ZPZV<0>, ZPZV<0>, ZPZV<0>, ZPZV<0>, ZPZV<0>, ZPZV<0>, ZPZV<0>, ZPZV<314>, ZPZV<28>, ZPZV<507»;
       }; // NOLINT
03995 template<> struct ConwayPolynomial<521, 1> { using ZPZ = aerobus::zpz<521>; using type =
```

```
POLYV<ZPZV<1>, ZPZV<518»; }; // NOLINT
03996 template<> struct ConwayPolynomial<521, 2> { using ZPZ = aerobus::zpz<521>; using type =
       POLYV<ZPZV<1>, ZPZV<515>, ZPZV<3»; }; // NOLINT
03997 template<> struct ConwayPolynomial<521, 3> { using ZPZ = aerobus::zpz<521>; using type =
POLYV<ZPZV<1>, ZPZV<0>, ZPZV<4>, ZPZV<518»; }; // NOLINT
03998 template<> struct ConwayPolynomial<521, 4> { using ZPZ = aerobus::zpz<521>; using type =
POLYV<ZPZV<1>, ZPZV<0>, ZPZV<50>, ZPZV<50>, ZPZV<3»; }; // NOLINT
03999 template<> struct ConwayPolynomial<521, 5> { using ZPZ = aerobus::zpz<521>; using type =
       POLYV<ZPZV<1>, ZPZV<0>, ZPZV<0>, ZPZV<0>, ZPZV<3>, ZPZV<518»; }; // NOLINT
04000 template<> struct ConwayPolynomial<521, 6> { using ZPZ = aerobus::zpz<521>; using type = POLYV<ZPZV<1>, ZPZV<0>, ZPZV<1>, ZPZV<315>, ZPZV<153>, ZPZV<280>, ZPZV<3»; }; // NOLINT 04001 template<> struct ConwayPolynomial<521, 7> { using ZPZ = aerobus::zpz<521>; using type =
       POLYV<ZPZV<1>, ZPZV<0>, ZPZV<0>, ZPZV<0>, ZPZV<0>, ZPZV<0>, ZPZV<1>, ZPZV<518»; };
04002 template<> struct ConwayPolynomial<521, 8> { using ZPZ = aerobus::zpz<521>; using type =
       POLYV<ZPZV<1>, ZPZV<0>, ZPZV<0>, ZPZV<0>, ZPZV<6>, ZPZV<462>, ZPZV<407>, ZPZV<312>, ZPZV<3); }; //
04003 template<> struct ConwayPolynomial<521, 9> { using ZPZ = aerobus::zpz<521>; using type = POLYV<ZPZV<1>, ZPZV<0>, ZPZV<0>, ZPZV<0>, ZPZV<0>, ZPZV<0>, ZPZV<5>, ZPZV<5181>, ZPZV<483>, ZPZV<483>, ZPZV<518»;
       }; // NOLINT
04004 template<> struct ConwayPolynomial<523, 1> { using ZPZ = aerobus::zpz<523>; using type =
       POLYV<ZPZV<1>, ZPZV<521»; }; // NOLINT
04005 template<> struct ConwayPolynomial<523, 2> { using ZPZ = aerobus::zpz<523>; using type =
POLYV<ZPZV<1>, ZPZV<522>, ZPZV<2»; }; // NOLINT
04006 template<> struct ConwayPolynomial<523, 3> { using ZPZ = aerobus::zpz<523>; using type =
       POLYV<ZPZV<1>, ZPZV<0>, ZPZV<5>, ZPZV<521»; }; // NOLINT
04007 template<> struct ConwayPolynomial<523, 4> { using ZPZ = aerobus::zpz<523>; using type =
       POLYV<ZPZV<1>, ZPZV<0>, ZPZV<2>, ZPZV<382>, ZPZV<2»; }; // NOLINT
04008 template<> struct ConwayPolynomial<523, 5> { using ZPZ = aerobus::zpz<523>; using type =
       POLYV<ZPZV<1>, ZPZV<0>, ZPZV<0>, ZPZV<0>, ZPZV<4>, ZPZV<521»; }; // NOLINT
04009 template<> struct ConwayPolynomial<523, 6> { using ZPZ = aerobus::zpz<523>; using type =
       POLYV<2PZV<1>, 2PZV<0>, ZPZV<0>, ZPZV<475>, ZPZV<475>, ZPZV<371>, ZPZV<2»; }; // NOLINT
04010 template<> struct ConwayPolynomial<523, 7> { using ZPZ = aerobus::zpz<523>;
                                                                                                using type
       POLYV<ZPZV<1>, ZPZV<0>, ZPZV<0>, ZPZV<0>, ZPZV<0>, ZPZV<0>, ZPZV<13>, ZPZV<521»; };
04011 template<> struct ConwayPolynomial<523, 8> { using ZPZ = aerobus::zpz<523>; using type =
       POLYV<ZPZV<1>, ZPZV<0>, ZPZV<0>, ZPZV<0>, ZPZV<4>, ZPZV<518>, ZPZV<184>, ZPZV<380>, ZPZV<2»; }; //
       NOLINT
04012 template<> struct ConwayPolynomial<523, 9> { using ZPZ = aerobus::zpz<523>; using type = POLYV<ZPZV<1>, ZPZV<0>, ZPZV<0>, ZPZV<0>, ZPZV<0>, ZPZV<1>, ZPZV<342>, ZPZV<345>, ZPZV<145>, ZPZV<521»;
       }; // NOLINT
04013 template<> struct ConwayPolynomial<541, 1> { using ZPZ = aerobus::zpz<541>; using type =
       POLYV<ZPZV<1>, ZPZV<539»; }; // NOLINT
04014 template<> struct ConwayPolynomial<541, 2> { using ZPZ = aerobus::zpz<541>; using type =
POLYV<ZPZV<1>, ZPZV<537, ZPZV<2»; }; // NOLINT
04015 template<> struct ConwayPolynomial<541, 3> { using ZPZ = aerobus::zpz<541>; using type =
       POLYV<ZPZV<1>, ZPZV<0>, ZPZV<2>, ZPZV<539»; }; // NOLINT
04016 template<> struct ConwayPolynomial<541, 4> { using ZPZ = aerobus::zpz<541>; using type =
POLYV<ZPZV<1>, ZPZV<0>, ZPZV<6>, ZPZV<333>, ZPZV<2»; }; // NOLINT
04017 template<> struct ConwayPolynomial<541, 5> { using ZPZ = aerobus::zpz<541>; using type =
       POLYV<ZPZV<1>, ZPZV<0>, ZPZV<0>, ZPZV<3>, ZPZV<539»; }; // NOLINT
04018 template<> struct ConwayPolynomial<541, 6> { using ZPZ = aerobus::zpz<541; using type =
       POLYV<ZPZV<1>, ZPZV<0>, ZPZV<0>, ZPZV<239>, ZPZV<320>, ZPZV<69>, ZPZV<2»; }; // NOLINT
04019 template<> struct ConwayPolynomial<541, 7> { using ZPZ = aerobus::zpz<541>; using type
       POLYV<ZPZV<1>, ZPZV<0>, ZPZV<0>, ZPZV<0>, ZPZV<0>, ZPZV<0>, ZPZV<2>, ZPZV<2>, ZPZV<539»; };
04020 template<> struct ConwayPolynomial<541, 8> { using ZPZ = aerobus::zpz<541>; using type =
       POLYV<ZPZV<1>, ZPZV<0>, ZPZV<0>, ZPZV<0>, ZPZV<3>, ZPZV<316>, ZPZV<108>, ZPZV<113>, ZPZV<2»; }; //
       NOLINT
04021 template<> struct ConwayPolynomial<541, 9> { using ZPZ = aerobus::zpz<541>; using type =
       POLYV<ZPZV<1>, ZPZV<0>, ZPZV<0>, ZPZV<0>, ZPZV<0>, ZPZV<0>, ZPZV<16>, ZPZV<340>, ZPZV<318>, ZPZV<539»;
       }; // NOLINT
04022 template<> struct ConwayPolynomial<547, 1> { using ZPZ = aerobus::zpz<547>; using type =
       POLYV<ZPZV<1>, ZPZV<545»; }; // NOLINT
04023 template<> struct ConwayPolynomial<547, 2> { using ZPZ = aerobus::zpz<547>; using type =
POLYV<ZPZV<1>, ZPZV<543>, ZPZV<2»; }; // NOLINT
04024 template<> struct ConwayPolynomial<547, 3> { using ZPZ = aerobus::zpz<547>; using type =
       POLYV<ZPZV<1>, ZPZV<0>, ZPZV<4>, ZPZV<545»; }; // NOLINT
04025 template<> struct ConwayPolynomial<547, 4> { using ZPZ = aerobus::zpz<547>; using type =
POLYV<ZPZV<1>, ZPZV<0>, ZPZV<8>, ZPZV<334>, ZPZV<2»; }; // NOLINT

04026 template<> struct ConwayPolynomial<547, 5> { using ZPZ = aerobus::zpz<547>; using type = POLYV<ZPZV<1>, ZPZV<0>, ZPZV<0>, ZPZV<0>, ZPZV<2>, ZPZV<545»; }; // NOLINT
04027 template<> struct ConwayPolynomial<547, 6> { using ZPZ = aerobus::zpz<547>; using type
       POLYV<ZPZV<1>, ZPZV<0>, ZPZV<0>, ZPZV<334>, ZPZV<153>, ZPZV<423>, ZPZV<2»; }; // NOLINT
04028 template<> struct ConwayPolynomial<547, 7> { using ZPZ = aerobus::zpz<547>; using type =
       POLYV<ZPZV<1>, ZPZV<0>, ZPZV<0>, ZPZV<0>, ZPZV<0>, ZPZV<0>, ZPZV<1>, ZPZV<545»; }; // NOLINT
04029 template<> struct ConwayPolynomial<547, 8> { using ZPZ = aerobus::zpz<547>; using type =
       POLYV<ZPZV<1>, ZPZV<0>, ZPZV<0>, ZPZV<0>, ZPZV<10>, ZPZV<368>, ZPZV<20>, ZPZV<180>, ZPZV<2»; }; //
04030 template<> struct ConwayPolynomial<547, 9> { using ZPZ = aerobus::zpz<547>; using type
       POLYV<ZPZV<1>, ZPZV<0>, ZPZV<0>, ZPZV<0>, ZPZV<0>, ZPZV<0>, ZPZV<0>, ZPZV<11>, ZPZV<238>, ZPZV<263>, ZPZV<545»;
       }; // NOLTNT
04031 template<> struct ConwayPolynomial<557, 1> { using ZPZ = aerobus::zpz<557>; using type =
       POLYV<ZPZV<1>, ZPZV<555»; }; // NOLINT
04032 template<> struct ConwayPolynomial<557, 2> { using ZPZ = aerobus::zpz<557>; using type =
POLYV<ZPZV<1>, ZPZV<553>, ZPZV<2»; }; // NOLINT
04033 template<> struct ConwayPolynomial<557, 3> { using ZPZ = aerobus::zpz<557>; using type =
       POLYV<ZPZV<1>, ZPZV<0>, ZPZV<3>, ZPZV<555»; }; // NOLINT
04034 template<> struct ConwayPolynomial<557, 4> { using ZPZ = aerobus::zpz<557>; using type = POLYV<ZPZV<1>, ZPZV<0>, ZPZV<7>, ZPZV<430>, ZPZV<2»; }; // NOLINT
```

```
04035 template<> struct ConwayPolynomial<557, 5> { using ZPZ = aerobus::zpz<557>; using type =
       POLYV<ZPZV<1>, ZPZV<0>, ZPZV<0>, ZPZV<0>, ZPZV<555»; }; // NOLINT
04036 template<> struct ConwayPolynomial<557, 6> { using ZPZ = aerobus::zpz<557>; using type =
      POLYV<ZPZV<1>, ZPZV<0>, ZPZV<1>, ZPZV<202>, ZPZV<192>, ZPZV<253>, ZPZV<2»; }; // NOLINT
04037 template<> struct ConwayPolynomial<557, 7> { using ZPZ = aerobus::zpz<557>; using type =
       POLYV<ZPZV<1>, ZPZV<0>, ZPZV<0>, ZPZV<0>, ZPZV<0>, ZPZV<0>, ZPZV<6>, ZPZV<555»; };
04038 template<> struct ConwayPolynomial<557, 8> { using ZPZ = aerobus::zpz<557>; using type
       POLYV<ZPZV<1>, ZPZV<0>, ZPZV<0>, ZPZV<0>, ZPZV<0>, ZPZV<480>, ZPZV<384>, ZPZV<113>, ZPZV<2»; }; //
       NOLTNT
04039 template<> struct ConwayPolynomial<557, 9> { using ZPZ = aerobus::zpz<557>; using type = POLYV<ZPZV<1>, ZPZV<0>, ZPZV<0>, ZPZV<0>, ZPZV<0>, ZPZV<0>, ZPZV<5, ZPZV<456>, ZPZV<434>, ZPZV<434>, ZPZV<555»;
       }; // NOLINT
04040 template<> struct ConwayPolynomial<563, 1> { using ZPZ = aerobus::zpz<563>; using type =
       POLYV<ZPZV<1>, ZPZV<561»; }; // NOLINT
04041 template<> struct ConwayPolynomial<563, 2> { using ZPZ = aerobus::zpz<563>; using type =
POLYV<ZPZV<1>, ZPZV<5599, ZPZV<2»; }; // NOLINT
04042 template<> struct ConwayPolynomial<563, 3> { using ZPZ = aerobus::zpz<563>; using type =
      POLYV<ZPZV<1>, ZPZV<0>, ZPZV<3>, ZPZV<561»; }; // NOLINT
04043 template<> struct ConwayPolynomial<563, 4> { using ZPZ = aerobus::zpz<563>; using type =
       POLYV<ZPZV<1>, ZPZV<0>, ZPZV<20>, ZPZV<399>, ZPZV<2»; }; // NOLINT
04044 template<> struct ConwayPolynomial<563, 5> { using ZPZ = aerobus::zpz<563>; using type =
      POLYV<ZPZV<1>, ZPZV<0>, ZPZV<0>, ZPZV<6>, ZPZV<561»; }; // NOLINT
04045 template<> struct ConwayPolynomial<563, 6> { using ZPZ = aerobus::zpz<563>; using type =
POLYV<ZPZV<1>, ZPZV<0>, ZPZV<1>, ZPZV<12>, ZPZV<303>, ZPZV<246>, ZPZV<2*; }; // NOLINT 04046 template<> struct ConwayPolynomial<563, 7> { using ZPZ = aerobus::zpz<563>; using type =
      POLYV<ZPZV<1>, ZPZV<0>, ZPZV<0>, ZPZV<0>, ZPZV<0>, ZPZV<0>, ZPZV<5>, ZPZV<561»; }; // NOLINT
04047 template<> struct ConwayPolynomial<563, 8> { using ZPZ = aerobus::zpz<563>; using type
       POLYV<ZPZV<1>, ZPZV<0>, ZPZV<0>, ZPZV<0>, ZPZV<2>, ZPZV<503>, ZPZV<176>, ZPZV<509>, ZPZV<2»; }; //
       NOLINT
04048 template<> struct ConwayPolynomial<563, 9> { using ZPZ = aerobus::zpz<563>; using type = POLYV<ZPZV<1>, ZPZV<0>, ZPZV<0>, ZPZV<0>, ZPZV<0>, ZPZV<3>, ZPZV<15>, ZPZV<19>, ZPZV<561»; };
       // NOLINT
04049 template<> struct ConwayPolynomial<569, 1> { using ZPZ = aerobus::zpz<569>; using type =
       POLYV<ZPZV<1>, ZPZV<566»; }; // NOLINT
04050 template<> struct ConwayPolynomial<569, 2> { using ZPZ = aerobus::zpz<569>; using type =
POLYY<ZPZV<1>, ZPZV<568>, ZPZV<3»; }; // NOLINT
04051 template<> struct ConwayPolynomial<569, 3> { using ZPZ = aerobus::zpz<569>; using type =
       POLYV<ZPZV<1>, ZPZV<0>, ZPZV<4>, ZPZV<566»; }; // NOLINT
04052 template<> struct ConwayPolynomial<569, 4> { using ZPZ = aerobus::zpz<569>; using type =
POLYV<ZPZV<1>, ZPZV<0>, ZPZV<7>, ZPZV<381>, ZPZV<3»; }; // NOLINT
04053 template<> struct ConwayPolynomial<569, 5> { using ZPZ = aerobus::zpz<569>; using type =
      POLYV<ZPZV<1>, ZPZV<0>, ZPZV<0>, ZPZV<0>, ZPZV<4>, ZPZV<566»; }; // NOLINT
04054 template<> struct ConwayPolynomial<569, 6> { using ZPZ = aerobus::zpz<569>; using type =
       POLYV<ZPZV<1>, ZPZV<0>, ZPZV<2>, ZPZV<50>, ZPZV<263>, ZPZV<480>, ZPZV<3»; ); // NOLINT
04055 template<> struct ConwayPolynomial<569, 7> { using ZPZ = aerobus::zpz<569>, using type
       POLYV<ZPZV<1>, ZPZV<0>, ZPZV<0>, ZPZV<0>, ZPZV<0>, ZPZV<5>, ZPZV<56%; };
04056 template<> struct ConwayPolynomial<569, 8> { using ZPZ = aerobus::zpz<569>; using type =
       POLYV<ZPZV<1>, ZPZV<0>, ZPZV<0>, ZPZV<0>, ZPZV<1>, ZPZV<52>, ZPZV<173>, ZPZV<241>, ZPZV<3»; }; //
       NOLINT
04057 template<> struct ConwayPolynomial<569, 9> { using ZPZ = aerobus::zpz<569>; using type = POLYV<ZPZV<1>, ZPZV<0>, ZPZV<0>, ZPZV<0>, ZPZV<0>, ZPZV<0>, ZPZV<566>, ZPZV<566>; ZPZV<566>;
       }; // NOLINT
04058 template<> struct ConwayPolynomial<571, 1> { using ZPZ = aerobus::zpz<571>; using type =
      POLYV<ZPZV<1>, ZPZV<568»; }; // NOLINT
04059 template<> struct ConwayPolynomial<571, 2> { using ZPZ = aerobus::zpz<571>; using type =
POLYV<ZPZV<1>, ZPZV<570>, ZPZV<3»; }; // NOLINT
04060 template<> struct ConwayPolynomial<571, 3> { using ZPZ = aerobus::zpz<571>; using type =
       POLYV<ZPZV<1>, ZPZV<0>, ZPZV<8>, ZPZV<568»; }; // NOLINT
04061 template<> struct ConwayPolynomial<571, 4> { using ZPZ = aerobus::zpz<571>; using type =
POLYV<ZPZV<1>, ZPZV<0>, ZPZV<2>, ZPZV<402>, ZPZV<3»; }; // NOLINT
04062 template<> struct ConwayPolynomial<571, 5> { using ZPZ = aerobus::zpz<571>; using type =
      POLYV<ZPZV<1>, ZPZV<0>, ZPZV<0>, ZPZV<4>, ZPZV<568»; }; // NOLINT
04063 template<> struct ConwayPolynomial<571, 6> { using ZPZ = aerobus::zpz<571>; using type =
       POLYV<ZPZV<1>, ZPZV<0>, ZPZV<0>, ZPZV<221>, ZPZV<295>, ZPZV<33>, ZPZV<3»; }; // NOLINT
04064 template<> struct ConwayPolynomial<571, 7> { using ZPZ = aerobus::zpz<571>; using type =
      POLYV<ZPZV<1>, ZPZV<0>, ZPZV<0>, ZPZV<0>, ZPZV<0>, ZPZV<0>, ZPZV<0>, ZPZV<568»; };
04065 template<> struct ConwayPolynomial<571, 8> { using ZPZ = aerobus::zpz<571>; using type
       POLYV<ZPZV<1>, ZPZV<0>, ZPZV<0>, ZPZV<0>, ZPZV<4>, ZPZV<363>, ZPZV<119>, ZPZV<371>, ZPZV<37), }; //
      NOLINT
04066 template<> struct ConwayPolynomial<571, 9> { using ZPZ = aerobus::zpz<571>; using type
       POLYV<ZPZV<1>, ZPZV<0>, ZPZV<0>, ZPZV<0>, ZPZV<0>, ZPZV<0>, ZPZV<0>, ZPZV<34>, ZPZV<545>, ZPZV<179>, ZPZV<568»;
       }; // NOLINT
04067 template<> struct ConwayPolynomial<577, 1> { using ZPZ = aerobus::zpz<577>; using type =
      POLYV<ZPZV<1>, ZPZV<572»; }; // NOLINT
04068 template<> struct ConwayPolynomial<577, 2> { using ZPZ = aerobus::zpz<577>; using type =
      POLYV<ZPZV<1>, ZPZV<572>, ZPZV<5»; }; // NOLINT
04069 template<> struct ConwayPolynomial<577, 3> { using ZPZ = aerobus::zpz<577>; using type =
       POLYV<ZPZV<1>, ZPZV<0>, ZPZV<2>, ZPZV<572»; }; // NOLINT
04070 template<> struct ConwayPolynomial<577, 4> { using ZPZ = aerobus::zpz<577>; using type =
POLYV<ZPZV<1>, ZPZV<0>, ZPZV<12>, ZPZV<494>, ZPZV<5»; }; // NOLINT

04071 template<> struct ConwayPolynomial<577, 5> { using ZPZ = aerobus::zpz<577>; using type = POLYV<ZPZV<1>, ZPZV<0>, ZPZV<0>, ZPZV<0>, ZPZV<3>, ZPZV<572»; }; // NOLINT
04072 template<> struct ConwayPolynomial<577, 6> { using ZPZ = aerobus::zpz<577>; using type =
      POLYV<ZPZV<1>, ZPZV<0>, ZPZV<1>, ZPZV<450>, ZPZV<25>, ZPZV<283>, ZPZV<5»; }; // NOLINT
04073 template<> struct ConwayPolynomial<577, 7> { using ZPZ = aerobus::zpz<577>; using type =
POLYV<ZPZV<1>, ZPZV<0>, ZPZV<0>, ZPZV<0>, ZPZV<0>, ZPZV<0>, ZPZV<0>, ZPZV<0>, ZPZV<8>, ZPZV<572»; }; // NOLINT 04074 template<> struct ConwayPolynomial<577, 8> { using ZPZ = aerobus::zpz<577>; using type =
```

```
POLYV<ZPZV<1>, ZPZV<0>, ZPZV<0>, ZPZV<0>, ZPZV<12>, ZPZV<450>, ZPZV<545>, ZPZV<321>, ZPZV<5*; }; //
04075 template<> struct ConwayPolynomial<577, 9> { using ZPZ = aerobus::zpz<577>; using type =
        POLYV<ZPZV<1>, ZPZV<0>, ZPZV<0>, ZPZV<0>, ZPZV<0>, ZPZV<0>, ZPZV<17>, ZPZV<576>, ZPŽV<449>, ZPZV<572»;
        }: // NOLINT
04076 template<> struct ConwayPolynomial<587, 1> { using ZPZ = aerobus::zpz<587>; using type =
        POLYV<ZPZV<1>, ZPZV<585»; }; // NOLINT
04077 template<> struct ConwayPolynomial<587, 2> { using ZPZ = aerobus::zpz<587>; using type =
        POLYV<ZPZV<1>, ZPZV<583>, ZPZV<2»; }; // NOLINT
04078 template<> struct ConwayPolynomial<587, 3> { using ZPZ = aerobus::zpz<587>; using type =
        POLYV<ZPZV<1>, ZPZV<0>, ZPZV<3>, ZPZV<585»; }; // NOLINT
04079 template<> struct ConwayPolynomial<587, 4> { using ZPZ = aerobus::zpz<587>; using type =
POLYV<ZPZV<1>, ZPZV<0>, ZPZV<16>, ZPZV<444>, ZPZV<2»; }; // NOLINT
04080 template<> struct ConwayPolynomial<587, 5> { using ZPZ = aerobus::zpz<587>; using type =
        POLYV<ZPZV<1>, ZPZV<0>, ZPZV<0>, ZPZV<0>, ZPZV<2>, ZPZV<585»; }; // NOLINT
04081 template<> struct ConwayPolynomial<587, 6> { using ZPZ = aerobus::zpz<587>; using type =
        POLYV<ZPZV<1>, ZPZV<0>, ZPZV<1>, ZPZV<204>, ZPZV<121>, ZPZV<226>, ZPZV<2»; }; // NOLINT
04082 template<> struct ConwayPolynomial<587, 7> { using ZPZ = aerobus::zpz<587>; using type =
        POLYV<ZPZV<1>, ZPZV<0>, ZPZV<0>, ZPZV<0>, ZPZV<0>, ZPZV<0>, ZPZV<3>, ZPZV<585»; }; // NOLINT
04083 template<> struct ConwayPolynomial<587, 8> { using ZPZ = aerobus::zpz<587>; using type
        POLYV<ZPZV<1>, ZPZV<0>, ZPZV<0>, ZPZV<0>, ZPZV<5>, ZPZV<492>, ZPZV<444>, ZPZV<91>, ZPZV<91; }; //
        NOLINT
04084 template<> struct ConwayPolynomial<587, 9> { using ZPZ = aerobus::zpz<587>; using type =
        POLYV<ZPZV<1>, ZPZV<0>, ZPZV<0>, ZPZV<0>, ZPZV<0>, ZPZV<0>, ZPZV<585»;
         }; // NOLINT
04085 template<> struct ConwayPolynomial<593, 1> { using ZPZ = aerobus::zpz<593>; using type =
        POLYV<ZPZV<1>, ZPZV<590»; }; // NOLINT
04086 template<> struct ConwayPolynomial<593, 2> { using ZPZ = aerobus::zpz<593>; using type =
        POLYV<ZPZV<1>, ZPZV<592>, ZPZV<3»; }; // NOLINT
04087 template<> struct ConwayPolynomial<593, 3> { using ZPZ = aerobus::zpz<593>; using type =
        POLYV<ZPZV<1>, ZPZV<0>, ZPZV<6>, ZPZV<590»; }; // NOLINT
04088 template<> struct ConwayPolynomial<593, 4> { using ZPZ = aerobus::zpz<593>; using type =
POLYV<ZPZV<1>, ZPZV<4>, ZPZV<419, ZPZV<419, ZPZV<3»; }; // NOLINT
04089 template<> struct ConwayPolynomial<593, 5> { using ZPZ = aerobus::zpz<593>; using type =
        04090 template<> struct ConwayPolynomial<593, 6> { using ZPZ = aerobus::zpz<593>; using type =
POLYV<ZPZV<1>, ZPZV<0>, ZPZV<2>, ZPZV<345>, ZPZV<65>, ZPZV<478>, ZPZV<3y; }; // NOLINT 04091 template<> struct ConwayPolynomial<593, 7> { using ZPZ = aerobus::zpz<593>; using type =
        POLYV<ZPZV<1>, ZPZV<0>, ZPZV<0>, ZPZV<0>, ZPZV<0>, ZPZV<0>, ZPZV<15>, ZPZV<590»; };
04092 template<> struct ConwayPolynomial<593, 8> { using ZPZ = aerobus::zpz<593>; using type =
        POLYV<ZPZV<1>, ZPZV<0>, ZPZV<0>, ZPZV<0>, ZPZV<9>, ZPZV<350>, ZPZV<291>, ZPZV<495>, ZPZV<495>, ZPZV<3»; }; //
        NOLINT
04093 template<> struct ConwayPolynomial<593, 9> { using ZPZ = aerobus::zpz<593>; using type = POLYV<ZPZV<1>, ZPZV<0>, ZPZV<0>, ZPZV<0>, ZPZV<0>, ZPZV<0>, ZPZV<590, Z
04094 template<> struct ConwayPolynomial<599, 1> { using ZPZ = aerobus::zpz<599>; using type =
        POLYV<ZPZV<1>, ZPZV<592»; }; // NOLINT
04095 template<> struct ConwayPolynomial<599, 2> { using ZPZ = aerobus::zpz<599>; using type =
        POLYV<ZPZV<1>, ZPZV<598>, ZPZV<7»; };
                                                             // NOLINT
04096 template<> struct ConwayPolynomial<599, 3> { using ZPZ = aerobus::zpz<599>; using type =
        POLYV<ZPZV<1>, ZPZV<0>, ZPZV<2>, ZPZV<592»; }; // NOLINT
04097 template<> struct ConwayPolynomial<599, 4> { using ZPZ = aerobus::zpz<599>; using type =
POLYV<ZPZV<1>, ZPZV<0>, ZPZV<3>, ZPZV<419>, ZPZV<7»; }; // NOLINT

04098 template<> struct ConwayPolynomial<599, 5> { using ZPZ = aerobus::zpz<599>; using type = POLYV<ZPZV<1>, ZPZV<0>, ZPZV<0>, ZPZV<0>, ZPZV<8>, ZPZV<592»; }; // NOLINT
04099 template<> struct ConwayPolynomial<599, 6> { using ZPZ = aerobus::zpz<599>; using type =
        POLYV<ZPZV<1>, ZPZV<0>, ZPZV<1>, ZPZV<515>, ZPZV<274>, ZPZV<586>, ZPZV<7»; }; // NOLINT
04100 template<> struct ConwayPolynomial<599, 7> { using ZPZ = aerobus::zpz<599>; using type
        POLYV<ZPZV<1>, ZPZV<0>, ZPZV<0>, ZPZV<0>, ZPZV<0>, ZPZV<0>, ZPZV<2>, ZPZV<592»; }; // NOLINT
04101 template<> struct ConwayPolynomial<599, 8> { using ZPZ = aerobus::zpz<599>; using type =
        POLYV<ZPZV<1>, ZPZV<0>, ZPZV<0>, ZPZV<0>, ZPZV<3>, ZPZV<440>, ZPZV<37>, ZPZV<124>, ZPZV<7»; }; //
        NOLINT
04102 template<> struct ConwayPolynomial<599, 9> { using ZPZ = aerobus::zpz<599>; using type =
        POLYV<ZPZV<1>, ZPZV<0>, ZPZV<0>, ZPZV<0>, ZPZV<0>, ZPZV<0>, ZPZV<0>, ZPZV<3>, ZPZV<114>, ZPZV<98>, ZPZV<592»;
         }; // NOLINT
04103 template<> struct ConwayPolynomial<601, 1> { using ZPZ = aerobus::zpz<601>; using type =
        POLYV<ZPZV<1>, ZPZV<594»; }; // NOLINT
04104 template<> struct ConwayPolynomial<601, 2> { using ZPZ = aerobus::zpz<601>; using type =
        POLYV<ZPZV<1>, ZPZV<598>, ZPZV<7»; }; // NOLINT
04105 template<> struct ConwayPolynomial<601, 3> { using ZPZ = aerobus::zpz<601>; using type =
        POLYV<ZPZV<1>, ZPZV<0>, ZPZV<1>, ZPZV<594»; }; // NOLINT
04106 template<> struct ConwayPolynomial<601, 4> { using ZPZ = aerobus::zpz<601>; using type =
POLYV<ZPZV<1>, ZPZV<0>, ZPZV<14>, ZPZV<347>, ZPZV<7»; }; // NOLINT
04107 template<> struct ConwayPolynomial<601, 5> { using ZPZ = aerobus::zpz<601>; using type =
        POLYV<ZPZV<1>, ZPZV<0>, ZPZV<0>, ZPZV<0>, ZPZV<13>, ZPZV<594»; }; // NOLINT
04108 template<> struct ConwayPolynomial<601, 6> { using ZPZ = aerobus::zpz<601; using type =
        POLYV<ZPZV<1>, ZPZV<0>, ZPZV<1>, ZPZV<128>, ZPZV<440>, ZPZV<49>, ZPZV<7»; }; // NOLINT
04109 template<> struct ConwayPolynomial<601, 7> { using ZPZ = aerobus::zpz<601>, using type =
        POLYV<ZPZV<1>, ZPZV<0>, ZPZV<0>, ZPZV<0>, ZPZV<0>, ZPZV<0>, ZPZV<5>, ZPZV<5>, ZPZV<594»; };
                                                                                                                           // NOLINT
04110 template<> struct ConwayPolynomial<601, 8> { using ZPZ = aerobus::zpz<601>; using type =
        POLYV<ZPZV<1>, ZPZV<0>, ZPZV<0>, ZPZV<0>, ZPZV<0>, ZPZV<550>, ZPZV<241>, ZPZV<490>, ZPZV<7»; }; //
04111 template<> struct ConwayPolynomial<601, 9> { using ZPZ = aerobus::zpz<601>; using type :
        POLYV<ZPZV<1>, ZPZV<0>, ZPZV<0>, ZPZV<0>, ZPZV<0>, ZPZV<0>, ZPZV<7>, ZPZV<487>, ZPZV<590>, ZPZV<594»;
        }; // NOLINT
04112 template<> struct ConwayPolynomial<607, 1> { using ZPZ = aerobus::zpz<607>; using type =
        POLYV<ZPZV<1>, ZPZV<604»; }; // NOLINT
```

```
04113 template<> struct ConwayPolynomial<607, 2> { using ZPZ = aerobus::zpz<607>; using type =
POLYY<ZPZV<1>, ZPZV<606>, ZPZV<3»; }; // NOLINT
04114 template<>> struct ConwayPolynomial<607, 3> { using ZPZ = aerobus::zpz<607>; using type =
         POLYV<ZPZV<1>, ZPZV<0>, ZPZV<5>, ZPZV<604»; }; // NOLINT
04115 template<> struct ConwayPolynomial<607, 4> { using ZPZ = aerobus::zpz<607>; using type = POLYV<ZPZV<1>, ZPZV<0>, ZPZV<8>, ZPZV<449>, ZPZV<3»; }; // NOLINT
04116 template<> struct ConwayPolynomial<607, 5> { using ZPZ = aerobus::zpz<607>; using type =
         POLYV<ZPZV<1>, ZPZV<0>, ZPZV<0>, ZPZV<0>, ZPZV<4>, ZPZV<604»; }; // NOLINT
04117 template<> struct ConwayPolynomial<607, 6> { using ZPZ = aerobus::zpz<607>; using type =
         POLYV<ZPZV<1>, ZPZV<0>, ZPZV<0>, ZPZV<10>, ZPZV<45>, ZPZV<478>, ZPZV<3»; }; // NOLINT
04118 template<> struct ConwayPolynomial<607, 7> { using ZPZ = aerobus::zpz<607>; using type :
         POLYV<ZPZV<1>, ZPZV<0>, ZPZV<0>, ZPZV<0>, ZPZV<0>, ZPZV<0>, ZPZV<0>, ZPZV<9>, ZPZV<604»; }; // NOLINT
04119 template<> struct ConwayPolynomial<607, 8> { using ZPZ = aerobus::zpz<607>; using type
         POLYV<ZPZV<1>, ZPZV<0>, ZPZV<0>, ZPZV<0>, ZPZV<4>, ZPZV<468>, ZPZV<35>, ZPZV<449>, ZPZV<3»; }; //
         NOLINT
04120 template<> struct ConwayPolynomial<607, 9> { using ZPZ = aerobus::zpz<607>; using type =
         POLYV<ZPZV<1>, ZPZV<0>, ZPZV<0>, ZPZV<0>, ZPZV<0>, ZPZV<0>, ZPZV<1>, ZPZV<444>, ZPZV<129>, ZPZV<604»;
         ); // NOLINT
04121 template<> struct ConwayPolynomial<613, 1> { using ZPZ = aerobus::zpz<613>; using type =
         POLYV<ZPZV<1>, ZPZV<611»; }; // NOLINT
04122 template<> struct ConwayPolynomial<613, 2> { using ZPZ = aerobus::zpz<613>; using type =
                                                                 // NOLINT
         POLYV<ZPZV<1>, ZPZV<609>, ZPZV<2»; };
04123 template<> struct ConwayPolynomial<613, 3> { using ZPZ = aerobus::zpz<613>; using type =
POLYV<ZPZV<1>, ZPZV<0>, ZPZV<6>, ZPZV<611»; }; // NOLINT
04124 template<> struct ConwayPolynomial<613, 4> { using ZPZ = aerobus::zpz<613>; using type =
POLYV<ZPZV<1>, ZPZV<0>, ZPZV<12>, ZPZV<333>, ZPZV<2»; }; // NOLINT
04125 template<> struct ConwayPolynomial<613, 5> { using ZPZ = aerobus::zpz<613>; using type =
POLYV<ZPZV<1>, ZPZV<0>, ZPZV<0>, ZPZV<0>, ZPZV<32>, ZPZV<611»; }; // NOLINT 04126 template<> struct ConwayPolynomial<613, 6> { using ZPZ = aerobus::zpz<613>; using type =
         POLYV<ZPZV<1>, ZPZV<0>, ZPZV<0>, ZPZV<609>, ZPZV<595>, ZPZV<601>, ZPZV<2»; }; // NOLINT
04127 template<> struct ConwayPolynomial<613, 7> { using ZPZ = aerobus::zpz<613>; using type =
         POLYV<ZPZV<1>, ZPZV<0>, ZPZV<0>, ZPZV<0>, ZPZV<0>, ZPZV<6>, ZPZV<6>, ZPZV<6>, ZPZV<61:»; };
04128 template<> struct ConwayPolynomial<613, 8> { using ZPZ = aerobus::zpz<613>; using type
         POLYV<ZPZV<1>, ZPZV<0>, ZPZV<0>, ZPZV<0>, ZPZV<6>, ZPZV<489>, ZPZV<57>, ZPZV<539>, ZPZV<2»; }; //
         NOLINT
04129 template<> struct ConwayPolynomial<613, 9> { using ZPZ = aerobus::zpz<613>; using type =
         POLYV<ZPZV<1>, ZPZV<0>, ZPZV<0>, ZPZV<0>, ZPZV<0>, ZPZV<0>, ZPZV<8>, ZPZV<513>, ZPZV<516>, ZPZV<611»;
04130 template<> struct ConwayPolynomial<617, 1> { using ZPZ = aerobus::zpz<617>; using type =
         POLYV<ZPZV<1>, ZPZV<614»; }; // NOLINT
04131 template<> struct ConwayPolynomial<617, 2> { using ZPZ = aerobus::zpz<617>; using type =
POLYV<ZPZV<1>, ZPZV<612, ZPZV<3»; }; // NOLINT
04132 template<> struct ConwayPolynomial<617, 3> { using ZPZ = aerobus::zpz<617>; using type =
         POLYV<ZPZV<1>, ZPZV<0>, ZPZV<3>, ZPZV<614»; }; // NOLINT
04133 template<> struct ConwayPolynomial<617, 4> { using ZPZ = aerobus::zpz<617>; using type =
         POLYV<ZPZV<1>, ZPZV<0>, ZPZV<2>, ZPZV<503>, ZPZV<3»; }; // NOLINT
04134 template<> struct ConwayPolynomial<617, 5> { using ZPZ = aerobus::zpz<617>; using type =
          \verb"POLYV<ZPZV<1>, \verb"ZPZV<0>, \verb"ZPZV<0>, \verb"ZPZV<4>, \verb"ZPZV<614"; \verb"}; $ // \verb"NOLINT" | NOLINT" 
04135 template<> struct ConwayPolynomial<617, 6> { using ZPZ = aerobus::zpz<617>; using type =
         POLYV<ZPZV<1>, ZPZV<0>, ZPZV<1>, ZPZV<318>, ZPZV<595>, ZPZV<310>, ZPZV<3»; }; // NOLINT
04136 template<> struct ConwayPolynomial<617, 7> { using ZPZ = aerobus::zpz<617>; using type
         POLYV<ZPZV<1>, ZPZV<0>, ZPZV<0>, ZPZV<0>, ZPZV<0>, ZPZV<0>, ZPZV<614»; };
04137 template<> struct ConwayPolynomial<617, 8> { using ZPZ = aerobus::zpz<617>; using type =
         POLYV<ZPZV<1>, ZPZV<0>, ZPZV<0>, ZPZV<0>, ZPZV<50>, ZPZV<501>, ZPZV<501>, ZPZV<155>, ZPZV<3»; }; //
         NOLINT
04138 template<> struct ConwayPolynomial<617, 9> { using ZPZ = aerobus::zpz<617>; using type =
         POLYV<ZPZV<1>, ZPZV<0>, ZPZV<0>, ZPZV<0>, ZPZV<0>, ZPZV<0>, ZPZV<0>, ZPZV<15>, ZPZV<388>, ZPZV<543>, ZPZV<614»;
04139 template<> struct ConwayPolynomial<619, 1> { using ZPZ = aerobus::zpz<619>; using type =
         POLYV<ZPZV<1>, ZPZV<617»; }; // NOLINT
04140 template<> struct ConwayPolynomial<619, 2> { using ZPZ = aerobus::zpz<619>; using type = POLYV<ZPZV<1>, ZPZV<618>, ZPZV<2»; }; // NOLINT
04141 template<> struct ConwayPolynomial<619, 3> { using ZPZ = aerobus::zpz<619>; using type =
         POLYV<ZPZV<1>, ZPZV<0>, ZPZV<6>, ZPZV<617»; }; // NOLINT
04142 template<> struct ConwayPolynomial<619, 4> { using ZPZ = aerobus::zpz<619>; using type =
POLYV<ZPZV<1>, ZPZV<0>, ZPZV<6>, ZPZV<492>, ZPZV<2»; }; // NOLINT
04143 template<> struct ConwayPolynomial<619, 5> { using ZPZ = aerobus::zpz<619>; using type =
         POLYV<ZPZV<1>, ZPZV<0>, ZPZV<0>, ZPZV<0>, ZPZV<8>, ZPZV<617»; }; // NOLINT
04144 template<> struct ConwayPolynomial<619, 6> { using ZPZ = aerobus::zpz<619>; using type =
         POLYV<ZPZV<1>, ZPZV<0>, ZPZV<0>, ZPZV<238>, ZPZV<468>, ZPZV<347>, ZPZV<2»; }; // NOLINT
04145 template<> struct ConwayPolynomial<619, 7> { using ZPZ = aerobus::zpz<619>; using type
         POLYV<ZPZV<1>, ZPZV<0>, ZPZV<0>, ZPZV<0>, ZPZV<0>, ZPZV<0>, ZPZV<7>, ZPZV<7>, ZPZV<617»; };
04146 template<> struct ConwayPolynomial<619, 8> { using ZPZ = aerobus::zpz<619>; using type =
         POLYV<ZPZV<1>, ZPZV<0>, ZPZV<0>, ZPZV<0>, ZPZV<0>, ZPZV<10>, ZPZV<416>, ZPZV<383>, ZPZV<225>, ZPZV<2»; }; //
         NOLINT
04147 template<> struct ConwayPolynomial<619, 9> { using ZPZ = aerobus::zpz<619>; using type :
         POLYV<ZPZV<1>, ZPZV<0>, ZPZV<0>, ZPZV<0>, ZPZV<0>, ZPZV<0>, ZPZV<0>, ZPZV<0>, ZPZV<579>, ZPZV<510>, ZPZV<617»;
         }; // NOLINT
04148 template<> struct ConwayPolynomial<631, 1> { using ZPZ = aerobus::zpz<631>; using type =
         POLYV<ZPZV<1>, ZPZV<628»; }; // NOLINT
04149 template<> struct ConwayPolynomial<631, 2> { using ZPZ = aerobus::zpz<631>; using type =
POLYV<ZPZV<1>, ZPZV<629>, ZPZV<3»; }; // NOLINT
04150 template<> struct ConwayPolynomial<631, 3> { using ZPZ = aerobus::zpz<631>; using type =
         POLYV<ZPZV<1>, ZPZV<0>, ZPZV<5>, ZPZV<628»; }; // NOLINT
04151 template<> struct ConwayPolynomial<631, 4> { using ZPZ = aerobus::zpz<631>; using type = POLYV<ZPZV<1>, ZPZV<0>, ZPZV<6>, ZPZV<376>, ZPZV<3»; }; // NOLINT
04152 template<> struct ConwayPolynomial<631, 5> { using ZPZ = aerobus::zpz<631>; using type =
```

```
POLYV<ZPZV<1>, ZPZV<0>, ZPZV<0>, ZPZV<5>, ZPZV<628»; }; // NOLINT
04153 template<> struct ConwayPolynomial<631, 6> { using ZPZ = aerobus::zpz<631>; using type =
      POLYV<2PZV<1>, 2PZV<0>, ZPZV<0>, ZPZV<516>, ZPZV<541>, ZPZV<106>, ZPZV<3»; }; // NOLINT
04154 template<> struct ConwayPolynomial<631, 7> { using ZPZ = aerobus::zpz<631>; using type =
      POLYV<ZPZV<1>, ZPZV<0>, ZPZV<0>, ZPZV<0>, ZPZV<0>, ZPZV<0>, ZPZV<5>, ZPZV<628»; }; // NOLINT
04155 template<> struct ConwayPolynomial<631, 8> { using ZPZ = aerobus::zpz<631>; using type
       POLYV<ZPZV<1>, ZPZV<0>, ZPZV<0>, ZPZV<0>, ZPZV<2>, ZPZV<379>, ZPZV<516>, ZPZV<187>, ZPZV<3»; }; //
04156 template<> struct ConwayPolynomial<631, 9> { using ZPZ = aerobus::zpz<631>; using type =
       POLYV<ZPZV<1>, ZPZV<0>, ZPZV<0>, ZPZV<0>, ZPZV<0>, ZPZV<0>, ZPZV<3>, ZPZV<296>, ZPZV<413>, ZPZV<628»;
       }; // NOLINT
04157 template<> struct ConwavPolvnomial<641, 1> { using ZPZ = aerobus::zpz<641>; using type =
       POLYV<ZPZV<1>, ZPZV<638»; }; // NOLINT
04158 template<> struct ConwayPolynomial<641, 2> { using ZPZ = aerobus::zpz<641>; using type =
       POLYV<ZPZV<1>, ZPZV<635>, ZPZV<3»; }; // NOLINT
04159 template<> struct ConwayPolynomial<641, 3> { using ZPZ = aerobus::zpz<641>; using type =
POLYV<ZPZV<1>, ZPZV<0>, ZPZV<4>, ZPZV<638»; }; // NOLINT
04160 template<> struct ConwayPolynomial<641, 4> { using ZPZ = aerobus::zpz<641>; using type =
      POLYV<ZPZV<1>, ZPZV<0>, ZPZV<0>, ZPZV<629>, ZPZV<3»; }; // NOLINT
04161 template<> struct ConwayPolynomial<641, 5> { using ZPZ = aerobus::zpz<641>; using type =
       POLYV<ZPZV<1>, ZPZV<0>, ZPZV<0>, ZPZV<0>, ZPZV<1>, ZPZV<638»; }; // NOLINT
04162 template<> struct ConwayPolynomial<641, 6> { using ZPZ = aerobus::zpz<641>; using type =
POLYV<ZPZV<1>, ZPZV<0>, ZPZV<2>, ZPZV<105>, ZPZV<557>, ZPZV<294>, ZPZV<3»; }; // NOLINT 04163 template<> struct ConwayPolynomial<641, 7> { using ZPZ = aerobus::zpz<641>; using type =
      POLYV<ZPZV<1>, ZPZV<0>, ZPZV<0>, ZPZV<0>, ZPZV<0>, ZPZV<0>, ZPZV<0>, ZPZV<3>, ZPZV<638»; };
                                                                                                      // NOLINT
04164 template<> struct ConwayPolynomial<641, 8> { using ZPZ = aerobus::zpz<641>; using type
       POLYV<ZPZV<1>, ZPZV<0>, ZPZV<0>, ZPZV<0>, ZPZV<3>, ZPZV<356>, ZPZV<392>, ZPZV<332>, ZPZV<33; }; //
      NOT.TNT
04165 template<> struct ConwayPolynomial<641, 9> { using ZPZ = aerobus::zpz<641>; using type = POLYV<ZPZV<1>, ZPZV<0>, ZPZV<0>, ZPZV<0>, ZPZV<0>, ZPZV<0>, ZPZV<0>, ZPZV<66>, ZPZV<141>, ZPZV<638»;
       }; // NOLINT
04166 template<> struct ConwayPolynomial<643, 1> { using ZPZ = aerobus::zpz<643>; using type =
       POLYV<ZPZV<1>, ZPZV<632»; }; // NOLINT
04167 template<> struct ConwayPolynomial<643, 2> { using ZPZ = aerobus::zpz<643>; using type =
POLYV<ZPZV<1>, ZPZV<641, ZPZV<11»; }; // NOLINT
04168 template<> struct ConwayPolynomial<643, 3> { using ZPZ = aerobus::zpz<643>; using type =
      POLYV<ZPZV<1>, ZPZV<0>, ZPZV<1>, ZPZV<632»; }; // NOLINT
04169 template<> struct ConwayPolynomial<643, 4> { using ZPZ = aerobus::zpz<643>; using type =
      POLYV<ZPZV<1>, ZPZV<0>, ZPZV<0>, ZPZV<600>, ZPZV<11»; }; // NOLINT
04170 template<> struct ConwayPolynomial<643, 5> { using ZPZ = aerobus::zpz<643>; using type =
      POLYV<ZPZV<1>, ZPZV<0>, ZPZV<0>, ZPZV<0>, ZPZV<2>, ZPZV<632»; }; // NOLINT
04171 template<> struct ConwayPolynomial<643, 6> { using ZPZ = aerobus::zpz<643>; using type =
POLYV<ZPZV<1>, ZPZV<0>, ZPZV<0>, ZPZV<345>, ZPZV<412>, ZPZV<293>, ZPZV<11»; }; // NOLINT 04172 template<> struct ConwayPolynomial<643, 7> { using ZPZ = aerobus::zpz<643>; using type =
      POLYV<ZPZV<1>, ZPZV<0>, ZPZV<0>, ZPZV<0>, ZPZV<0>, ZPZV<5>, ZPZV<632»; }; // NOLINT
04173 template<> struct ConwayPolynomial<643, 8> { using ZPZ = aerobus::zpz<643>; using type =
       POLYV<ZPZV<1>, ZPZV<0>, ZPZV<0>, ZPZV<0>, ZPZV<1>, ZPZV<631>, ZPZV<573>, ZPZV<569>, ZPZV<11»; }; //
      NOLINT
04174 template<> struct ConwayPolynomial<643, 9> { using ZPZ = aerobus::zpz<643>; using type =
      POLYV<ZPZV<1>, ZPZV<0>, ZPZV<0>, ZPZV<0>, ZPZV<0>, ZPZV<0>, ZPZV<3>, ZPZV<591>, ZPZV<475>, ZPZV<632»;
       }; // NOLINT
04175 template<> struct ConwayPolynomial<647, 1> { using ZPZ = aerobus::zpz<647>; using type =
      POLYV<ZPZV<1>, ZPZV<642»; }; // NOLINT
04176 template<> struct ConwayPolynomial<647, 2> { using ZPZ = aerobus::zpz<647>; using type =
POLYV<ZPZV<1>, ZPZV<645>, ZPZV<5»; }; // NOLINT
04177 template<> struct ConwayPolynomial<647, 3> { using ZPZ = aerobus::zpz<647>; using type =
       POLYV<ZPZV<1>, ZPZV<0>, ZPZV<6>, ZPZV<642»; }; // NOLINT
04178 template<> struct ConwayPolynomial<647, 4> { using ZPZ = aerobus::zpz<647>; using type =
POLYV<ZPZV<1>, ZPZV<3>, ZPZV<643>, ZPZV<643>, ZPZV<5»; }; // NOLINT
04179 template<> struct ConwayPolynomial<647, 5> { using ZPZ = aerobus::zpz<647>; using type =
POLYV<ZPZV<1>, ZPZV<0>, ZPZV<0>, ZPZV<0>, ZPZV<11>, ZPZV<642»; }; // NOLINT
04180 template<> struct ConwayPolynomial<647, 6> { using ZPZ = aerobus::zpz<647>; using type =
      POLYV<2PZV<1>, 2PZV<0>, ZPZV<5>, ZPZV<308>, ZPZV<385>, ZPZV<642>, ZPZV<5»; }; // NOLINT
04181 template<> struct ConwayPolynomial<647, 7> { using ZPZ = aerobus::zpz<647>;
                                                                                             using type
       POLYV<ZPZV<1>, ZPZV<0>, ZPZV<0>, ZPZV<0>, ZPZV<0>, ZPZV<0>, ZPZV<1>, ZPZV<642»; };
04182 template<> struct ConwayPolynomial<647, 8> { using ZPZ = aerobus::zpz<647>; using type =
      POLYV<ZPZV<1>, ZPZV<0>, ZPZV<0>, ZPZV<9>, ZPZV<603>, ZPZV<259>, ZPZV<271>, ZPZV<5»; }; //
       NOLINT
04183 template<> struct ConwayPolynomial<647, 9> { using ZPZ = aerobus::zpz<647>; using type =
       POLYV<ZPZV<1>, ZPZV<0>, ZPZV<0>, ZPZV<0>, ZPZV<0>, ZPZV<0>, ZPZV<13>, ZPZV<561>, ZPZV<123>, ZPZV<642»;
       }; // NOLINT
04184 template<> struct ConwayPolynomial<653, 1> { using ZPZ = aerobus::zpz<653>; using type =
       POLYV<ZPZV<1>, ZPZV<651»; }; // NOLINT
04185 template<> struct ConwayPolynomial<653, 2> { using ZPZ = aerobus::zpz<653>; using type =
POLYV<ZPZV<1>, ZPZV<649>, ZPZV<2»; }; // NOLINT
04186 template<> struct ConwayPolynomial<653, 3> { using ZPZ = aerobus::zpz<653>; using type =
       POLYV<ZPZV<1>, ZPZV<0>, ZPZV<3>, ZPZV<651»; }; // NOLINT
04187 template<> struct ConwayPolynomial<653, 4> { using ZPZ = aerobus::zpz<653>; using type = POLYV<ZPZV<1>, ZPZV<0>, ZPZV<6>, ZPZV<596>, ZPZV<2»; }; // NOLINT
04188 template<> struct ConwayPolynomial<653, 5> { using ZPZ = aerobus::zpz<653>; using type =
      POLYV<ZPZV<1>, ZPZV<0>, ZPZV<0>, ZPZV<5>, ZPZV<651»; }; // NOLINT
04189 template<> struct ConwayPolynomial<653, 6> { using ZPZ = aerobus::zpz<653>; using type =
      POLYV<ZPZV<1>, ZPZV<0>, ZPZV<2>, ZPZV<45>, ZPZV<220>, ZPZV<242>, ZPZV<24>; }; // NOLINT
04190 template<> struct ConwayPolynomial<653, 7> { using ZPZ = aerobus::zpz<653>; using type =
      POLYV<ZPZV<1>, ZPZV<0>, ZPZV<0>, ZPZV<0>, ZPZV<0>, ZPZV<0>, ZPZV<1>, ZPZV<651»; }; // NOLINT
04191 template<> struct ConwayPolynomial<653, 8> { using ZPZ = aerobus::zpz<653>; using type = POLYV<ZPZV<1>, ZPZV<0>, ZPZV<0>, ZPZV<0>, ZPZV<0>, ZPZV<385>, ZPZV<18>, ZPZV<296>, ZPZV<2»; }; //
```

```
NOLINT
04192 template<> struct ConwayPolynomial<653, 9> { using ZPZ = aerobus::zpz<653>; using type =
       POLYV<ZPZV<1>, ZPZV<0>, ZPZV<0>, ZPZV<0>, ZPZV<0>, ZPZV<0>, ZPZV<0>, ZPZV<651»;
       }; // NOLINT
04193 template<> struct ConwayPolynomial<659, 1> { using ZPZ = aerobus::zpz<659>; using type =
       POLYV<ZPZV<1>, ZPZV<657»; }; // NOLINT
04194 template<> struct ConwayPolynomial<659, 2> { using ZPZ = aerobus::zpz<659>; using type =
       POLYV<ZPZV<1>, ZPZV<655>, ZPZV<2»; };
                                                   // NOLINT
04195 template<> struct ConwayPolynomial<659, 3> { using ZPZ = aerobus::zpz<659>; using type =
POLYV<ZPZV<1>, ZPZV<0>, ZPZV<2>, ZPZV<657»; }; // NOLINT
04196 template<> struct ConwayPolynomial<659, 4> { using ZPZ = aerobus::zpz<659>; using type =
POLYV<ZPZV<1>, ZPZV<0>, ZPZV<8>, ZPZV<351>, ZPZV<2»; }; // NOLINT
04197 template<> struct ConwayPolynomial<659, 5> { using ZPZ = aerobus::zpz<659>; using type =
       POLYV<ZPZV<1>, ZPZV<0>, ZPZV<0>, ZPZV<4>, ZPZV<657»; }; // NOLINT
04198 template<> struct ConwayPolynomial<659, 6> { using ZPZ = aerobus::zpz<659>; using type =
POLYV<ZPZV<1>, ZPZV<0>, ZPZV<6>, ZPZV<371>, ZPZV<105>, ZPZV<223>, ZPZV<2»; }; // NOLINT 04199 template<> struct ConwayPolynomial<659, 7> { using ZPZ = aerobus::zpz<659>; using type =
       POLYV<ZPZV<1>, ZPZV<0>, ZPZV<0>, ZPZV<0>, ZPZV<0>, ZPZV<0>, ZPZV<0>, ZPZV<5>, ZPZV<657»; };
                                                                                                       // NOLINT
04200 template<> struct ConwayPolynomial<659, 8> { using ZPZ = aerobus::zpz<659>; using type =
       POLYV<ZPZV<1>, ZPZV<0>, ZPZV<0>, ZPZV<0>, ZPZV<6>, ZPZV<358>, ZPZV<246>, ZPZV<90>, ZPZV<2»; }; //
       NOLINT
04201 template<> struct ConwayPolynomial<659, 9> { using ZPZ = aerobus::zpz<659>; using type =
       POLYV<ZPZV<1>, ZPZV<0>, ZPZV<0>, ZPZV<0>, ZPZV<0>, ZPZV<0>, ZPZV<2>, ZPZV<592>, ZPZV<46>, ZPZV<657»;
       }: // NOLINT
04202 template<> struct ConwayPolynomial<661, 1> { using ZPZ = aerobus::zpz<661>; using type =
       POLYV<ZPZV<1>, ZPZV<659»; }; // NOLINT
04203 template<> struct ConwayPolynomial<661, 2> { using ZPZ = aerobus::zpz<661>; using type =
       POLYV<ZPZV<1>, ZPZV<660>, ZPZV<2»; }; // NOLINT
04204 template<> struct ConwayPolynomial<661, 3> { using ZPZ = aerobus::zpz<661>; using type =
POLYV<ZPZV<1>, ZPZV<0>, ZPZV<4>, ZPZV<659»; }; // NOLINT
04205 template<> struct ConwayPolynomial<661, 4> { using ZPZ = aerobus::zpz<661>; using type =
POLYV<ZPZV<1>, ZPZV<0>, ZPZV<7>, ZPZV<616>, ZPZV<2»; }; // NOLINT
04206 template<> struct ConwayPolynomial<661, 5> { using ZPZ = aerobus::zpz<661>; using type =
       POLYV<ZPZV<1>, ZPZV<0>, ZPZV<0>, ZPZV<0>, ZPZV<19>, ZPZV<659»; }; // NOLINT
04207 template<> struct ConwayPolynomial<661, 6> { using ZPZ = aerobus::zpz<661>; using type = POLYV<ZPZV<1>, ZPZV<0>, ZPZV<0>, ZPZV<551>, ZPZV<456>, ZPZV<382>, ZPZV<2»; }; // NOLINT
04208 template<> struct ConwayPolynomial<661, 7> { using ZPZ = aerobus::zpz<661>; using type
       POLYV<ZPZV<1>, ZPZV<0>, ZPZV<0>, ZPZV<0>, ZPZV<0>, ZPZV<0>, ZPZV<0>, ZPZV<2>, ZPZV<659»; }; // NOLINT
04209 template<> struct ConwayPolynomial<661, 8> { using ZPZ = aerobus::zpz<661>; using type
       POLYV<ZPZV<1>, ZPZV<0>, ZPZV<0>, ZPZV<0>, ZPZV<0>, ZPZV<612>, ZPZV<285>, ZPZV<72>, ZPZV<72»; }; //
       NOT.TNT
04210 template<> struct ConwayPolynomial<661, 9> { using ZPZ = aerobus::zpz<661>; using type =
       POLYV<ZPZV<1>, ZPZV<0>, ZPZV<0>, ZPZV<0>, ZPZV<0>, ZPZV<659»;
       }; // NOLINT
04211 template<> struct ConwayPolynomial<673, 1> { using ZPZ = aerobus::zpz<673>; using type =
       POLYV<ZPZV<1>, ZPZV<668»; }; // NOLINT
04212 template<> struct ConwayPolynomial<673, 2> { using ZPZ = aerobus::zpz<673>; using type =
POLYV<ZPZV<1>, ZPZV<672>, ZPZV<5»; }; // NOLINT
04213 template<> struct ConwayPolynomial<673, 3> { using ZPZ = aerobus::zpz<673>; using type =
       POLYV<ZPZV<1>, ZPZV<0>, ZPZV<1>, ZPZV<668»; }; // NOLINT
04214 template<> struct ConwayPolynomial<673, 4> { using ZPZ = aerobus::zpz<673>; using type =
POLYV<ZPZV<1>, ZPZV<6>, ZPZV<6>, ZPZV<416>, ZPZV<5»; }; // NOLINT
04215 template<> struct ConwayPolynomial<673, 5> { using ZPZ = aerobus::zpz<673>; using type =
POLYV<ZPZV<1>, ZPZV<0>, ZPZV<0>, ZPZV<0>, ZPZV<0>, ZPZV<6668»; }; // NOLINT
04216 template<> struct ConwayPolynomial<673, 6> { using ZPZ = aerobus::zpz<673>; using type =
       POLYV<ZPZV<1>, ZPZV<0>, ZPZV<0>, ZPZV<524>, ZPZV<248>, ZPZV<35>, ZPZV<5»; }; // NOLINT
04217 template<> struct ConwayPolynomial<673, 7> { using ZPZ = aerobus::zpz<673>; using type
       POLYV<ZPZV<1>, ZPZV<0>, ZPZV<0>, ZPZV<0>, ZPZV<0>, ZPZV<0>, ZPZV<6>, ZPZV<6>, ZPZV<68»; };
04218 template<> struct ConwayPolynomial<673, 8> { using ZPZ = aerobus::zpz<673>; using type =
       POLYV<ZPZV<1>, ZPZV<0>, ZPZV<0>, ZPZV<6>, ZPZV<669>, ZPZV<587>, ZPZV<302>, ZPZV<5»; }; //
       NOLINT
04219 template<> struct ConwayPolynomial<673, 9> { using ZPZ = aerobus::zpz<673>; using type = POLYV<ZPZV<1>, ZPZV<0>, ZPZV<0>, ZPZV<0>, ZPZV<0>, ZPZV<1>, ZPZV<347>, ZPZV<553>, ZPZV<668»;
       }; // NOLINT
04220 template<> struct ConwayPolynomial<677, 1> { using ZPZ = aerobus::zpz<677>; using type =
       POLYV<ZPZV<1>, ZPZV<675»; }; // NOLINT
04221 template<> struct ConwayPolynomial<677, 2> { using ZPZ = aerobus::zpz<677>; using type =
POLYY<ZPZV<1>, ZPZV<672>, ZPZV<2»; }; // NOLINT
04222 template<> struct ConwayPolynomial<677, 3> { using ZPZ = aerobus::zpz<677>; using type =
       POLYV<ZPZV<1>, ZPZV<0>, ZPZV<2>, ZPZV<675»; }; // NOLINT
04223 template<> struct ConwayPolynomial<677, 4> { using ZPZ = aerobus::zpz<677>; using type = POLYV<ZPZV<1>, ZPZV<0>, ZPZV<0>, ZPZV<631>, ZPZV<2»; }; // NOLINT
04224 template<> struct ConwayPolynomial<677, 5> { using ZPZ = aerobus::zpz<677>; using type =
       POLYV<ZPZV<1>, ZPZV<0>, ZPZV<0>, ZPZV<5>, ZPZV<675»; }; // NOLINT
04225 template<> struct ConwayPolynomial<677, 6> { using ZPZ = aerobus::zpz<677>; using type =
       POLYV<ZPZV<1>, ZPZV<0>, ZPZV<1>, ZPZV<446>, ZPZV<632>, ZPZV<50>, ZPZV<2»; }; // NOLINT
04226 template<> struct ConwayPolynomial<677, 7> { using ZPZ = aerobus::zpz<677>; using type
       POLYV<ZPZV<1>, ZPZV<0>, ZPZV<0>, ZPZV<0>, ZPZV<0>, ZPZV<0>, ZPZV<10>, ZPZV<675»; }; // NOLINT
04227 template<> struct ConwayPolynomial<677, 8> { using ZPZ = aerobus::zpz<677>; using type =
       POLYV<ZPZV<1>, ZPZV<0>, ZPZV<0>, ZPZV<0>, ZPZV<0>, ZPZV<3>, ZPZV<619>, ZPZV<519>, ZPZV<152>, ZPZV<2»; }; //
       NOLINT
04228 template<> struct ConwayPolynomial<677, 9> { using ZPZ = aerobus::zpz<677>; using type =
       POLYV<ZPZV<1>, ZPZV<0>, ZPZV<0>, ZPZV<0>, ZPZV<0>, ZPZV<0>, ZPZV<0>, ZPZV<0>, ZPZV<504>, ZPZV<504>, ZPZV<404>, ZPZV<675»;
       }; // NOLINT
04229 template<> struct ConwayPolynomial<683, 1> { using ZPZ = aerobus::zpz<683>; using type =
       POLYV<ZPZV<1>, ZPZV<678»; }; // NOLINT
04230 template<> struct ConwayPolynomial<683, 2> { using ZPZ = aerobus::zpz<683>; using type =
```

```
POLYV<ZPZV<1>, ZPZV<682>, ZPZV<5»; };
                                                                                    // NOLINT
04231 template<> struct ConwayPolynomial<683, 3> { using ZPZ = aerobus::zpz<683>; using type =
           POLYV<ZPZV<1>, ZPZV<0>, ZPZV<5>, ZPZV<678»; }; // NOLINT
04232 template<> struct ConwayPolynomial<683, 4> { using ZPZ = aerobus::zpz<683>; using type =
POLYV<ZPZV<1>, ZPZV<0>, ZPZV<5>, ZPZV<455>, ZPZV<5>, ZPZV<5>, ZPZV<6»; }; // NOLINT
04233 template<> struct ConwayPolynomial<683, 5> { using ZPZ = aerobus::zpz<683>; using type =
           POLYV<ZPZV<1>, ZPZV<0>, ZPZV<0>, ZPZV<7>, ZPZV<678»; }; // NOLINT
04234 template<> struct ConwayPolynomial<683, 6> { using ZPZ = aerobus::zpz<683>; using type =
           POLYV<ZPZV<1>, ZPZV<0>, ZPZV<2>, ZPZV<644>, ZPZV<109>, ZPZV<434>, ZPZV<5»; }; // NOLINT
04235 template<> struct ConwayPolynomial<683, 7> { using ZPZ = aerobus::zpz<683>; using type :
           POLYV<ZPZV<1>, ZPZV<0>, ZPZV<0>, ZPZV<0>, ZPZV<0>, ZPZV<0>, ZPZV<30>, ZPZV<678»; }; // NOLINT
04236 template<> struct ConwayPolynomial<683, 8> { using ZPZ = aerobus::zpz<683>; using type =
           POLYV<ZPZV<1>, ZPZV<0>, ZPZV<0>, ZPZV<0>, ZPZV<0>, ZPZV<383>, ZPZV<184>, ZPZV<65>, ZPZV<65»; }; //
04237 template<> struct ConwayPolynomial<683, 9> { using ZPZ = aerobus::zpz<683>; using type =
           POLYV<ZPZV<1>, ZPZV<0>, ZPZV<0
            1: // NOT.TNT
04238 template<> struct ConwayPolynomial<691, 1> { using ZPZ = aerobus::zpz<691>; using type =
           POLYV<ZPZV<1>, ZPZV<688»; }; // NOLINT
04239 template<> struct ConwayPolynomial<691, 2> { using ZPZ = aerobus::zpz<691>; using type =
           POLYV<ZPZV<1>, ZPZV<686>, ZPZV<3»; }; // NOLINT
04240 template<> struct ConwayPolynomial<691, 3> { using ZPZ = aerobus::zpz<691>; using type =
           POLYV<ZPZV<1>, ZPZV<0>, ZPZV<14>, ZPZV<688»; }; // NOLINT
04241 template<> struct ConwayPolynomial<691, 4> { using ZPZ = aerobus::zpz<691>; using type = POLYV<ZPZV<1>, ZPZV<0>, ZPZV<3>, ZPZV<632>, ZPZV<3»; }; // NOLINT
04242 template<> struct ConwayPolynomial<691, 5> { using ZPZ = aerobus::zpz<691>; using type =
           POLYV<ZPZV<1>, ZPZV<0>, ZPZV<0>, ZPZV<0>, ZPZV<4>, ZPZV<688»; }; // NOLINT
04243 template<> struct ConwayPolynomial<691, 6> { using ZPZ = aerobus::zpz<691>; using type =
POLYV<ZPZV<1>, ZPZV<0>, ZPZV<0>, ZPZV<579>, ZPZV<408>, ZPZV<262>, ZPZV<3»; }; // NOLINT
04244 template<> struct ConwayPolynomial<691, 7> { using ZPZ = aerobus::zpz<691>; using type =
POLYV<ZPZV<1>, ZPZV<0>, ZPZV<0>, ZPZV<0>, ZPZV<0>, ZPZV<0>, ZPZV<4>, ZPZV<688»; }; // NOLINT
04245 template<> struct ConwayPolynomial<691, 8> { using ZPZ = aerobus::zpz<691>; using type
            POLYV<ZPZV<1>, ZPZV<0>, ZPZV<0>, ZPZV<0>, ZPZV<7>, ZPZV<356>, ZPZV<425>, ZPZV<321>, ZPZV<3»; }; //
           NOLINT
04246 template<> struct ConwayPolynomial<691, 9> { using ZPZ = aerobus::zpz<691>; using type = POLYV<ZPZV<1>, ZPZV<0>, ZPZV<0>, ZPZV<0>, ZPZV<0>, ZPZV<0>, ZPZV<5>, ZPZV<556>, ZPZV<556>, ZPZV<4443>, ZPZV<688»;
            }; // NOLINT
04247 template<> struct ConwayPolynomial<701, 1> { using ZPZ = aerobus::zpz<701>; using type =
           POLYV<ZPZV<1>, ZPZV<699»; }; // NOLINT
04248 template<> struct ConwayPolynomial<701, 2> { using ZPZ = aerobus::zpz<701>; using type =
POLYV<ZPZV<1>, ZPZV<697, ZPZV<2»; }; // NOLINT
04249 template<> struct ConwayPolynomial<701, 3> { using ZPZ = aerobus::zpz<701>; using type =
           POLYV<ZPZV<1>, ZPZV<0>, ZPZV<2>, ZPZV<699»; }; // NOLINT
04250 template<> struct ConwayPolynomial<701, 4> { using ZPZ = aerobus::zpz<701>; using type =
POLYV<ZPZV<1>, ZPZV<0>, ZPZV<12>, ZPZV<379>, ZPZV<2»; }; // NOLINT
04251 template<> struct ConwayPolynomial<701, 5> { using ZPZ = aerobus::zpz<701>; using type =
           POLYV<ZPZV<1>, ZPZV<0>, ZPZV<0>, ZPZV<5>, ZPZV<699»; }; // NOLINT
04252 template<> struct ConwayPolynomial<701, 6> { using ZPZ = aerobus::zpz<701>; using type =
POLYV<ZPZV<1>, ZPZV<0>, ZPZV<1>, ZPZV<571>, ZPZV<571>, ZPZV<327>, ZPZV<285>, ZPZV<2»; }; // NOLINT
04253 template<> struct ConwayPolynomial<701, 7> { using ZPZ = aerobus::zpz<701>; using type =
POLYV<ZPZV<1>, ZPZV<0>, ZPZV<0 , Z
04254 template<> struct ConwayPolynomial<701, 8> { using ZPZ = aerobus::zpz<701>; using type =
           POLYV<ZPZV<1>, ZPZV<0>, ZPZV<0>, ZPZV<0>, ZPZV<1>, ZPZV<619>, ZPZV<206>, ZPZV<593>, ZPZV<2»; }; //
           NOLINT
04255 template<> struct ConwayPolynomial<701, 9> { using ZPZ = aerobus::zpz<701>; using type =
           POLYV<ZPZV<1>, ZPZV<0>, ZPZV<0>, ZPZV<0>, ZPZV<0>, ZPZV<0>, ZPZV<1>, ZPZV<45>, ZPZV<459>, ZPZV<373>, ZPZV<699»;
04256 template<> struct ConwayPolynomial<709, 1> { using ZPZ = aerobus::zpz<709>; using type =
           POLYV<ZPZV<1>, ZPZV<707»; }; // NOLINT
04257 template<> struct ConwayPolynomial<709, 2> { using ZPZ = aerobus::zpz<709>; using type =
POLYV<ZPZV<1>, ZPZV<705>, ZPZV<2»; }; // NOLINT
04258 template<> struct ConwayPolynomial<709, 3> { using ZPZ = aerobus::zpz<709>; using type =
           POLYV<ZPZV<1>, ZPZV<0>, ZPZV<2>, ZPZV<707»; }; // NOLINT
04259 template<> struct ConwayPolynomial<709, 4> { using ZPZ = aerobus::zpz<709>; using type = POLYV<ZPZV<1>, ZPZV<0>, ZPZV<6>, ZPZV<384>, ZPZV<2»; }; // NOLINT
04260 template<> struct ConwayPolynomial<709, 5> { using ZPZ = aerobus::zpz<709>; using type =
POLYV<ZPZV<1>, ZPZV<0>, ZPZV<0>, ZPZV<0>, ZPZV<10>, ZPZV<70*, ZPZV<707*, }; // NOLINT
04261 template<> struct ConwayPolynomial<709, 6> { using ZPZ = aerobus::zpz<709>; using type =
           POLYV<ZPZV<1>, ZPZV<0>, ZPZV<0>, ZPZV<66>, ZPZV<514>, ZPZV<295>, ZPZV<2»; }; // NOLINT
04262 template<> struct ConwayPolynomial<709, 7> { using ZPZ = aerobus::zpz<709>;
                                                                                                                                                            using type
           POLYV<ZPZV<1>, ZPZV<0>, ZPZV<0>, ZPZV<0>, ZPZV<0>, ZPZV<0>, ZPZV<0>, ZPZV<4>, ZPZV<707»; };
04263 template<> struct ConwayPolynomial<709, 8> { using ZPZ = aerobus::zpz<709>; using type =
           POLYV<ZPZV<1>, ZPZV<0>, ZPZV<0>, ZPZV<0>, ZPZV<2>, ZPZV<689>, ZPZV<233>, ZPZV<79>, ZPZV<2»; }; //
           NOLINT
04264 template<> struct ConwayPolynomial<709, 9> { using ZPZ = aerobus::zpz<709>; using type
           POLYV<ZPZV<1>, ZPZV<0>, ZPZV<0>, ZPZV<0>, ZPZV<0>, ZPZV<0>, ZPZV<0>, ZPZV<3>, ZPZV<257>, ZPZV<171>, ZPZV<707»;
            }; // NOLINT
04265 template<> struct ConwayPolynomial<719, 1> { using ZPZ = aerobus::zpz<719>; using type =
           POLYV<ZPZV<1>, ZPZV<708»; }; // NOLINT
04266 template<> struct ConwayPolynomial<719, 2> { using ZPZ = aerobus::zpz<719>; using type =
POLYY<ZPZY<1>, ZPZV<715>, ZPZV<71s, ZPZV<11s; ; // NOLINT

04267 template<> struct ConwayPolynomial<719, 3> { using ZPZ = aerobus::zpz<719>; using type =
           POLYV<ZPZV<1>, ZPZV<0>, ZPZV<1>, ZPZV<708»; }; // NOLINT
04268 template<> struct ConwayPolynomial<719, 4> { using ZPZ = aerobus::zpz<719>; using type =
POLYV<ZPZV<1>, ZPZV<0>, ZPZV<5>, ZPZV<602>, ZPZV<11»; }; // NOLINT
04269 template<> struct ConwayPolynomial<719, 5> { using ZPZ = aerobus::zpz<719>; using type = POLYV<ZPZV<1>, ZPZV<0>, ZPZV<0>, ZPZV<0>, ZPZV<1>, ZPZV<708»; }; // NOLINT
```

```
04270 template<> struct ConwayPolynomial<719, 6> { using ZPZ = aerobus::zpz<719>; using type :
POLYY<ZPZV<1>, ZPZV<0>, ZPZV<1>, ZPZV<533>, ZPZV<591>, ZPZV<182>, ZPZV<11%; }; // NOLINT 04271 template<> struct ConwayPolynomial<719, 7> { using ZPZ = aerobus::zpz<719>; using type =
         POLYV<ZPZV<1>, ZPZV<0>, ZPZV<0>, ZPZV<0>, ZPZV<0>, ZPZV<0>, ZPZV<1>, ZPZV<708»; }; // NOLINT
04272 template<> struct ConwayPolynomial<719, 8> { using ZPZ = aerobus::zpz<719>; using type
          POLYV<ZPZV<1>, ZPZV<0>, ZPZV<0>, ZPZV<0>, ZPZV<5>, ZPZV<714>, ZPZV<362>, ZPZV<244>, ZPZV<11»; }; //
04273 template<> struct ConwayPolynomial<719, 9> { using ZPZ = aerobus::zpz<719>; using type
          POLYV<ZPZV<1>, ZPZV<0>, ZPZV<0>, ZPZV<0>, ZPZV<0>, ZPZV<0>, ZPZV<3>, ZPZV<38>, ZPZV<288>, ZPZV<560>, ZPZV<708»;
          }; // NOLINT
04274 template<> struct ConwayPolynomial<727, 1> { using ZPZ = aerobus::zpz<727>; using type =
         POLYV<ZPZV<1>, ZPZV<722»; }; // NOLINT
04275 template<> struct ConwayPolynomial<727, 2> { using ZPZ = aerobus::zpz<727>; using type =
          POLYV<ZPZV<1>, ZPZV<725>, ZPZV<5»; }; // NOLINT
04276 template<> struct ConwayPolynomial<727, 3> { using ZPZ = aerobus::zpz<727>; using type =
          POLYV<ZPZV<1>, ZPZV<0>, ZPZV<7>, ZPZV<722»; }; // NOLINT
04277 template<> struct ConwayPolynomial<727, 4> { using ZPZ = aerobus::zpz<727>; using type =
POLYV<ZPZV<1>, ZPZV<3>, ZPZV<3>, ZPZV<723>, ZPZV<5»; }; // NOLINT

04278 template<> struct ConwayPolynomial<727, 5> { using ZPZ = aerobus::zpz<727>; using type =
          POLYV<ZPZV<1>, ZPZV<0>, ZPZV<0>, ZPZV<5>, ZPZV<722»; }; // NOLINT
04279 template<> struct ConwayPolynomial<727, 6> { using ZPZ = aerobus::zpz<727>; using type =
         POLYV<2PZV<1>, 2PZV<0>, 2PZV<0>, 2PZV<86>, ZPZV<397>, ZPZV<672>, ZPŽV<5»; }; // NOLINT
04280 template<> struct ConwayPolynomial<727, 7> { using ZPZ = aerobus::zpz<727>; using type =
POLYV<ZPZV<1>, ZPZV<0>, ZPZV<0>, ZPZV<0>, ZPZV<0>, ZPZV<1>, ZPZV<17>, ZPZV<722»; }; // NOLINT 04281 template<> struct ConwayPolynomial<727, 8> { using ZPZ = aerobus::zpz<727>; using type =
          POLYV<ZPZV<1>, ZPZV<0>, ZPZV<0>, ZPZV<0>, ZPZV<0>, ZPZV<639>, ZPZV<671>, ZPZV<368>, ZPZV<5»; }; //
04282 template<> struct ConwayPolynomial<727, 9> { using ZPZ = aerobus::zpz<727>; using type =
          POLYV<ZPZV<1>, ZPZV<0>, ZPZV<0>, ZPZV<0>, ZPZV<0>, ZPZV<0>, ZPZV<573>, ZPZV<573>, ZPZV<502>, ZPZV<722»;
          }; // NOLINT
04283 template<> struct ConwayPolynomial<733, 1> { using ZPZ = aerobus::zpz<733>; using type =
         POLYV<ZPZV<1>, ZPZV<727»; }; // NOLINT
04284 template<> struct ConwayPolynomial<733, 2> { using ZPZ = aerobus::zpz<733>; using type =
          POLYV<ZPZV<1>, ZPZV<732>, ZPZV<6»; }; // NOLINT
04285 template<> struct ConwayPolynomial<733, 3> { using ZPZ = aerobus::zpz<733>; using type =
POLYV<ZPZV<1>, ZPZV<0>, ZPZV<4>, ZPZV<727»; }; // NOLINT
04286 template<> struct ConwayPolynomial<733, 4> { using ZPZ = aerobus::zpz<733>; using type =
         POLYV<ZPZV<1>, ZPZV<0>, ZPZV<12>, ZPZV<539>, ZPZV<6»; }; // NOLINT
04287 template<> struct ConwayPolynomial<733, 5> { using ZPZ = aerobus::zpz<733>; using type =
          POLYV<ZPZV<1>, ZPZV<0>, ZPZV<0>, ZPZV<0>, ZPZV<8>, ZPZV<727»; }; // NOLINT
04288 template<> struct ConwayPolynomial<733, 6> { using ZPZ = aerobus::zpz<733>; using type =
POLYV<ZPZV<1>, ZPZV<0>, ZPZV<174>, ZPZV<549>, ZPZV<151>, ZPZV<6»; }; // NOLINT
04289 template<> struct ConwayPolynomial<733, 7> { using ZPZ = aerobus::zpz<733>; using type =
POLYV<ZPZV<1>, ZPZV<0>, ZPZV<0 , ZPZ
04290 template<> struct ConwayPolynomial<733, 8> { using ZPZ = aerobus::zpz<733>; using type
          POLYV<ZPZV<1>, ZPZV<0>, ZPZV<0>, ZPZV<0>, ZPZV<1>, ZPZV<532>, ZPZV<610>, ZPZV<142>, ZPZV<6»; }; //
04291 template<> struct ConwayPolynomial<733, 9> { using ZPZ = aerobus::zpz<733>; using type = POLYV<ZPZV<1>, ZPZV<0>, ZPZV<0>, ZPZV<0>, ZPZV<0>, ZPZV<2>, ZPZV<337>, ZPZV<6>, ZPZV<727»; };
          // NOLINT
04292 template<> struct ConwayPolynomial<739, 1> { using ZPZ = aerobus::zpz<739>; using type =
          POLYV<ZPZV<1>, ZPZV<736»; }; // NOLINT
04293 template<> struct ConwayPolynomial<739, 2> { using ZPZ = aerobus::zpz<739>; using type =
POLYV<ZPZV<1>, ZPZV<734>, ZPZV<3»; }; // NOLINT
04294 template<> struct ConwayPolynomial<739, 3> { using ZPZ = aerobus::zpz<739>; using type =
         POLYV<ZPZV<1>, ZPZV<0>, ZPZV<11>, ZPZV<736»; }; // NOLINT
04295 template<> struct ConwayPolynomial<739, 4> { using ZPZ = aerobus::zpz<739>; using type =
POLYV<ZPZV<1>, ZPZV<3>, ZPZV<678>, ZPZV<678>, ZPZV<3>; }; // NOLINT
04296 template<> struct ConwayPolynomial<739, 5> { using ZPZ = aerobus::zpz<739>; using type =
POLYV<ZPZV<1>, ZPZV<0>, ZPZV<0>, ZPZV<0>, ZPZV<9>, ZPZV<736»; }; // NOLINT
04297 template<> struct ConwayPolynomial<739, 6> { using ZPZ = aerobus::zpz<739>; using type =
         POLYV<ZPZV<1>, ZPZV<0>, ZPZV<0>, ZPZV<42>, ZPZV<447>, ZPZV<625>, ZPZV<3»; }; // NOLINT
04298 template<> struct ConwayPolynomial<739, 7> { using ZPZ = aerobus::zpz<739>; using type =
          POLYV<ZPZV<1>, ZPZV<0>, ZPZV<0>, ZPZV<0>, ZPZV<0>, ZPZV<0>, ZPZV<44>, ZPZV<736»; }; // NOLINT
04299 template<> struct ConwayPolynomial<739, 8> { using ZPZ = aerobus::zpz<739>; using type =
          POLYV<ZPZV<1>, ZPZV<0>, ZPZV<0>, ZPZV<0>, ZPZV<2>, ZPZV<401>, ZPZV<169>, ZPZV<25>, ZPZV<3»; }; //
          NOLINT
04300 template<> struct ConwayPolynomial<739, 9> { using ZPZ = aerobus::zpz<739>; using type = POLYV<ZPZV<1>, ZPZV<0>, ZPZV<0>, ZPZV<0>, ZPZV<0>, ZPZV<1>, ZPZV<616>, ZPZV<81>, ZPZV<81
          }; // NOLINT
04301 template<> struct ConwayPolynomial<743, 1> { using ZPZ = aerobus::zpz<743>; using type =
          POLYV<ZPZV<1>, ZPZV<738»; }; // NOLINT
04302 template<> struct ConwayPolynomial<743, 2> { using ZPZ = aerobus::zpz<743>; using type =
POLYV<ZPZV<1>, ZPZV<742, ZPZV<5»; }; // NOLINT
04303 template<> struct ConwayPolynomial<743, 3> { using ZPZ = aerobus::zpz<743>; using type =
          POLYV<ZPZV<1>, ZPZV<0>, ZPZV<3>, ZPZV<738»; }; // NOLINT
04304 template<> struct ConwayPolynomial<743, 4> { using ZPZ = aerobus::zpz<743>; using type =
POLYV<ZPZV<1>, ZPZV<0>, ZPZV<5>, ZPZV<425>, ZPZV<5»; }; // NOLINT
04305 template<> struct ConwayPolynomial<743, 5> { using ZPZ = aerobus::zpz<743>; using type =
         POLYV<ZPZV<1>, ZPZV<0>, ZPZV<0>, ZPZV<6>, ZPZV<6>, ZPZV<738»; }; // NOLINT
04306 template<> struct ConwayPolynomial<743, 6> { using ZPZ = aerobus::zpz<743>; using type =
          POLYV<ZPZV<1>, ZPZV<0>, ZPZV<1>, ZPZV<236>, ZPZV<471>, ZPZV<88>, ZPZV<5»; }; // NOLINT
04307 template<> struct ConwayPolynomial<743, 7> { using ZPZ = aerobus::zpz<743>; using type
         POLYV<ZPZV<1>, ZPZV<0>, ZPZV<0>, ZPZV<0>, ZPZV<0>, ZPZV<0>, ZPZV<6>, ZPZV<6>, ZPZV<738»; };
04308 template<> struct ConwayPolynomial<743, 8> { using ZPZ = aerobus::zpz<743>; using type =
          POLYV<ZPZV<1>, ZPZV<0>, ZPZV<0>, ZPZV<0>, ZPZV<0>, ZPZV<55; }; //
          NOLTNT
```

```
04309 template<> struct ConwayPolynomial<743, 9> { using ZPZ = aerobus::zpz<743>; using type =
              POLYV<ZPZV<1>, ZPZV<0>, ZPZV<0>, ZPZV<0>, ZPZV<0>, ZPZV<0>, ZPZV<3>, ZPZV<3>, ZPZV<327>, ZPZV<676>, ZPZV<738»;
              }; // NOLINT
04310 template<> struct ConwayPolynomial<751, 1> { using ZPZ = aerobus::zpz<751>; using type =
             POLYV<ZPZV<1>, ZPZV<748»; }; // NOLINT
04311 template<> struct ConwayPolynomial<751, 2> { using ZPZ = aerobus::zpz<751>; using type =
             POLYV<ZPZV<1>, ZPZV<749>, ZPZV<3»; }; // NOLINT
04312 template<> struct ConwayPolynomial<751, 3> { using ZPZ = aerobus::zpz<751>; using type =
             POLYV<ZPZV<1>, ZPZV<0>, ZPZV<5>, ZPZV<748»; }; // NOLINT
04313 template<> struct ConwayPolynomial<751, 4> { using ZPZ = aerobus::zpz<751>; using type =
POLYV<ZPZV<1>, ZPZV<3>, ZPZV<3>, ZPZV<525>, ZPZV<3»; }; // NOLINT
04314 template<> struct ConwayPolynomial<751, 5> { using ZPZ = aerobus::zpz<751>; using type =
             POLYV<ZPZV<1>, ZPZV<0>, ZPZV<0>, ZPZV<0>, ZPZV<7>, ZPZV<748»; }; // NOLINT
04315 template<> struct ConwayPolynomial<751, 6> { using ZPZ = aerobus::zpz<751>; using type =
             POLYV<ZPZV<1>, ZPZV<0>, ZPZV<2>, ZPZV<298>, ZPZV<633>, ZPZV<539>, ZPZV<3»; }; // NOLINT
04316 template<> struct ConwayPolynomial<751, 7> { using ZPZ = aerobus::zpz<751>; using type =
             POLYV<ZPZV<1>, ZPZV<0>, ZPZV<0>, ZPZV<0>, ZPZV<0>, ZPZV<7>, ZPZV<748»; };
04317 template<> struct ConwayPolynomial<751, 8> { using ZPZ = aerobus::zpz<751>; using type =
              POLYV<ZPZV<1>, ZPZV<0>, ZPZV<0>, ZPZV<0>, ZPZV<3>, ZPZV<741>, ZPZV<243>, ZPZV<672>, ZPZV<3»; }; //
04318 template<> struct ConwayPolynomial<751, 9> { using ZPZ = aerobus::zpz<751>; using type =
             POLYV<ZPZV<1>, ZPZV<0>, ZPZV<0>, ZPZV<0>, ZPZV<0>, ZPZV<0>, ZPZV<0>, ZPZV<10>, ZPZV<703>, ZPZV<489>, ZPZV<4489>, ZPZV<489>, ZPZV<489
              }; // NOLINT
04319 template<> struct ConwayPolynomial<757, 1> { using ZPZ = aerobus::zpz<757>; using type =
             POLYV<ZPZV<1>, ZPZV<755»; }; // NOLINT
04320 template<> struct ConwayPolynomial<757, 2> { using ZPZ = aerobus::zpz<757>; using type =
POLYV<ZPZV<1>, ZPZV<753>, ZPZV<2»; }; // NOLINT
04321 template<> struct ConwayPolynomial<757, 3> { using ZPZ = aerobus::zpz<757>; using type =
POLYY<ZPZV<1>, ZPZV<6>, ZPZV<6>, ZPZV<755»; }; // NOLINT

04322 template<> struct ConwayPolynomial<757, 4> { using ZPZ = aerobus::zpz<757>; using type = POLYV<ZPZV<1>, ZPZV<0>, ZPZV<10>, ZPZV<537>, ZPZV<2»; }; // NOLINT

04323 template<> struct ConwayPolynomial<757, 5> { using ZPZ = aerobus::zpz<757>; using type =
             POLYV<ZPZV<1>, ZPZV<0>, ZPZV<0>, ZPZV<0>, ZPZV<13>, ZPZV<755»; }; // NOLINT
04324 template<> struct ConwayPolynomial<757, 6> { using ZPZ = aerobus::zpz<757>; using type =
POLYV<ZPZV<1>, ZPZV<0>, ZPZV<0>, ZPZV<753>, ZPZV<739>, ZPZV<745>, ZPZV<2»; }; // NOLINT 04325 template<> struct ConwayPolynomial<757, 7> { using ZPZ = aerobus::zpz<757>; using type =
POLYV<ZPZV<1>, ZPZV<0>, ZPZV<0>, ZPZV<0>, ZPZV<0>, ZPZV<0>, ZPZV<0>, ZPZV<4>, ZPZV<5, ; ; // N
04326 template<> struct ConwayPolynomial<757, 8> { using ZPZ = aerobus::zpz<757>; using type =
             POLYV<ZPZV<1>, ZPZV<0>, ZPZV<0>, ZPZV<0>, ZPZV<12>, ZPZV<494>, ZPZV<110>, ZPZV<509>, ZPZV<2»; }; //
04327 template<> struct ConwayPolynomial<757, 9> { using ZPZ = aerobus::zpz<757>; using type =
             POLYV<ZPZV<1>, ZPZV<0>, ZPZV<0>, ZPZV<0>, ZPZV<0>, ZPZV<0>, ZPZV<8>, ZPZV<688>, ZPZV<688>, ZPZV<702>, ZPZV<755»;
              }: // NOLINT
04328 template<> struct ConwayPolynomial<761, 1> { using ZPZ = aerobus::zpz<761>; using type =
             POLYV<ZPZV<1>, ZPZV<755»; }; // NOLINT
04329 template<> struct ConwayPolynomial<761, 2> { using ZPZ = aerobus::zpz<761>; using type =
             POLYV<ZPZV<1>, ZPZV<758>, ZPZV<6»; }; // NOLINT
04330 template<> struct ConwayPolynomial<761, 3> { using ZPZ = aerobus::zpz<761>; using type =
             POLYV<ZPZV<1>, ZPZV<0>, ZPZV<12>, ZPZV<755»; }; // NOLINT
04331 template<> struct ConwayPolynomial<761, 4> { using ZPZ = aerobus::zpz<761>; using type =
POLYV<ZPZV<1>, ZPZV<0>, ZPZV<6>, ZPZV<658>, ZPZV<6*, } // NOLINT
04332 template<> struct ConwayPolynomial<761, 5> { using ZPZ = aerobus::zpz<761>; using type =
             POLYV<ZPZV<1>, ZPZV<0>, ZPZV<0>, ZPZV<6>, ZPZV<755»; }; // NOLINT
04333 template<> struct ConwayPolynomial<761, 6> { using ZPZ = aerobus::zpz<761>; using type = POLYV<ZPZV<1>, ZPZV<0>, ZPZV<2>, ZPZV<634>, ZPZV<597>, ZPZV<155>, ZPZV<6»; }; // NOLINT 04334 template<> struct ConwayPolynomial<761, 7> { using ZPZ = aerobus::zpz<761>; using type =
             POLYV<ZPZV<1>, ZPZV<0>, ZPZV<0>, ZPZV<0>, ZPZV<0>, ZPZV<0>, ZPZV<6>, ZPZV<755»; };
04335 template<> struct ConwayPolynomial<761, 8> { using ZPZ = aerobus::zpz<761>; using type =
             POLYV<ZPZV<1>, ZPZV<0>, ZPZV<0>, ZPZV<0>, ZPZV<11>, ZPZV<603>, ZPZV<144>, ZPZV<540>, ZPZV<5%; }; //
             NOLINT
04336 template<> struct ConwayPolynomial<761, 9> { using ZPZ = aerobus::zpz<761>; using type =
             POLYV<ZPZV<1>, ZPZV<0>, ZPZV<0>, ZPZV<0>, ZPZV<0>, ZPZV<0>, ZPZV<4>, ZPZV<317>, ZPZV<571>, ZPZV<755»;
              }; // NOLINT
04337 template<> struct ConwayPolynomial<769, 1> { using ZPZ = aerobus::zpz<769>; using type =
             POLYV<ZPZV<1>, ZPZV<758»; }; // NOLINT
04338 template<> struct ConwayPolynomial<769, 2> { using ZPZ = aerobus::zpz<769>; using type =
POLYY<ZPZV<1>, ZPZV<765>, ZPZV<11»; }; // NOLINT

04339 template<> struct ConwayPolynomial<769, 3> { using ZPZ = aerobus::zpz<769>; using type =
             POLYV<ZPZV<1>, ZPZV<0>, ZPZV<2>, ZPZV<758»; }; // NOLINT
04340 template<> struct ConwayPolynomial<769, 4> { using ZPZ = aerobus::zpz<769>; using type =
             POLYV<ZPZV<1>, ZPZV<0>, ZPZV<32>, ZPZV<741>, ZPZV<11»; }; // NOLINT
04341 template<> struct ConwayPolynomial<769, 5> { using ZPZ = aerobus::zpz<769>; using type =
             POLYV<ZPZV<1>, ZPZV<0>, ZPZV<0>, ZPZV<0>, ZPZV<1>, ZPZV<758»; }; // NOLINT
04342 template<> struct ConwayPolynomial<769, 6> { using ZPZ = aerobus::zpz<769>; using type =
POLYV<ZPZV<1>, ZPZV<0>, ZPZV<4>>, ZPZV<326>, ZPZV<650>, ZPZV<11»; }; // NOLINT 04343 template<> struct ConwayPolynomial<769, 7> { using ZPZ = aerobus::zpz<769>; using type =
             POLYV<ZPZV<1>, ZPZV<0>, ZPZV<0>, ZPZV<0>, ZPZV<0>, ZPZV<0>, ZPZV<0>, ZPZV<5>, ZPZV<5>, ZPZV<5>, ZPZV<5, ZPZV<5
04344 template<> struct ConwayPolynomial<769, 8> { using ZPZ = aerobus::zpz<769>; using type
             POLYV<ZPZV<1>, ZPZV<0>, ZPZV<0>, ZPZV<0>, ZPZV<1>, ZPZV<560>, ZPZV<574>, ZPZV<632>, ZPZV<11»; }; //
             NOLINT
04345 template<> struct ConwayPolynomial<769, 9> { using ZPZ = aerobus::zpz<769>; using type = POLYV<ZPZV<1>, ZPZV<0>, ZPZV<0>, ZPZV<0>, ZPZV<0>, ZPZV<0>, ZPZV<0>, ZPZV<5, ZPZV<5, ZPZV<623>, ZPZV<751>, ZPZV<75
04346 template<> struct ConwayPolynomial<773, 1> { using ZPZ = aerobus::zpz<773>; using type =
             POLYV<ZPZV<1>, ZPZV<771»; }; // NOLINT
04347 template<> struct ConwayPolynomial<773, 2> { using ZPZ = aerobus::zpz<773>; using type =
             POLYV<ZPZV<1>, ZPZV<772>, ZPZV<2»; }; // NOLINT
```

```
04348 template<> struct ConwayPolynomial<773, 3> { using ZPZ = aerobus::zpz<773>; using type =
       POLYV<ZPZV<1>, ZPZV<0>, ZPZV<2>, ZPZV<771»; }; // NOLINT
04349 template<> struct ConwayPolynomial<773, 4> { using ZPZ = aerobus::zpz<773>; using type =
POLYV<ZPZV<1>, ZPZV<0>, ZPZV<4>, ZPZV<444>, ZPZV<2»; }; // NOLINT
04350 template<> struct ConwayPolynomial<773, 5> { using ZPZ = aerobus::zpz<773>; using type =
       POLYV<ZPZV<1>, ZPZV<0>, ZPZV<0>, ZPZV<0>, ZPZV<8>, ZPZV<771»; }; // NOLINT
04351 template<> struct ConwayPolynomial<773, 6> { using ZPZ = aerobus::zpz<773>; using type =
       POLYV<ZPZV<1>, ZPZV<0>, ZPZV<1>, ZPZV<91>, ZPZV<3>, ZPZV<581>, ZPZV<2»; }; // NOLINT
04352 template<> struct ConwayPolynomial<773, 7> { using ZPZ = aerobus::zpz<773>; using type
       POLYV<ZPZV<1>, ZPZV<0>, ZPZV<0>, ZPZV<0>, ZPZV<0>, ZPZV<0>, ZPZV<0>, ZPZV<9>, ZPZV<771»; };
04353 template<> struct ConwayPolynomial<773, 8> { using ZPZ = aerobus::zpz<773>; using type =
       POLYV<ZPZV<1>, ZPZV<0>, ZPZV<0>, ZPZV<0>, ZPZV<0>, ZPZV<484>, ZPZV<94>, ZPZV<693>, ZPZV<2*; }; //
       NOLINT
04354 template<> struct ConwayPolynomial<773, 9> { using ZPZ = aerobus::zpz<773>; using type =
       POLYV<ZPZV<1>, ZPZV<0>, ZPZV<0>, ZPZV<0>, ZPZV<0>, ZPZV<0>, ZPZV<4>, ZPZV<216>, ZPZV<574>, ZPZV<771»;
        }; // NOLINT
04355 template<> struct ConwayPolynomial<787, 1> { using ZPZ = aerobus::zpz<787>; using type =
       POLYV<ZPZV<1>, ZPZV<785»; }; // NOLINT
04356 template<> struct ConwayPolynomial<787, 2> { using ZPZ = aerobus::zpz<787>; using type =
POLYV<ZPZV<1>, ZPZV<786, ZPZV<2»; }; // NOLINT
04357 template<> struct ConwayPolynomial<787, 3> { using ZPZ = aerobus::zpz<787>; using type =
       POLYV<ZPZV<1>, ZPZV<0>, ZPZV<2>, ZPZV<785»; }; // NOLINT
04358 template<> struct ConwayPolynomial<787, 4> { using ZPZ = aerobus::zpz<787>; using type = 04358 template<> struct ConwayPolynomial<787, 4> { using ZPZ = aerobus::zpz<787>; using type = 04358 template<> struct ConwayPolynomial<787, 4> { using ZPZ = aerobus::zpz<787>; using type = 04358 template<> struct ConwayPolynomial<787, 4> { using ZPZ = aerobus::zpz<787>; using type = 04358 template<> struct ConwayPolynomial<787, 4> { using ZPZ = aerobus::zpz<787>; using type = 04358 template</ > struct ConwayPolynomial<787, 4> { using ZPZ = aerobus::zpz<787>; using type = 04358 template</br>
POLYY<ZPZV<1>, ZPZV<0>, ZPZV<11>, ZPZV<605>, ZPZV<2»; }; // NOLINT
04359 template<> struct ConwayPolynomial<787, 5> { using ZPZ = aerobus::zpz<787>; using type =
       POLYV<ZPZV<1>, ZPZV<0>, ZPZV<0>, ZPZV<9>, ZPZV<785»; }; // NOLINT
04360 template<> struct ConwayPolynomial<787, 6> { using ZPZ = aerobus::zpz<787>; using type =
       POLYV<ZPZV<1>, ZPZV<0>, ZPZV<0>, ZPZV<98>, ZPZV<512>, ZPZV<606>, ZPZV<2»; }; // NOLINT
04361 template<> struct ConwayPolynomial<787, 7> { using ZPZ = aerobus::zpz<787>; using type =
       POLYV<ZPZV<1>, ZPZV<0>, ZPZV<0>, ZPZV<0>, ZPZV<0>, ZPZV<0>, ZPZV<0>, ZPZV<3>, ZPZV<785»; };
                                                                                                             // NOLINT
04362 template<> struct ConwayPolynomial<787, 8> { using ZPZ = aerobus::zpz<787>; using type = POLYV<ZPZV<1>, ZPZV<0>, ZPZV<0>, ZPZV<0>, ZPZV<5>, ZPZV<612>, ZPZV<26>, ZPZV<26>, ZPZV<715>, ZPZV<2»; }; //
04363 template<> struct ConwayPolynomial<787, 9> { using ZPZ = aerobus::zpz<787>; using type =
       POLYV<ZPZV<1>, ZPZV<0>, ZPZV<0>, ZPZV<0>, ZPZV<0>, ZPZV<0>, ZPZV<5>, ZPZV<480>, ZPZV<573>, ZPZV<785»;
        }; // NOLINT
04364 template<> struct ConwayPolynomial<797, 1> { using ZPZ = aerobus::zpz<797>; using type =
       POLYV<ZPZV<1>, ZPZV<795»; }; // NOLINT
04365 template<> struct ConwayPolynomial<797, 2> { using ZPZ = aerobus::zpz<797>; using type =
POLYY<ZPZV<1>, ZPZV<793>, ZPZV<2»; }; // NOLINT
04366 template<> struct ConwayPolynomial<797, 3> { using ZPZ = aerobus::zpz<797>; using type =
       POLYV<ZPZV<1>, ZPZV<0>, ZPZV<2>, ZPZV<795»; }; // NOLINT
POLYV<ZPZV<1>, ZPZV<0>, ZPZV<0>, ZPZV<0>, ZPZV<3>, ZPZV<795»; }; // NOLINT
04369 template<> struct ConwayPolynomial<797, 6> { using ZPZ = aerobus::zpz<797>; using type =
POLYV<ZPZV<1>, ZPZV<0>, ZPZV<1>, ZPZV<657>, ZPZV<396>, ZPZV<71>, ZPZV<2»; }; // NOLINT
04370 template<> struct ConwayPolynomial<797, 7> { using ZPZ = aerobus::zpz<797>; using type =
POLYV<ZPZV<1>, ZPZV<0>, ZPZV<0>, ZPZV<0>, ZPZV<0>, ZPZV<1>, ZPZV<795»; }; // NOLINT
04371 template<> struct ConwayPolynomial<797, 8> { using ZPZ = aerobus::zpz<797>; using type =
       POLYV<ZPZV<1>, ZPZV<0>, ZPZV<0>, ZPZV<0>, ZPZV<1>, ZPZV<596>, ZPZV<747>, ZPZV<389>, ZPZV<2»; }; //
       NOLINT
04372 template<> struct ConwayPolynomial<797, 9> { using ZPZ = aerobus::zpz<797>; using type = POLYV<ZPZV<1>, ZPZV<0>, ZPZV<0>, ZPZV<0>, ZPZV<0>, ZPZV<0>, ZPZV<2>, ZPZV<240>, ZPZV<599>, ZPZV<795»;
       }; // NOLINT
04373 template<> struct ConwayPolynomial<809, 1> { using ZPZ = aerobus::zpz<809>; using type =
       POLYV<ZPZV<1>, ZPZV<806»; }; // NOLINT
04374 template<> struct ConwayPolynomial<809, 2> { using ZPZ = aerobus::zpz<809>; using type =
       POLYV<ZPZV<1>, ZPZV<799>, ZPZV<3»; };
                                                      // NOLINT
04375 template<> struct ConwayPolynomial<809, 3> { using ZPZ = aerobus::zpz<809>; using type =
       POLYV<ZPZV<1>, ZPZV<0>, ZPZV<1>, ZPZV<806»; }; // NOLINT
04376 template<> struct ConwayPolynomial<809, 4> { using ZPZ = aerobus::zpz<809>; using type =
POLYV<ZPZV<1>, ZPZV<64, ZPZV<644>, ZPZV<64y; }; // NOLINT
04377 template<> struct ConwayPolynomial<809, 5> { using ZPZ = aerobus::zpz<809>; using type =
       POLYV<ZPZV<1>, ZPZV<0>, ZPZV<0>, ZPZV<0>, ZPZV<3>, ZPZV<806»; }; // NOLINT
04378 template<> struct ConwayPolynomial<809, 6> { using ZPZ = aerobus::zpz<809>; using type = POLYV<ZPZV<1>, ZPZV<0>, ZPZV<1>, ZPZV<562>, ZPZV<75>, ZPZV<43>, ZPZV<3»; }; // NOLINT
04379 template<> struct ConwayPolynomial<809, 7> { using ZPZ = aerobus::zpz<809>; using type
       POLYV<ZPZV<1>, ZPZV<0>, ZPZV<0>, ZPZV<0>, ZPZV<0>, ZPZV<0>, ZPZV<0>, ZPZV<3>, ZPZV<806»; };
04380 template<> struct ConwayPolynomial<809, 8> { using ZPZ = aerobus::zpz<809>; using type
       POLYV<ZPZV<1>, ZPZV<0>, ZPZV<0>, ZPZV<0>, ZPZV<1>, ZPZV<593>, ZPZV<745>, ZPZV<673>, ZPZV<3»; }; //
       NOLINT
04381 template<> struct ConwayPolynomial<809, 9> { using ZPZ = aerobus::zpz<809>; using type =
       POLYV<ZPZV<1>, ZPZV<0>, ZPZV<0>, ZPZV<0>, ZPZV<0>, ZPZV<0>, ZPZV<1>, ZPZV<341>, ZPZV<727>, ZPZV<806»;
       }; // NOLINT
04382 template<> struct ConwayPolynomial<811, 1> { using ZPZ = aerobus::zpz<811>; using type =
       POLYV<ZPZV<1>, ZPZV<808»; }; // NOLINT
04383 template<> struct ConwayPolynomial<811, 2> { using ZPZ = aerobus::zpz<811>; using type =
       POLYV<ZPZV<1>, ZPZV<806>, ZPZV<3»; }; // NOLINT
04384 template<> struct ConwayPolynomial<811, 3> { using ZPZ = aerobus::zpz<811>; using type =
       POLYV<ZPZV<1>, ZPZV<0>, ZPZV<1>, ZPZV<808»; }; // NOLINT
04385 template<> struct ConwayPolynomial<811, 4> { using ZPZ = aerobus::zpz<811>; using type =
       POLYV<ZPZV<1>, ZPZV<0>, ZPZV<3>, ZPZV<453>, ZPZV<3»; }; // NOLINT
04386 template<> struct ConwayPolynomial<811, 5> { using ZPZ = aerobus::zpz<811>; using type =
POLYV<ZPZV<1>, ZPZV<0>, ZPZV<0>, ZPZV<0>, ZPZV<3>, ZPZV<80*, ZPZV<80*, ZPZV<808*; }; // NOLINT 04387 template<> struct ConwayPolynomial<811, 6> { using ZPZ = aerobus::zpz<811>; using type =
```

```
POLYV<ZPZV<1>, ZPZV<0>, ZPZV<780>, ZPZV<755>, ZPZV<307>, ZPZV<3»; };
04388 template<> struct ConwayPolynomial<811, 7> { using ZPZ = aerobus::zpz<811>; using type
                                                                                                                                                          // NOLINT
          POLYV<ZPZV<1>, ZPZV<0>, ZPZV<0>, ZPZV<0>, ZPZV<0>, ZPZV<0>, ZPZV<1>, ZPZV<1>, ZPZV<808»; };
04389 template<> struct ConwayPolynomial<811, 8> { using ZPZ = aerobus::zpz<811>; using type =
          POLYV<ZPZV<1>, ZPZV<0>, ZPZV<0>, ZPZV<0>, ZPZV<3>, ZPZV<663>, ZPZV<806>, ZPZV<805>, ZPZV<525>, ZPZV<3»; }; //
          NOT.TNT
04390 template<> struct ConwayPolynomial<811, 9> { using ZPZ = aerobus::zpz<811>; using type :
          POLYV<ZPZV<1>, ZPZV<0>, ZPZV<0>, ZPZV<0>, ZPZV<0>, ZPZV<0>, ZPZV<10>, ZPZV<382>, ZPZV<200>, ZPZV<808»;
           }; // NOLINT
04391 template<> struct ConwayPolynomial<821, 1> { using ZPZ = aerobus::zpz<821>; using type =
          POLYV<ZPZV<1>, ZPZV<819»; }; // NOLINT
04392 template<> struct ConwayPolynomial<821, 2> { using ZPZ = aerobus::zpz<821>; using type =
          POLYV<ZPZV<1>, ZPZV<816>, ZPZV<2»; };
                                                                            // NOLINT
04393 template<> struct ConwayPolynomial<821, 3> { using ZPZ = aerobus::zpz<821>; using type =
          POLYV<ZPZV<1>, ZPZV<0>, ZPZV<2>, ZPZV<819»; }; // NOLINT
04394 template<> struct ConwayPolynomial<821, 4> { using ZPZ = aerobus::zpz<821>; using type =
POLYV<ZPZV<1>, ZPZV<0>, ZPZV<15>, ZPZV<662>, ZPZV<2»; }; // NOLINT
04395 template<> struct ConwayPolynomial<821, 5> { using ZPZ = aerobus::zpz<821>; using type =
          POLYV<ZPZV<1>, ZPZV<0>, ZPZV<0>, ZPZV<9>, ZPZV<819»; }; // NOLINT
04396 template<> struct ConwayPolynomial<821, 6> { using ZPZ = aerobus::zpz<821>; using type =
          POLYV<ZPZV<1>, ZPZV<0>, ZPZV<2>, ZPZV<160>, ZPZV<130>, ZPZV<803>, ZPZV<2»; }; // NOLINT
04397 template<> struct ConwayPolynomial<821, 7> { using ZPZ = aerobus::zpz<821>; using type =
          POLYV<ZPZV<1>, ZPZV<0>, ZPZV<0>, ZPZV<0>, ZPZV<0>, ZPZV<0>, ZPZV<1>, ZPZV<10>, ZPZV<10
04398 template<> struct ConwayPolynomial<821, 8> { using ZPZ = aerobus::zpz<821>; using type = POLYV<ZPZV<1>, ZPZV<0>, ZPZV<0>, ZPZV<0>, ZPZV<5>, ZPZV<626>, ZPZV<556>, ZPZV<589>, ZPZV<28; }; //
          NOLINT
04399 template<> struct ConwayPolynomial<821, 9> { using ZPZ = aerobus::zpz<821>; using type =
          POLYV<ZPZV<1>, ZPZV<0>, ZPZV<0>, ZPZV<0>, ZPZV<0>, ZPZV<0>, ZPZV<1>, ZPZV<650>, ZPZV<557>, ZPZV<819»;
           }; // NOLINT
04400 template<> struct ConwayPolynomial<823, 1> { using ZPZ = aerobus::zpz<823>; using type =
          POLYV<ZPZV<1>, ZPZV<820»; }; // NOLINT
04401 template<> struct ConwayPolynomial<823, 2> { using ZPZ = aerobus::zpz<823>; using type =
          POLYV<ZPZV<1>, ZPZV<821>, ZPZV<3»; }; // NOLINT
04402 template<> struct ConwayPolynomial<823, 3> { using ZPZ = aerobus::zpz<823>; using type =
          POLYV<ZPZV<1>, ZPZV<0>, ZPZV<3>, ZPZV<820»; }; // NOLINT
04403 template<> struct ConwayPolynomial<823, 4> { using ZPZ = aerobus::zpz<823>; using type =
          POLYV<ZPZV<1>, ZPZV<0>, ZPZV<4>, ZPZV<819>, ZPZV<3»; }; // NOLINT
04404 template<> struct ConwayPolynomial<823, 5> { using ZPZ = aerobus::zpz<823>; using type =
          POLYV<ZPZV<1>, ZPZV<0>, ZPZV<0>, ZPZV<0>, ZPZV<3>, ZPZV<820»; }; // NOLINT
04405 template<> struct ConwayPolynomial<823, 6> { using ZPZ = aerobus::zpz<823>; using type =
          POLYV<ZPZV<1>, ZPZV<0>, ZPZV<0>, ZPZV<822>, ZPZV<616>, ZPZV<744>, ZPZV<3»; }; // NOLINT
04406 template<> struct ConwayPolynomial<823, 7> { using ZPZ = aerobus::zpz<823>; using type =
          POLYV<ZPZV<1>, ZPZV<0>, ZPZV<0>, ZPZV<0>, ZPZV<0>, ZPZV<0>, ZPZV<10>, ZPZV<820»; }; // NOLINT
04407 template<> struct ConwayPolynomial<823, 8> { using ZPZ = aerobus::zpz<823>; using type
          POLYV<ZPZV<1>, ZPZV<0>, ZPZV<0>, ZPZV<0>, ZPZV<0>, ZPZV<451>, ZPZV<437>, ZPZV<31>, ZPZV<3»; }; //
          NOLINT
04408 template<> struct ConwayPolynomial<823, 9> { using ZPZ = aerobus::zpz<823>; using type =
          POLYV<ZPZV<1>, ZPZV<0>, ZPZV<0>, ZPZV<0>, ZPZV<0>, ZPZV<6>, ZPZV<6>, ZPZV<6>, ZPZV<64>, ZPZV<740>, ZPZV<609>, ZPZV<820»;
          }; // NOLINT
04409 template<> struct ConwayPolynomial<827, 1> { using ZPZ = aerobus::zpz<827>; using type =
          POLYV<ZPZV<1>, ZPZV<825»; }; // NOLINT
04410 template<> struct ConwayPolynomial<827, 2> { using ZPZ = aerobus::zpz<827>; using type =
POLYV<ZPZV<1>, ZPZV<821>, ZPZV<2»; }; // NOLINT

04411 template<> struct ConwayPolynomial<827, 3> { using ZPZ = aerobus::zpz<827>; using type =
POLYV<ZPZV<1>, ZPZV<0>, ZPZV<5>, ZPZV<825»; }; // NOLINT
04412 template<> struct ConwayPolynomial<827, 4> { using ZPZ = aerobus::zpz<827>; using type =
          POLYV<ZPZV<1>, ZPZV<0>, ZPZV<18>, ZPZV<605>, ZPZV<2»; }; // NOLINT
04413 template<> struct ConwayPolynomial<827, 5> { using ZPZ = aerobus::zpz<827>; using type =
          POLYV<ZPZV<1>, ZPZV<0>, ZPZV<0>, ZPZV<0>, ZPZV<3>, ZPZV<825»; }; // NOLINT
04414 template<> struct ConwayPolynomial<827, 6> { using ZPZ = aerobus::zpz<827>; using type =
POLYV<ZPZV<1>, ZPZV<0>, ZPZV<9>, ZPZV<685>, ZPZV<601>, ZPZV<691>, ZPZV<2»; }; // NOLINT 04415 template<> struct ConwayPolynomial<827, 7> { using ZPZ = aerobus::zpz<827>; using type =
          POLYV<ZPZV<1>, ZPZV<0>, ZPZV<0>, ZPZV<0>, ZPZV<0>, ZPZV<0>, ZPZV<5>, ZPZV<5>, ZPZV<825»; };
04416 template<> struct ConwayPolynomial<827, 8> { using ZPZ = aerobus::zpz<827>; using type
          POLYV<ZPZV<1>, ZPZV<0>, ZPZV<0>, ZPZV<0>, ZPZV<2>, ZPZV<812>, ZPZV<79>, ZPZV<32>, ZPZV<32>; }; //
          NOLINT
04417 template<> struct ConwayPolynomial<827, 9> { using ZPZ = aerobus::zpz<827>; using type = POLYV<ZPZV<1>, ZPZV<0>, ZPZV<0>, ZPZV<0>, ZPZV<0>, ZPZV<0>, ZPZV<0>, ZPZV<3>, ZPZV<3 , Z
          }; // NOLINT
04418 template<> struct ConwayPolynomial<829, 1> { using ZPZ = aerobus::zpz<829>; using type =
          POLYV<ZPZV<1>, ZPZV<827»; }; // NOLINT
04419 template<> struct ConwayPolynomial<829, 2> { using ZPZ = aerobus::zpz<829>; using type =
POLYV<ZPZV<1>, ZPZV<828>, ZPZV<2»; }; // NOLINT
04420 template<> struct ConwayPolynomial<829, 3> { using ZPZ = aerobus::zpz<829>; using type =
          POLYV<ZPZV<1>, ZPZV<0>, ZPZV<5>, ZPZV<827»; }; // NOLINT
04421 template<> struct ConwayPolynomial<829, 4> { using ZPZ = aerobus::zpz<829>; using type =
POLYV<ZPZV<1>, ZPZV<0>, ZPZV<604>, ZPZV<2»; }; // NOLINT
04422 template<> struct ConwayPolynomial<829, 5> { using ZPZ = aerobus::zpz<829>; using type =
          POLYV<ZPZV<1>, ZPZV<0>, ZPZV<0>, ZPZV<0>, ZPZV<7>, ZPZV<827»; }; // NOLINT
04423 template<> struct ConwayPolynomial<829, 6> { using ZPZ = aerobus::zpz<829>; using type =
          POLYV<ZPZV<1>, ZPZV<0>, ZPZV<1>, ZPZV<341>, ZPZV<476>, ZPZV<817>, ZPZV<2»; }; // NOLINT
04424 template<> struct ConwayPolynomial<829, 7> { using ZPZ = aerobus::zpz<829>; using type
          POLYV<ZPZV<1>, ZPZV<0>, ZPZV<0>, ZPZV<0>, ZPZV<0>, ZPZV<0>, ZPZV<5>, ZPZV<5>, ZPZV<827»; };
04425 template<> struct ConwayPolynomial<829, 8> { using ZPZ = aerobus::zpz<829>; using type
          POLYV<ZPZV<1>, ZPZV<0>, ZPZV<0>, ZPZV<0>, ZPZV<1>, ZPZV<468>, ZPZV<241>, ZPZV<138>, ZPZV<2»; }; //
          NOLINT
04426 template<> struct ConwayPolynomial<829, 9> { using ZPZ = aerobus::zpz<829>; using type =
```

```
POLYV<ZPZV<1>, ZPZV<0>, ZPZV<0>, ZPZV<0>, ZPZV<0>, ZPZV<0>, ZPZV<0>, ZPZV<5>, ZPZV<552>, ZPZV<827»;
04427 template<> struct ConwayPolynomial<839, 1> { using ZPZ = aerobus::zpz<839>; using type =
             POLYV<ZPZV<1>, ZPZV<828»; }; // NOLINT
04428 template<> struct ConwayPolynomial<839, 2> { using ZPZ = aerobus::zpz<839>; using type =
POLYV<ZPZV<1>, ZPZV<838>, ZPZV<11s; }; // NOLINT
04429 template<> struct ConwayPolynomial<839, 3> { using ZPZ = aerobus::zpz<839>; using type =
             POLYV<ZPZV<1>, ZPZV<0>, ZPZV<4>, ZPZV<828»; }; // NOLINT
04430 template<> struct ConwayPolynomial<839, 4> { using ZPZ = aerobus::zpz<839>; using type =
POLYV<ZPZV<1>, ZPZV<0>, ZPZV<2>, ZPZV<609>, ZPZV<11»; }; // NOLINT
04431 template<> struct ConwayPolynomial<839, 5> { using ZPZ = aerobus::zpz<839>; using type =
             POLYV<ZPZV<1>, ZPZV<0>, ZPZV<0>, ZPZV<0>, ZPZV<2>, ZPZV<828»; }; // NOLINT
04432 template<> struct ConwayPolynomial<839, 6> { using ZPZ = aerobus::zpz<839>; using type =
             POLYV<ZPZV<1>, ZPZV<0>, ZPZV<1>, ZPZV<370>, ZPZV<537>, ZPZV<23>, ZPZV<11»; }; // NOLINT
04433 template<> struct ConwayPolynomial<839, 7> { using ZPZ = aerobus::zpz<839>; using type =
             POLYV<ZPZV<1>, ZPZV<0>, ZPZV<0>, ZPZV<0>, ZPZV<0>, ZPZV<0>, ZPZV<0>, ZPZV<5>, ZPZV<5-, ZPZV<5
04434 template<> struct ConwayPolynomial<839, 8> { using ZPZ = aerobus::zpz<839>; using type =
             POLYV<ZPZV<1>, ZPZV<0>, ZPZV<0>, ZPZV<0>, ZPZV<16>, ZPZV<553>, ZPZV<779>, ZPZV<329>, ZPZV<11»; }; //
04435 template<> struct ConwayPolynomial<839, 9> { using ZPZ = aerobus::zpz<839>; using type =
             POLYV<ZPZV<1>, ZPZV<0>, ZPZV<0>, ZPZV<0>, ZPZV<0>, ZPZV<0>, ZPZV<0>, ZPZV<0>, ZPZV<0>, ZPZV<349>, ZPZV<206>, ZPZV<828*;
              }; // NOLINT
04436 template<> struct ConwayPolynomial<853, 1> { using ZPZ = aerobus::zpz<853>; using type =
             POLYV<ZPZV<1>, ZPZV<851»; }; // NOLINT
04437 template<> struct ConwayPolynomial<853, 2> { using ZPZ = aerobus::zpz<853>; using type =
             POLYV<ZPZV<1>, ZPZV<852>, ZPZV<2»; }; // NOLINT
04438 template<> struct ConwayPolynomial<853, 3> { using ZPZ = aerobus::zpz<853>; using type =
             POLYV<ZPZV<1>, ZPZV<0>, ZPZV<4>, ZPZV<851»; }; // NOLINT
04439 template<> struct ConwayPolynomial<853, 4> { using ZPZ = aerobus::zpz<853>; using type =
POLYV<ZPZV<1>, ZPZV<0>, ZPZV<3>, ZPZV<623>, ZPZV<2»; }; // NOLINT
04440 template<> struct ConwayPolynomial<853, 5> { using ZPZ = aerobus::zpz<853>; using type =
             POLYV<ZPZV<1>, ZPZV<0>, ZPZV<0>, ZPZV<0>, ZPZV<2>, ZPZV<851»; // NOLINT
04441 template<> struct ConwayPolynomial<853, 6> { using ZPZ = aerobus::zpz<853>; using type =
             POLYV<ZPZV<1>, ZPZV<0>, ZPZV<0>, ZPZV<276>, ZPZV<194>, ZPZV<512>, ZPZV<2»; }; // NOLINT
04442 template<> struct ConwayPolynomial<853, 7> { using ZPZ = aerobus::zpz<853>; using type =
             POLYV<ZPZV<1>, ZPZV<0>, ZPZV<0>, ZPZV<0>, ZPZV<0>, ZPZV<4>, ZPZV<851»; };
04443 template<> struct ConwayPolynomial<853, 8> { using ZPZ = aerobus::zpz<853>; using type
             POLYV<ZPZV<1>, ZPZV<0>, ZPŽV<0>, ZPŽV<0>, ZPZV<3>, ZPZV<544>, ZPZV<846>, ZPZV<118>, ZPZV<2»; }; //
04444 template<> struct ConwayPolynomial<853, 9> { using ZPZ = aerobus::zpz<853>; using type =
             POLYV<ZPZV<1>, ZPZV<0>, ZPZV<0>, ZPZV<0>, ZPZV<0>, ZPZV<0>, ZPZV<0>, ZPZV<11>, ZPZV<677>, ZPZV<821>, ZPZV<851»;
              }; // NOLINT
04445 template<> struct ConwayPolynomial<857, 1> { using ZPZ = aerobus::zpz<857>; using type =
             POLYV<ZPZV<1>, ZPZV<854»; }; // NOLINT
04446 template<> struct ConwayPolynomial<857, 2> { using ZPZ = aerobus::zpz<857>; using type =
             POLYV<ZPZV<1>, ZPZV<850>, ZPZV<3»; };
                                                                                               // NOLINT
04447 template<> struct ConwayPolynomial<857, 3> { using ZPZ = aerobus::zpz<857>; using type =
POLYV<ZPZV<1>, ZPZV<0>, ZPZV<4>, ZPZV<854»; }; // NOLINT
04448 template<> struct ConwayPolynomial<857, 4> { using ZPZ = aerobus::zpz<857>; using type =
POLYY<ZPZV<1>, ZPZV<0>, ZPZV<528>, ZPZV<3%; }; // NOLINT
04449 template<> struct ConwayPolynomial<857, 5> { using ZPZ = aerobus::zpz<857>; using type =
             POLYV<ZPZV<1>, ZPZV<0>, ZPZV<0>, ZPZV<0>, ZPZV<3>, ZPZV<854»; }; // NOLINT
04450 template<> struct ConwayPolynomial<857, 6> { using ZPZ = aerobus::zpz<857>; using type =
POLYV<ZPZV<1>, ZPZV<0>, ZPZV<1>, ZPZV<32>, ZPZV<824>, ZPZV<65>, ZPZV<85, }; // NOLINT 04451 template<> struct ConwayPolynomial<857, 7> { using ZPZ = aerobus::zpz<857>; using type
POLYY<ZPZV<1>, ZPZV<0>, ZPZV<0>, ZPZV<0>, ZPZV<0>, ZPZV<0>, ZPZV<0>, ZPZV<5, Z
                                                                                                                                                                                                // NOLINT
             POLYV<ZPZV<1>, ZPZV<0>, ZPZV<0>, ZPZV<0>, ZPZV<3>, ZPZV<611>, ZPZV<552>, ZPZV<494>, ZPZV<3»; }; //
             NOLINT
04453 template<> struct ConwayPolynomial<857, 9> { using ZPZ = aerobus::zpz<857>; using type = POLYV<ZPZV<1>, ZPZV<0>, ZPZV<0>, ZPZV<0>, ZPZV<0>, ZPZV<0>, ZPZV<0>, ZPZV<30>, ZPZV<308>, ZPZV<719>, ZPZV<854»;
             }; // NOLINT
04454 template<> struct ConwayPolynomial<859, 1> { using ZPZ = aerobus::zpz<859>; using type =
             POLYV<ZPZV<1>, ZPZV<857»; }; // NOLINT
04455 template<> struct ConwayPolynomial<859, 2> { using ZPZ = aerobus::zpz<859>; using type =
             POLYV<ZPZV<1>, ZPZV<858>, ZPZV<2»; }; // NOLINT
04456 template<> struct ConwayPolynomial<859, 3> { using ZPZ = aerobus::zpz<859>; using type =
             POLYV<ZPZV<1>, ZPZV<0>, ZPZV<6>, ZPZV<857»; }; // NOLINT
04457 template<> struct ConwayPolynomial<859, 4> { using ZPZ = aerobus::zpz<859>; using type =
             POLYV<ZPZV<1>, ZPZV<0>, ZPZV<2>, ZPZV<530>, ZPZV<2»; }; // NOLINT
04458 template<> struct ConwayPolynomial<859, 5> { using ZPZ = aerobus::zpz<859>; using type =
              \verb"POLYV<ZPZV<1>, \verb"ZPZV<0>, \verb"ZPZV<0>, \verb"ZPZV<1>, \verb"ZPZV<857"; \verb"}; $ // \verb"NOLINT" | NOLINT" 
04459 template<> struct ConwayPolynomial<859, 6> { using ZPZ = aerobus::zpz<859>; using type = POLYV<ZPZV<1>, ZPZV<0>, ZPZV<0>, ZPZV<419>, ZPZV<646>, ZPZV<566>, ZPZV<2»; }; // NOLINT 04460 template<> struct ConwayPolynomial<859, 7> { using ZPZ = aerobus::zpz<859>; using type =
             POLYV<ZPZV<1>, ZPZV<0>, ZPZV<0>, ZPZV<0>, ZPZV<0>, ZPZV<0>, ZPZV<0>, ZPZV<2>, ZPZV<857»; }; // NOLINT
04461 template<> struct ConwayPolynomial<859, 8> { using ZPZ = aerobus::zpz<859>; using type =
             POLYV<ZPZV<1>, ZPZV<0>, ZPZV<0>, ZPZV<0>, ZPZV<0>, ZPZV<52>, ZPZV<446>, ZPZV<672>, ZPZV<672>; }; //
             NOLTNT
04462 template<> struct ConwayPolynomial<859, 9> { using ZPZ = aerobus::zpz<859>; using type =
             POLYV<ZPZV<1>, ZPZV<0>, ZPZV<0>, ZPZV<0>, ZPZV<0>, ZPZV<0>, ZPZV<0>, ZPZV<7>, ZPZV<648>, ZPZV<845>, ZPZV<857»;
             }; // NOLINT
04463 template<> struct ConwayPolynomial<863, 1> { using ZPZ = aerobus::zpz<863>; using type =
             POLYV<ZPZV<1>, ZPZV<858»; }; // NOLINT
04464 template<> struct ConwayPolynomial<863, 2> { using ZPZ = aerobus::zpz<863>; using type =
POLYV<ZPZV<1>, ZPZV<862>, ZPZV<5»; }; // NOLINT
04465 template<> struct ConwayPolynomial<863, 3> { using ZPZ = aerobus::zpz<863>; using type =
```

```
POLYV<ZPZV<1>, ZPZV<0>, ZPZV<5>, ZPZV<858»; };
04466 template<> struct ConwayPolynomial<br/>
863, 4> { using ZPZ = aerobus::zpz<863>; using type = POLYV<ZPZV<1>, ZPZV<0>, ZPZV<2>, ZPZV<770>, ZPZV<5>; }; // NOLINT<br/>
04467 template<> struct ConwayPolynomial<br/>
863, 5> { using ZPZ = aerobus::zpz<863>; using type =
POLYV<ZPZV<1>, ZPZV<0>, ZPZV<0>, ZPZV<0>, ZPZV<10>, ZPZV<50>, ZPZV
           POLYV<ZPZV<1>, ZPZV<0>, ZPZV<1>, ZPZV<330>, ZPZV<62>, ZPZV<300>, ZPZV<5»; }; // NOLINT
04469 template<> struct ConwayPolynomial<863, 7> { using ZPZ = aerobus::zpz<863>;
           POLYV<ZPZV<1>, ZPZV<0>, ZPZV<0>, ZPZV<0>, ZPZV<0>, ZPZV<0>, ZPZV<0>, ZPZV<1>, ZPZV<5858»; };
04470 template<> struct ConwayPolynomial<863, 8> { using ZPZ = aerobus::zpz<863>; using type =
           POLYV<ZPZV<1>, ZPZV<0>, ZPZV<0>, ZPZV<0>, ZPZV<9>, ZPZV<765>, ZPZV<576>, ZPZV<849>, ZPZV<849>, ZPZV<576>, ZPZV<576-, ZPZV
           NOLINT
04471 template<> struct ConwayPolynomial<863, 9> { using ZPZ = aerobus::zpz<863>; using type =
           POLYV<ZPZV<1>, ZPZV<0>, ZPZV<0>, ZPZV<0>, ZPZV<0>, ZPZV<0>, ZPZV<1>, ZPZV<381>, ZPZV<31>, ZPZV<41>, ZPZV<858»; };
            // NOLINT
04472 template<> struct ConwayPolynomial<877, 1> { using ZPZ = aerobus::zpz<877>; using type =
           POLYV<ZPZV<1>, ZPZV<875»; }; // NOLINT
04473 template<> struct ConwayPolynomial<877, 2> { using ZPZ = aerobus::zpz<877>; using type =
           POLYV<ZPZV<1>, ZPZV<873>, ZPZV<2»; }; // NOLINT
04474 template<> struct ConwayPolynomial<877, 3> { using ZPZ = aerobus::zpz<877>; using type =
           POLYV<ZPZV<1>, ZPZV<0>, ZPZV<5>, ZPZV<875»; }; // NOLINT
04475 template<> struct ConwayPolynomial<877, 4> { using ZPZ = aerobus::zpz<877>; using type =
POLYV<ZPZV<1>, ZPZV<6>, ZPZV<60, ZPZV<604>, ZPZV<2»; }; // NOLINT
04476 template<> struct ConwayPolynomial<877, 5> { using ZPZ = aerobus::zpz<877>; using type =
           POLYV<ZPZV<1>, ZPZV<0>, ZPZV<0>, ZPZV<6>, ZPZV<6>, ZPZV<875»; }; // NOLINT
04477 template<> struct ConwayPolynomial<877, 6> { using ZPZ = aerobus::zpz<877>; using type =
           POLYV<ZPZV<1>, ZPZV<0>, ZPZV<0>, ZPZV<629>, ZPZV<400>, ZPZV<855>, ZPZV<2»; }; // NOLINT
04478 template<> struct ConwayPolynomial<877, 7> { using ZPZ = aerobus::zpz<877>; using type =
           POLYV<ZPZV<1>, ZPZV<0>, ZPZV<0>, ZPZV<0>, ZPZV<0>, ZPZV<0>, ZPZV<3>, ZPZV<3>, ZPZV<875»; };
04479 template<> struct ConwayPolynomial<877, 8> { using ZPZ = aerobus::zpz<877>; using type =
           POLYV<ZPZV<1>, ZPZV<0>, ZPZV<0>, ZPZV<0>, ZPZV<4>, ZPZV<767>, ZPZV<319>, ZPZV<347>, ZPZV<2*; }; //
           NOLINT
04480 template<> struct ConwayPolynomial<877, 9> { using ZPZ = aerobus::zpz<877>; using type =
           POLYV<ZPZV<1>, ZPZV<0>, ZPZV<0>, ZPZV<0>, ZPZV<0>, ZPZV<0>, ZPZV<770>, ZPZV<278>, ZPZV<875»;
            }; // NOLINT
04481 template<> struct ConwayPolynomial<881, 1> { using ZPZ = aerobus::zpz<881>; using type =
           POLYV<ZPZV<1>, ZPZV<878»; }; // NOLINT
04482 template<> struct ConwayPolynomial<881, 2> { using ZPZ = aerobus::zpz<881>; using type =
           POLYV<ZPZV<1>, ZPZV<869>, ZPZV<3»; }; // NOLINT
04483 template<> struct ConwayPolynomial<881, 3> { using ZPZ = aerobus::zpz<881>; using type =
           POLYV<ZPZV<1>, ZPZV<0>, ZPZV<1>, ZPZV<878»; }; // NOLINT
04484 template<> struct ConwayPolynomial<881, 4> { using ZPZ = aerobus::zpz<881>; using type =
POLYV<ZPZV<1>, ZPZV<0>, ZPZV<0>, ZPZV<447>, ZPZV<3»; }; // NOLINT
04485 template<> struct ConwayPolynomial<881, 5> { using ZPZ = aerobus::zpz<881>; using type =
           POLYV<ZPZV<1>, ZPZV<0>, ZPZV<0>, ZPZV<0>, ZPZV<8>, ZPZV<878»; }; // NOLINT
04486 template<> struct ConwayPolynomial<881, 6> { using ZPZ = aerobus::zpz<881>; using type =
           POLYV<ZPZV<1>, ZPZV<0>, ZPZV<1>, ZPZV<218>, ZPZV<419>, ZPZV<231>, ZPZV<3»; }; // NOLINT
04487 template<> struct ConwayPolynomial<881, 7> { using ZPZ = aerobus::zpz<881>; using type :
           POLYV<ZPZV<1>, ZPZV<0>, ZPZV<0>, ZPZV<0>, ZPZV<0>, ZPZV<0>, ZPZV<0>, ZPZV<6>, ZPZV<6>, ZPZV<878»; };
                                                                                                                                                                      // NOLINT
04488 template<> struct ConwayPolynomial<881, 8> { using ZPZ = aerobus::zpz<881>; using type =
           POLYV<ZPZV<1>, ZPZV<0>, ZPZV<0>, ZPZV<0>, ZPZV<21>, ZPZV<635>, ZPZV<490>, ZPZV<561>, ZPZV<3»; }; //
04489 template<> struct ConwayPolynomial<881, 9> { using ZPZ = aerobus::zpz<881>; using type =
           POLYV<ZPZV<1>, ZPZV<0>, ZPZV<0>, ZPZV<0>, ZPZV<0>, ZPZV<0>, ZPZV<2>, ZPZV<587>, ZPZV<510>, ZPZV<878»;
           }; // NOLINT
04490 template<> struct ConwayPolynomial<883, 1> { using ZPZ = aerobus::zpz<883>; using type =
           POLYV<ZPZV<1>, ZPZV<881»; }; // NOLINT
04491 template<> struct ConwayPolynomial<883, 2> { using ZPZ = aerobus::zpz<883>; using type =
           POLYV<ZPZV<1>, ZPZV<879>, ZPZV<2»; }; // NOLINT
04492 template<> struct ConwayPolynomial<883, 3> { using ZPZ = aerobus::zpz<883>; using type =
POLYV<ZPZV<1>, ZPZV<0>, ZPZV<6>, ZPZV<881»; }; // NOLINT
04493 template<> struct ConwayPolynomial<883, 4> { using ZPZ = aerobus::zpz<883>; using type =
POLYV<ZPZV<1>, ZPZV<8>, ZPZV<8>, ZPZV<715>, ZPZV<2»; }; // NOLINT
04494 template<> struct ConwayPolynomial<883, 5> { using ZPZ = aerobus::zpz<883>; using type =
           POLYV<ZPZV<1>, ZPZV<0>, ZPZV<0>, ZPZV<0>, ZPZV<14>, ZPZV<881»; }; // NOLINT
04495 template<> struct ConwayPolynomial<883, 6> { using ZPZ = aerobus::zpz<883>; using type =
POLYV<ZPZV<1>, ZPZV<0>, ZPZV<0>, ZPZV<879>, ZPZV×865>, ZPZV<871>, ZPZV<2»; }; // NOLINT
04496 template<> struct ConwayPolynomial<883, 7> { using ZPZ = aerobus::zpz<883>; using type =
POLYV<ZPZV<1>, ZPZV<0>, ZPZV<0>, ZPZV<0>, ZPZV<0>, ZPZV<0>, ZPZV<6>, ZPZV<6>, ZPZV<68+; }; // NOLINT
04497 template<> struct ConwayPolynomial<883, 8> { using ZPZ = aerobus::zpz<883>; using type
           POLYV<ZPZV<1>, ZPZV<0>, ZPZV<0>, ZPZV<0>, ZPZV<7>, ZPZV<740>, ZPZV<762>, ZPZV<768>, ZPZV<72»; }; //
           NOLINT
04498 template<> struct ConwayPolynomial<883, 9> { using ZPZ = aerobus::zpz<883>; using type = POLYV<ZPZV<1>, ZPZV<0>, ZPZV<0>, ZPZV<0>, ZPZV<0>, ZPZV<0>, ZPZV<5>, ZPZV<360>, ZPZV<360>, ZPZV<557>, ZPZV<881»;
           }; // NOLINT
04499 template<> struct ConwayPolynomial<887, 1> { using ZPZ = aerobus::zpz<887>; using type =
           POLYV<ZPZV<1>, ZPZV<882»; }; // NOLINT
04500 template<> struct ConwayPolynomial<887, 2> { using ZPZ = aerobus::zpz<887>; using type =
POLYV<ZPZV<1>, ZPZV<885>, ZPZV<5»; }; // NOLINT
04501 template<> struct ConwayPolynomial<887, 3> { using ZPZ = aerobus::zpz<887>; using type =
           POLYV<ZPZV<1>, ZPZV<0>, ZPZV<1>, ZPZV<882»; }; // NOLINT
04502 template<> struct ConwayPolynomial<887, 4> { using ZPZ = aerobus::zpz<887>; using type =
POLYV<ZPZV<1>, ZPZV<0>, ZPZV<883>, ZPZV<883>, ZPZV<5»; }; // NOLINT
04503 template<> struct ConwayPolynomial<887, 5> { using ZPZ = aerobus::zpz<887>; using type =
           POLYV<ZPZV<1>, ZPZV<0>, ZPZV<0>, ZPZV<5>, ZPZV<882»; }; // NOLINT
04504 template<> struct ConwayPolynomial<887, 6> { using ZPZ = aerobus::zpz<887>; using type = POLYV<ZPZV<1>, ZPZV<0>, ZPZV<1>, ZPZV<775>, ZPZV<341>, ZPZV<28>, ZPZV<5»; }; // NOLINT
```

```
04505 template<> struct ConwayPolynomial<887, 7> { using ZPZ = aerobus::zpz<887>; using type
         POLYV<ZPZV<1>, ZPZV<0>, ZPZV<0>, ZPZV<0>, ZPZV<0>, ZPZV<0>, ZPZV<8>, ZPZV<82»; };
04506 template<> struct ConwayPolynomial<887, 8> { using ZPZ = aerobus::zpz<887>; using type =
         POLYV<ZPZV<1>, ZPZV<0>, ZPZV<0>, ZPZV<0>, ZPZV<0>, ZPZV<781>, ZPZV<381>, ZPZV<706>, ZPZV<5»; }; //
         NOLINT
04507 template<> struct ConwayPolynomial<887, 9> { using ZPZ = aerobus::zpz<887>; using type =
         POLYV<ZPZV<1>, ZPZV<0>, ZPZV<0>, ZPZV<0>, ZPZV<0>, ZPZV<0>, ZPZV<4>, ZPZV<727>, ZPZV<345>, ZPZV<882»;
         }; // NOLINT
04508 template<> struct ConwayPolynomial<907, 1> { using ZPZ = aerobus::zpz<907>; using type =
         POLYV<ZPZV<1>, ZPZV<905»; }; // NOLINT
04509 template<> struct ConwayPolynomial<907, 2> { using ZPZ = aerobus::zpz<907>; using type =
POLYV<ZPZV<1>, ZPZV<903>, ZPZV<2»; }; // NOLINT
04510 template<> struct ConwayPolynomial<907, 3> { using ZPZ = aerobus::zpz<907>; using type =
         POLYV<ZPZV<1>, ZPZV<0>, ZPZV<4>, ZPZV<905»; }; // NOLINT
04511 template<> struct ConwayPolynomial<907, 4> { using ZPZ = aerobus::zpz<907>; using type =
POLYV<ZPZV<1>, ZPZV<0>, ZPZV<14>, ZPZV<478>, ZPZV<2»; }; // NOLINT
04512 template<> struct ConwayPolynomial<907, 5> { using ZPZ = aerobus::zpz<907>; using type =
         POLYV<ZPZV<1>, ZPZV<0>, ZPZV<0>, ZPZV<0>, ZPZV<2>, ZPZV<905»; }; // NOLINT
04513 template<> struct ConwayPolynomial<907, 6> { using ZPZ = aerobus::zpz<907>; using type =
         POLYV<ZPZV<1>, ZPZV<0>, ZPZV<0>, ZPZV<626>, ZPZV<752>, ZPZV<266>, ZPZV<2»; }; // NOLINT
04514 template<> struct ConwayPolynomial<907, 7> { using ZPZ = aerobus::zpz<907>;
         POLYV<ZPZV<1>, ZPZV<0>, ZPZV<0>, ZPZV<0>, ZPZV<0>, ZPZV<0>, ZPZV<0>, ZPZV<0>, ZPZV<2>, ZPZV<905»; };
04515 template<> struct ConwayPolynomial<907, 8> { using ZPZ = aerobus::zpz<907>; using type =
         POLYV<ZPZV<1>, ZPZV<0>, ZPZV<0>, ZPZV<0>, ZPZV<4>, ZPZV<584>, ZPZV<518>, ZPZV<811>, ZPZV<81; }; //
         NOLINT
04516 template<> struct ConwayPolynomial<907, 9> { using ZPZ = aerobus::zpz<907>; using type =
         POLYV<ZPZV<1>, ZPZV<0>, ZPZV<0>, ZPZV<0>, ZPZV<0>, ZPZV<0>, ZPZV<0>, ZPZV<5>, ZPZV<5-, ZPZV<5
          }; // NOLINT
04517 template<> struct ConwayPolynomial<911, 1> { using ZPZ = aerobus::zpz<911>; using type =
         POLYV<ZPZV<1>, ZPZV<894»; }; // NOLINT
04518 template<> struct ConwayPolynomial<911, 2> { using ZPZ = aerobus::zpz<911>; using type =
POLYV<ZPZV<1>, ZPZV<909>, ZPZV<17s; }; // NOLINT
04519 template<> struct ConwayPolynomial<911, 3> { using ZPZ = aerobus::zpz<911>; using type =
         POLYV<ZPZV<1>, ZPZV<0>, ZPZV<1>, ZPZV<894»; }; // NOLINT
04520 template<> struct ConwayPolynomial<911, 4> { using ZPZ = aerobus::zpz<911>; using type =
POLYV<ZPZV<1>, ZPZV<0>, ZPZV<11>, ZPZV<887>, ZPZV<17»; }; // NOLINT
04521 template<> struct ConwayPolynomial<911, 5> { using ZPZ = aerobus::zpz<911>; using type =
         POLYV<ZPZV<1>, ZPZV<0>, ZPZV<0>, ZPZV<2>, ZPZV<894»; }; // NOLINT
04522 template<> struct ConwayPolynomial<911, 6> { using ZPZ = aerobus::zpz<911>; using type
         POLYV<ZPZV<1>, ZPZV<0>, ZPZV<1>, ZPZV<172>, ZPZV<683>, ZPZV<19>, ZPZV<17»; }; // NOLINT
04523 template<> struct ConwayPolynomial<911, 7> { using ZPZ = aerobus::zpz<911>; using type =
         POLYV<ZPZV<1>, ZPZV<0>, ZPZV<0>, ZPZV<0>, ZPZV<0>, ZPZV<0>, ZPZV<4>, ZPZV<894»; };
04524 template<> struct ConwayPolynomial<911, 8> { using ZPZ = aerobus::zpz<911>; using type = POLYV<ZPZV<1>, ZPZV<0>, ZPZV<0>, ZPZV<0>, ZPZV<6>, ZPZV<708>, ZPZV<590>, ZPZV<168>, ZPZV<168>, ZPZV<17»; }; //
04525 template<> struct ConwayPolynomial<911, 9> { using ZPZ = aerobus::zpz<911>; using type =
         POLYV<ZPZV<1>, ZPZV<0>, ZPZV<0>, ZPZV<0>, ZPZV<0>, ZPZV<0>, ZPZV<0>, ZPZV<679>, ZPZV<116>, ZPZV<894»;
         }; // NOLINT
04526 template<> struct ConwayPolynomial<919, 1> { using ZPZ = aerobus::zpz<919>; using type =
         POLYV<ZPZV<1>, ZPZV<912»; }; // NOLINT
04527 template<> struct ConwayPolynomial<919, 2> { using ZPZ = aerobus::zpz<919>; using type =
         POLYV<ZPZV<1>, ZPZV<910>, ZPZV<7»; }; // NOLINT
04528 template<> struct ConwayPolynomial<919, 3> { using ZPZ = aerobus::zpz<919>; using type =
POLYY<ZPZV<1>, ZPZV<0>, ZPZV<2>, ZPZV<912»; }; // NOLINT

04529 template<> struct ConwayPolynomial<919, 4> { using ZPZ = aerobus::zpz<919>; using type = POLYV<ZPZV<1>, ZPZV<0>, ZPZV<3>, ZPZV<602>, ZPZV<7»; }; // NOLINT

04530 template<> struct ConwayPolynomial<919, 5> { using ZPZ = aerobus::zpz<919>; using type =
         POLYV<ZPZV<1>, ZPZV<0>, ZPZV<0>, ZPZV<0>, ZPZV<11>, ZPZV<912»; }; // NOLINT
04531 template<> struct ConwayPolynomial<919, 6> { using ZPZ = aerobus::zpz<919>; using type =
POLYV<ZPZV<1>, ZPZV<0>, ZPZV<0>, ZPZV<312>, ZPZV<817>, ZPZV<113>, ZPZV<7»; }; // NOLINT 04532 template<> struct ConwayPolynomial<919, 7> { using ZPZ = aerobus::zpz<919>; using type =
         POLYV<ZPZV<1>, ZPZV<0>, ZPZV<0>, ZPZV<0>, ZPZV<0>, ZPZV<0>, ZPZV<9>, ZPZV<9>, ZPZV<9), ZPZV<9
04533 template<> struct ConwayPolynomial<919, 8> { using ZPZ = aerobus::zpz<919>; using type =
         POLYV<ZPZV<1>, ZPZV<0>, ZPZV<0>, ZPZV<0>, ZPZV<0>, ZPZV<6>, ZPZV<708>, ZPZV<202>, ZPZV<504>, ZPZV<70*; }; //
         NOT.TNT
04534 template<> struct ConwayPolynomial<919, 9> { using ZPZ = aerobus::zpz<919>; using type =
         POLYV<ZPZV<1>, ZPZV<0>, ZPZV<0>, ZPZV<0>, ZPZV<0>, ZPZV<0>, ZPZV<7>, ZPZV<410>, ZPZV<623>, ZPZV<912»;
         }; // NOLINT
04535 template<> struct ConwayPolynomial<929, 1> { using ZPZ = aerobus::zpz<929>; using type =
         POLYV<ZPZV<1>, ZPZV<926»; }; // NOLINT
04536 template<> struct ConwayPolynomial<929, 2> { using ZPZ = aerobus::zpz<929>; using type =
         POLYV<ZPZV<1>, ZPZV<917>, ZPZV<3»; }; // NOLINT
04537 template<> struct ConwayPolynomial<929, 3> { using ZPZ = aerobus::zpz<929>; using type =
         POLYV<ZPZV<1>, ZPZV<0>, ZPZV<5>, ZPZV<926»; }; // NOLINT
04538 template<> struct ConwayPolynomial<929, 4> { using ZPZ = aerobus::zpz<929>; using type = POLYV<ZPZV<1>, ZPZV<0>, ZPZV<0>, ZPZV<787>, ZPZV<3»; }; // NOLINT
04539 template<> struct ConwayPolynomial<929, 5> { using ZPZ = aerobus::zpz<929>; using type =
         POLYV<ZPZV<1>, ZPZV<0>, ZPZV<0>, ZPZV<0>, ZPZV<3>, ZPZV<926»; }; // NOLINT
04540 template<> struct ConwayPolynomial<929, 6> { using ZPZ = aerobus::zpz<929>; using type =
         POLYV<ZPZV<1>, ZPZV<0>, ZPZV<2>, ZPZV<805>, ZPZV<92>, ZPZV<86>, ZPZV<3»; }; // NOLINT
04541 template<> struct ConwayPolynomial<929, 7> { using ZPZ = aerobus::zpz<929; using type PoLYV<ZPZV<1>, ZPZV<0>, ZPZV<0>, ZPZV<0>, ZPZV<0>, ZPZV<0>, ZPZV<5, ZPZV<9; }; //
04542 template<> struct ConwayPolynomial<929, 8> { using ZPZ = aerobus::zpz<929>; using type
         POLYV<ZPZV<1>, ZPZV<0>, ZPZV<0>, ZPZV<0>, ZPZV<5>, ZPZV<699>, ZPZV<292>, ZPZV<586>, ZPZV<3»; }; //
04543 template<> struct ConwayPolynomial<929, 9> { using ZPZ = aerobus::zpz<929>; using type = POLYV<ZPZV<1>, ZPZV<0>, ZPZV<0>, ZPZV<0>, ZPZV<0>, ZPZV<0>, ZPZV<481>, ZPZV<199>, ZPZV<926»;
```

```
}; // NOLINT
04544 template<> struct ConwayPolynomial<937, 1> { using ZPZ = aerobus::zpz<937>; using type =
       POLYV<ZPZV<1>, ZPZV<932»; }; // NOLINT
04545 template<> struct ConwayPolynomial<937, 2> { using ZPZ = aerobus::zpz<937>; using type =
POLYV<ZPZV<1>, ZPZV<934>, ZPZV<5»; }; // NOLINT
04546 template<> struct ConwayPolynomial<937, 3> { using ZPZ = aerobus::zpz<937>; using type =
       POLYV<ZPZV<1>, ZPZV<0>, ZPZV<3>, ZPZV<932»; }; // NOLINT
04547 template<> struct ConwayPolynomial<937, 4> { using ZPZ = aerobus::zpz<937>; using type =
POLYV<ZPZV<1>, ZPZV<0>, ZPZV<23>, ZPZV<585>, ZPZV<5%; }; // NOLINT
04548 template<> struct ConwayPolynomial<937, 5> { using ZPZ = aerobus::zpz<937>; using type =
       POLYV<ZPZV<1>, ZPZV<0>, ZPZV<0>, ZPZV<0>, ZPZV<5>, ZPZV<932»; }; // NOLINT
04549 template<> struct ConwayPolynomial<937, 6> { using ZPZ = aerobus::zpz<937>; using type =
POLYV<ZPZV<1>, ZPZV<0>, ZPZV<794>, ZPZV<727>, ZPZV<5>; ); // NOLINT 04550 template<> struct ConwayPolynomial<937, 7> { using ZPZ = aerobus::zpz<937>; using type =
       POLYV<ZPZV<1>, ZPZV<0>, ZPZV<0>, ZPZV<0>, ZPZV<0>, ZPZV<0>, ZPZV<0>, ZPZV<24>, ZPZV<932»; };
04551 template<> struct ConwayPolynomial<937, 8> { using ZPZ = aerobus::zpz<937>; using type =
       POLYV<ZPZV<1>, ZPZV<0>, ZPZV<0>, ZPZV<0>, ZPZV<6>8>, ZPZV<26>, ZPZV<26>, ZPZV<53>, ZPZV<5»; }; //
       NOLINT
04552 template<> struct ConwayPolynomial<937, 9> { using ZPZ = aerobus::zpz<937>; using type =
       POLYV<ZPZV<1>, ZPZV<0>, ZPZV<0>, ZPZV<0>, ZPZV<0>, ZPZV<0>, ZPZV<0>, ZPZV<28>, ZPZV<533>, ZPZV<483>, ZPZV<932»;
       }; // NOLINT
04553 template<> struct ConwayPolynomial<941, 1> { using ZPZ = aerobus::zpz<941>; using type =
       POLYV<ZPZV<1>, ZPZV<939»; }; // NOLINT
04554 template<> struct ConwayPolynomial<941, 2> { using ZPZ = aerobus::zpz<941>; using type =
       POLYV<ZPZV<1>, ZPZV<940>, ZPZV<2»; }; // NOLINT
04555 template<> struct ConwayPolynomial<941, 3> { using ZPZ = aerobus::zpz<941>; using type =
       POLYV<ZPZV<1>, ZPZV<0>, ZPZV<3>, ZPZV<939»; }; // NOLINT
04556 template<> struct ConwayPolynomial<941, 4> { using ZPZ = aerobus::zpz<941>; using type =
POLYV<ZPZV<1>, ZPZV<0>, ZPZV<3>, ZPZV<505>, ZPZV<2»; }; // NOLINT

04557 template<> struct ConwayPolynomial<941, 5> { using ZPZ = aerobus::zpz<941>; using type = POLYV<ZPZV<1>, ZPZV<0>, ZPZV<0>, ZPZV<0>, ZPZV<2>, ZPZV<939»; }; // NOLINT
04558 template<> struct ConwayPolynomial<941, 6> { using ZPZ = aerobus::zpz<941>; using type
       POLYV<ZPZV<1>, ZPZV<0>, ZPZV<2>, ZPZV<459>, ZPZV<694>, ZPZV<538>, ZPZV<2»; }; // NOLINT
04559 template<> struct ConwayPolynomial<941, 7> { using ZPZ = aerobus::zpz<941>; using type :
       POLYV<ZPZV<1>, ZPZV<0>, ZPZV<0>, ZPZV<0>, ZPZV<0>, ZPZV<0>, ZPZV<0>, ZPZV<4>, ZPZV<4>, ZPZV<939»; };
04560 template<> struct ConwayPolynomial<941, 8> { using ZPZ = aerobus::zpz<941>; using type =
       POLYV<ZPZV<1>, ZPZV<0>, ZPZV<0>, ZPZV<0>, ZPZV<0>, ZPZV<80>, ZPZV<805>, ZPZV<675>, ZPZV<590>, ZPZV<2»; }; //
04561 template<> struct ConwayPolynomial<941, 9> { using ZPZ = aerobus::zpz<941>; using type
       POLYV<ZPZV<1>, ZPZV<0>, ZPZV<0>, ZPZV<0>, ZPZV<0>, ZPZV<0>, ZPZV<1>, ZPZV<708>, ZPZV<197>, ZPZV<19939»;
       }; // NOLINT
04562 template<> struct ConwayPolynomial<947, 1> { using ZPZ = aerobus::zpz<947>; using type =
       POLYV<ZPZV<1>, ZPZV<945»; }; // NOLINT
04563 template<> struct ConwayPolynomial<947, 2> { using ZPZ = aerobus::zpz<947>; using type =
       POLYV<ZPZV<1>, ZPZV<943>, ZPZV<2»; }; // NOLINT
04564 template<> struct ConwayPolynomial<947, 3> { using ZPZ = aerobus::zpz<947>; using type =
       POLYV<ZPZV<1>, ZPZV<0>, ZPZV<3>, ZPZV<945»; }; // NOLINT
04565 template<> struct ConwayPolynomial<947, 4> { using ZPZ = aerobus::zpz<947>; using type =
POLYV<ZPZV<1>, ZPZV<0>, ZPZV<8>, ZPZV<894>, ZPZV<2»; }; // NOLINT

04566 template<> struct ConwayPolynomial<947, 5> { using ZPZ = aerobus::zpz<947>; using type =
       POLYV<ZPZV<1>, ZPZV<0>, ZPZV<0>, ZPZV<0>, ZPZV<15>, ZPZV<945»; }; // NOLINT
04567 template<> struct ConwayPolynomial<947, 6> { using ZPZ = aerobus::zpz<947>; using type =
       POLYV<ZPZV<1>, ZPZV<0>, ZPZV<2>, ZPZV<880>, ZPZV<787>, ZPZV<95>, ZPZV<2»; }; // NOLINT
04568 template<> struct ConwayPolynomial<947, 7> { using ZPZ = aerobus::zpz<947>; using type =
       POLYV<ZPZV<1>, ZPZV<0>, ZPZV<0>, ZPZV<0>, ZPZV<0>, ZPZV<6>, ZPZV<6>, ZPZV<945»; };
04569 template<> struct ConwayPolynomial<947, 8> { using ZPZ = aerobus::zpz<947>; using type
       POLYV<ZPZV<1>, ZPZV<0>, ZPZV<0>, ZPZV<0>, ZPZV<1>, ZPZV<845>, ZPZV<597>, ZPZV<581>, ZPZV<2»; }; //
04570 template<> struct ConwayPolynomial<947, 9> { using ZPZ = aerobus::zpz<947>; using type =
       POLYV<ZPZV<1>, ZPZV<0>, ZPZV<0>, ZPZV<0>, ZPZV<0>, ZPZV<0>, ZPZV<0>, ZPZV<26>, ZPZV<26>, ZPZV<269>, ZPZV<808>, ZPZV<945»;
       }; // NOLINT
04571 template<> struct ConwayPolynomial<953, 1> { using ZPZ = aerobus::zpz<953>; using type =
       POLYV<ZPZV<1>, ZPZV<950»; }; // NOLINT
04572 template<> struct ConwayPolynomial<953, 2> { using ZPZ = aerobus::zpz<953>; using type =
       POLYV<ZPZV<1>, ZPZV<947>, ZPZV<3»; }; // NOLINT
04573 template<> struct ConwayPolynomial<953, 3> { using ZPZ = aerobus::zpz<953>; using type =
POLYV<ZPZV<1>, ZPZV<0>, ZPZV<7>, ZPZV<950»; }; // NOLINT

04574 template<> struct ConwayPolynomial<953, 4> { using ZPZ = aerobus::zpz<953>; using type = POLYV<ZPZV<1>, ZPZV<0>, ZPZV<1>, ZPZV<865>, ZPZV<3»; }; // NOLINT

04575 template<> struct ConwayPolynomial<953, 5> { using ZPZ = aerobus::zpz<953>; using type =
       POLYV<ZPZV<1>, ZPZV<0>, ZPZV<0>, ZPZV<0>, ZPZV<1>, ZPZV<950»; }; // NOLINT
04576 template<> struct ConwayPolynomial<953, 6> { using ZPZ = aerobus::zpz<953>; using type =
POLYV<ZPZV<1>, ZPZV<0>, ZPZV<1>, ZPZV<507>, ZPZV<829>, ZPZV<730>, ZPZV<3»; }; // NOLINT 04577 template<> struct ConwayPolynomial<953, 7> { using ZPZ = aerobus::zpz<953>; using type =
       POLYV<ZPZV<1>, ZPZV<0>, ZPZV<0>, ZPZV<0>, ZPZV<0>, ZPZV<0>, ZPZV<5>, ZPZV<5>, ZPZV<950»; };
04578 template<> struct ConwayPolynomial<953, 8> { using ZPZ = aerobus::zpz<953>; using type =
       POLYV<ZPZV<1>, ZPZV<0>, ZPZV<0>, ZPZV<0>, ZPZV<6>, ZPZV<579>, ZPZV<658>, ZPZV<108>, ZPZV<3»; }; //
04579 template<> struct ConwayPolynomial<953, 9> { using ZPZ = aerobus::zpz<953>; using type = POLYV<ZPZV<1>, ZPZV<0>, ZPZV<0>, ZPZV<0>, ZPZV<0>, ZPZV<0>, ZPZV<1>, ZPZV<819>, ZPZV<316>, ZPZV<950»;
       }; // NOLINT
04580 template<> struct ConwayPolynomial<967, 1> { using ZPZ = aerobus::zpz<967>; using type =
       POLYV<ZPZV<1>, ZPZV<962»; }; // NOLINT
04581 template<> struct ConwayPolynomial<967, 2> { using ZPZ = aerobus::zpz<967>; using type =
POLYV<ZPZV<1>, ZPZV<965>, ZPZV<5»; }; // NOLINT
04582 template<> struct ConwayPolynomial<967, 3> { using ZPZ = aerobus::zpz<967>; using type = POLYV<ZPZV<1>, ZPZV<0>, ZPZV<1>, ZPZV<962»; }; // NOLINT
```

```
04583 template<> struct ConwayPolynomial<967, 4> { using ZPZ = aerobus::zpz<967>; using type =
POLYY<ZPZV<1>, ZPZV<0>, ZPZV<963>, ZPZV<963>, ZPZV<5»; }; // NOLINT
04584 template<> struct ConwayPolynomial<967, 5> { using ZPZ = aerobus::zpz<967>; using type =
         POLYV<ZPZV<1>, ZPZV<0>, ZPZV<0>, ZPZV<0>, ZPZV<2>, ZPZV<962»; }; // NOLINT
04585 template<> struct ConwayPolynomial<967, 6> { using ZPZ = aerobus::zpz<967>; using type =
POLYV<ZPZV<1>, ZPZV<0>, ZPZV<6>, ZPZV<805>, ZPZV<948>, ZPZV<848>, ZPZV<848>, ZPZV<848>; // NOLINT 04586 template<> struct ConwayPolynomial<967, 7> { using ZPZ = aerobus::zpz<967>; using type =
         POLYV<ZPZV<1>, ZPZV<0>, ZPZV<0>, ZPZV<0>, ZPZV<0>, ZPZV<0>, ZPZV<0>, ZPZV<9>, ZPZV<92»; };
04587 template<> struct ConwayPolynomial<967, 8> { using ZPZ = aerobus::zpz<967>; using type =
         POLYV<ZPZV<1>, ZPZV<0>, ZPZV<0>, ZPZV<0>, ZPZV<15>, ZPZV<840>, ZPZV<502>, ZPZV<136>, ZPZV<5»; }; //
         NOLINT
04588 template<> struct ConwayPolynomial<967, 9> { using ZPZ = aerobus::zpz<967>; using type = POLYV<ZPZV<1>, ZPZV<0>, ZPZV<0>, ZPZV<0>, ZPZV<0>, ZPZV<0>, ZPZV<51>, ZPZV<512>, ZPZV<783>, 
         }; // NOLINT
04589 template<> struct ConwayPolynomial<971, 1> { using ZPZ = aerobus::zpz<971>; using type =
         POLYV<ZPZV<1>, ZPZV<965»; }; // NOLINT
04590 template<> struct ConwayPolynomial<971, 2> { using ZPZ = aerobus::zpz<971>; using type =
POLYV<ZPZV<1>, ZPZV<970>, ZPZV<6»; }; // NOLINT

04591 template<> struct ConwayPolynomial<971, 3> { using ZPZ = aerobus::zpz<971>; using type =
         POLYV<ZPZV<1>, ZPZV<0>, ZPZV<3>, ZPZV<965»; }; // NOLINT
04592 template<> struct ConwayPolynomial<971, 4> { using ZPZ = aerobus::zpz<971>; using type = POLYV<ZPZV<1>, ZPZV<0>, ZPZV<2>, ZPZV<527>, ZPZV<6>; }; // NOLINT
04593 template<> struct ConwayPolynomial<971, 5> { using ZPZ = aerobus::zpz<971>; using type =
POLYY<ZPZV<1>, ZPZV<0>, ZPZV<0>, ZPZV<0>, ZPZV<14>, ZPZV<965»; }; // NOLINT
04594 template<> struct ConwayPolynomial<971, 6> { using ZPZ = aerobus::zpz<971>; using type =
         POLYV<ZPZV<1>, ZPZV<0>, ZPZV<1>, ZPZV<970>, ZPZV<729>, ZPZV<718>, ZPZV<6»; }; // NOLINT
04595 template<> struct ConwayPolynomial<971, 7> { using ZPZ = aerobus::zpz<971>;
                                                                                                                            using type
         POLYV<ZPZV<1>, ZPZV<0>, ZPZV<0>, ZPZV<0>, ZPZV<0>, ZPZV<0>, ZPZV<13>, ZPZV<965»; }; // NOLINT
04596 template<> struct ConwayPolynomial<971, 8> { using ZPZ = aerobus::zpz<971>; using type =
         POLYV<ZPZV<1>, ZPZV<0>, ZPZV<0>, ZPZV<0>, ZPZV<72>, ZPZV<725>, ZPZV<281>, ZPZV<206>, ZPZV<6»; }; //
         NOLINT
04597 template<> struct ConwayPolynomial<971, 9> { using ZPZ = aerobus::zpz<971>; using type
         POLYV<ZPZV<1>, ZPZV<0>, ZPZV<0>, ZPZV<0>, ZPZV<0>, ZPZV<0>, ZPZV<0>, ZPZV<805>, ZPZV<805>, ZPZV<473>, ZPZV<965»;
         }; // NOLINT
04598 template<> struct ConwayPolynomial<977, 1> { using ZPZ = aerobus::zpz<977>; using type =
         POLYV<ZPZV<1>, ZPZV<974»; }; // NOLINT
04599 template<> struct ConwayPolynomial<977, 2> { using ZPZ = aerobus::zpz<977>; using type =
         POLYV<ZPZV<1>, ZPZV<972>, ZPZV<3»; }; // NOLINT
04600 template<> struct ConwayPolynomial<977, 3> { using ZPZ = aerobus::zpz<977>; using type =
         POLYV<ZPZV<1>, ZPZV<0>, ZPZV<1>, ZPZV<974»; }; // NOLINT
04601 template<> struct ConwayPolynomial<977, 4> { using ZPZ = aerobus::zpz<977>; using type =
POLYV<ZPZV<1>, ZPZV<0>, ZPZV<0>, ZPZV<800>, ZPZV<3»; }; // NOLINT
04602 template<> struct ConwayPolynomial<977, 5> { using ZPZ = aerobus::zpz<977>; using type =
         POLYV<ZPZV<1>, ZPZV<0>, ZPZV<0>, ZPZV<0>, ZPZV<11>, ZPZV<974»; }; // NOLINT
04603 template<> struct ConwayPolynomial<977, 6> { using ZPZ = aerobus::zpz<977>; using type =
         POLYV<ZPZV<1>, ZPZV<0>, ZPZV<2>, ZPZV<729>, ZPZV<830>, ZPZV<753>, ZPZV<3»; };
04604 template<> struct ConwayPolynomial<977, 7> { using ZPZ = aerobus::zpz<977>; using type =
         POLYV<ZPZV<1>, ZPZV<0>, ZPZV<0>, ZPZV<0>, ZPZV<0>, ZPZV<0>, ZPZV<7>, ZPZV<74»; }; // NOLINT
04605 template<> struct ConwayPolynomial<977, 8> { using ZPZ = aerobus::zpz<977>; using type = POLYV<ZPZV<1>, ZPZV<0>, ZPZV<0>, ZPZV<0>, ZPZV<0>, ZPZV<85>, ZPZV<807>, ZPZV<77>, ZPZV<3»; }; //
04606 template<> struct ConwayPolynomial<977, 9> { using ZPZ = aerobus::zpz<977>; using type =
         POLYV<ZPZV<1>, ZPZV<0>, ZPZV<0>, ZPZV<0>, ZPZV<0>, ZPZV<0>, ZPZV<0>, ZPZV<450>, ZPZV<740>, ZPZV<740>,
          }; // NOLINT
04607 template<> struct ConwayPolynomial<983, 1> { using ZPZ = aerobus::zpz<983>; using type =
         POLYV<ZPZV<1>, ZPZV<978»; }; // NOLINT
04608 template<> struct ConwayPolynomial<983, 2> { using ZPZ = aerobus::zpz<983>; using type =
         POLYV<ZPZV<1>, ZPZV<981>, ZPZV<5»; }; // NOLINT
04609 template<> struct ConwayPolynomial<983, 3> { using ZPZ = aerobus::zpz<983>; using type =
POLYY<ZPZV<1>, ZPZV<0>, ZPZV<1>, ZPZV<978»; }; // NOLINT

04610 template<> struct ConwayPolynomial<983, 4> { using ZPZ = aerobus::zpz<983>; using type = POLYV<ZPZV<1>, ZPZV<0>, ZPZV<5>, ZPZV<567>, ZPZV<5»; }; // NOLINT

04611 template<> struct ConwayPolynomial<983, 5> { using ZPZ = aerobus::zpz<983>; using type =
         POLYV<ZPZV<1>, ZPZV<0>, ZPZV<0>, ZPZV<0>, ZPZV<8>, ZPZV<978»; }; // NOLINT
04612 template<> struct ConwayPolynomial<983, 6> { using ZPZ = aerobus::zpz<983>; using type =
         POLYV<ZPZV<1>, ZPZV<0>, ZPZV<2>, ZPZV<849>, ZPZV<296>, ZPZV<228>, ZPZV<5»; }; // NOLINT
04613 template<> struct ConwayPolynomial<983, 7> { using ZPZ = aerobus::zpz<983>; using type =
         POLYV<ZPZV<1>, ZPZV<0>, ZPZV<0>, ZPZV<0>, ZPZV<0>, ZPZV<0>, ZPZV<0>, ZPZV<3>, ZPZV<978»; };
04614 template<> struct ConwayPolynomial<983, 8> { using ZPZ = aerobus::zpz<983>; using type =
         POLYV<ZPZV<1>, ZPZV<0>, ZPZV<0>, ZPZV<0>, ZPZV<7>, ZPZV<738>, ZPZV<276>, ZPZV<530>, ZPZV<5»; }; //
04615 template<> struct ConwayPolynomial<983, 9> { using ZPZ = aerobus::zpz<983>; using type =
         POLYV<ZPZV<1>, ZPZV<0>, ZPZV<0>, ZPZV<0>, ZPZV<0>, ZPZV<0>, ZPZV<0>, ZPZV<0>, ZPZV<0>, ZPZV<858>, ZPZV<878,
         }; // NOLINT
04616 template<> struct ConwayPolynomial<991, 1> { using ZPZ = aerobus::zpz<991>; using type =
         POLYV<ZPZV<1>, ZPZV<985»; }; // NOLINT
04617 template<> struct ConwayPolynomial<991, 2> { using ZPZ = aerobus::zpz<991>; using type =
         POLYV<ZPZV<1>, ZPZV<989>, ZPZV<6»; }; // NOLINT
04618 template<> struct ConwayPolynomial<991, 3> { using ZPZ = aerobus::zpz<991>; using type =
         POLYV<ZPZV<1>, ZPZV<0>, ZPZV<4>, ZPZV<985»; }; // NOLINT
04619 template<> struct ConwayPolynomial<991, 4> { using ZPZ = aerobus::zpz<991>; using type =
POLYV<ZPZV<1>, ZPZV<0>, ZPZV<10>, ZPZV<794>, ZPZV<6»; }; // NOLINT
04620 template<> struct ConwayPolynomial<991, 5> { using ZPZ = aerobus::zpz<991>; using type =
         POLYV<ZPZV<1>, ZPZV<0>, ZPZV<0>, ZPZV<0>, ZPZV<3>, ZPZV<985»; }; // NOLINT
04621 template<> struct ConwayPolynomial<991, 6> { using ZPZ = aerobus::zpz<991>; using type =
POLYV<ZPZV<1>, ZPZV<0>, ZPZV<0>, ZPZV<637>, ZPZV×855>, ZPZV<278>, ZPZV<6%; }; // NOLINT 04622 template<> struct ConwayPolynomial<991, 7> { using ZPZ = aerobus::zpz<991>; using type =
```

```
POLYV<ZPZV<1>, ZPZV<0>, ZPZV<0>, ZPZV<0>, ZPZV<0>, ZPZV<0>, ZPZV<7>, ZPZV<985»; }; // NOLINT
04623 template<> struct ConwayPolynomial<991, 8> { using ZPZ = aerobus::zpz<991>; using type =
                         POLYV<ZPZV<1>, ZPZV<0>, ZPZV<0>, ZPZV<0>, ZPZV<15>, ZPZV<786>, ZPZV<234>, ZPZV<6»; }; //
                        NOLINT
04624 template<> struct ConwayPolynomial<991, 9> { using ZPZ = aerobus::zpz<991>; using type = POLYV<ZPZV<1>, ZPZV<0>, ZPZV<0>, ZPZV<0>, ZPZV<0>, ZPZV<0>, ZPZV<0>, ZPZV<2>, ZPZV<2>, ZPZV<2>, ZPZV<2>, ZPZV<2>, ZPZV<3>, ZPZV<3
                         }; // NOLINT
 04625 template<> struct ConwayPolynomial<997, 1> { using ZPZ = aerobus::zpz<997>; using type =
                        POLYV<ZPZV<1>, ZPZV<990»; }; // NOLINT
 04626 template<> struct ConwayPolynomial<997, 2> { using ZPZ = aerobus::zpz<997>; using type =
O4626 template<> struct conwayrolynomials, // NoLINT

O4627 template<> struct conwayrolynomials, // NoLINT

04627 template<> struct conwayrolynomials, 3> { using ZPZ = aerobus::zpz<997>; using type = POLYV<ZPZV<1>, ZPZV<0>, ZPZV<2>, ZPZV<990»; }; // NOLINT

04628 template<> struct Conwayrolynomials, 297, 4> { using ZPZ = aerobus::zpz<997>; using type = CONWAYROLYNOMIALS, 2000, 2000, 2000, 2000, 2000, 2000, 2000, 2000, 2000, 2000, 2000, 2000, 2000, 2000, 2000, 2000, 2000, 2000, 2000, 2000, 2000, 2000, 2000, 2000, 2000, 2000, 2000, 2000, 2000, 2000, 2000, 2000, 2000, 2000, 2000, 2000, 2000, 2000, 2000, 2000, 2000, 2000, 2000, 2000, 2000, 2000, 2000, 2000, 2000, 2000, 2000, 2000, 2000, 2000, 2000, 2000, 2000, 2000, 2000, 2000, 2000, 2000, 2000, 2000, 2000, 2000, 2000, 2000, 2000, 2000, 2000, 2000, 2000, 2000, 2000, 2000, 2000, 2000, 2000, 2000, 2000, 2000, 2000, 2000, 2000, 2000, 2000, 2000, 2000, 2000, 2000, 2000, 2000, 2000, 2000, 2000, 2000, 2000, 2000, 2000, 2000, 2000, 2000, 2000, 2000, 2000, 2000, 2000, 2000, 2000, 2000, 2000, 2000, 2000, 2000, 2000, 2000, 2000, 2000, 2000, 2000, 2000, 2000, 2000, 2000, 2000, 2000, 2000, 2000, 2000, 2000, 2000, 2000, 2000, 2000, 2000, 2000, 2000, 2000, 2000, 2000, 2000, 2000, 2000, 2000, 2000, 2000, 2000, 2000, 2000, 2000, 2000, 2000, 2000, 2000, 2000, 2000, 2000, 2000, 2000, 2000, 2000, 2000, 2000, 2000, 2000, 2000, 2000, 2000, 2000, 2000, 2000, 2000, 2000, 2000, 2000, 2000, 2000, 2000, 2000, 2000, 2000, 2000, 2000, 2000, 2000, 2000, 2000, 2000, 2000, 2000, 2000, 2000, 2000, 2000, 2000, 2000, 2000, 2000, 2000, 2000, 2000, 2000, 2000, 2000, 2000, 2000, 2000, 2000, 2000, 2000, 2000, 2000, 2000, 2000, 2000, 2000, 2000, 2000, 2000, 2000, 2000, 2000, 2000, 2000, 2000, 2000, 2000, 2000, 2000, 2000, 2000, 2000, 2000, 2000, 2000, 2000, 2000, 2000, 2000, 2000, 2000, 2000, 2000, 2000, 2000, 2000, 2000, 2000, 2000, 2000, 2000, 2000, 2000, 2000, 2000, 2000, 2000, 2000, 2000, 2000, 2000, 2000, 2000, 2000, 2000, 2000, 2000, 2000, 2000, 2000, 2000, 2000, 2000, 2000, 2000, 2000, 2000, 2000
POLYV<ZPZV<1>, ZPZV<0>, ZPZV<4>, ZPZV<622>, ZPZV<7»; }; // NOLINT

04629 template<> struct ConwayPolynomial
997, 5> { using ZPZ = aerobus::zpz<997>; using type = POLYV<ZPZV<1>, ZPZV<0>, ZPZV<0>, ZPZV<0>, ZPZV<1>, ZPZV<990»; }; // NOLINT

04630 template<> struct ConwayPolynomial
997, 6> { using ZPZ = aerobus::zpz<997>; using type =
                        POLYV<ZPZV<1>, ZPZV<0>, ZPZV<0>, ZPZV<981>, ZPZV<58>, ZPZV<260>, ZPZV<7»; }; // NOLINT
04631 template<> struct ConwayPolynomial<997, 7> { using ZPZ = aerobus::zpz<997>; using type
                        POLYV<ZPZV<1>, ZPZV<0>, ZPZV<0>, ZPZV<0>, ZPZV<0>, ZPZV<0>, ZPZV<0>, ZPZV<0>, ZPZV<1>, ZPZV<990»; };
04632 template<> struct ConwayPolynomial<997, 8> { using ZPZ = aerobus::zpz<997>; using type =
                        POLYV<ZPZV<1>, ZPZV<0>, ZPZV<0>, ZPZV<0>, ZPZV<0>, ZPZV<3>, ZPZV<473>, ZPZV<473>, ZPZV<241>, ZPZV<2*, 
                        NOLINT
04633 template<> struct ConwayPolynomial<997, 9> { using ZPZ = aerobus::zpz<997>; using type =
                        POLYV<ZPZV<1>, ZPZV<0>, ZPZV<0>, ZPZV<0>, ZPZV<0>, ZPZV<0>, ZPZV<3>, ZPZV<732>, ZPZV<616>, ZPZV<990»;
                          }; // NOLINT
 04634 #endif // AEROBUS_CONWAY_IMPORTS
 04635
04636 #endif // __INC_AEROBUS__ // NOLINT
```

Chapter 7

Examples

7.1 i32::template

inject a native constant

inject a native constant

Template Parameters

x | inject_constant_2<2> -> i32::template val<2>

7.2 i64::template

injects constant as an i64 value

injects constant as an i64 value

Template Parameters

x inject_constant_t<2>

7.3 polynomial

makes the constant (native type) polynomial a_0

makes the constant (native type) polynomial a_0

Template Parameters

x <i32>::template inject_constant_t<2>

124 Examples

7.4 PI_fraction::val

representation of PI as a continued fraction -> 3.14...

7.5 E_fraction::val

approximation of e -> 2.718...

approximation of e -> 2.718...

Index

```
add t
                                                                   valueRing, P >::inner< index, stop >, 19
                                                         aerobus::polynomial < Ring, variable_name >::horner_evaluation <
     aerobus::i64, 15
     aerobus::polynomial < Ring, variable name >, 21
                                                                   valueRing, P >::inner< stop, stop >, 19
                                                         aerobus::polynomial< Ring, variable name >::val< co-
aerobus::ContinuedFraction < a0 >, 10
aerobus::ContinuedFraction < a0, rest... >, 11
                                                                   effN >, 38
aerobus::ContinuedFraction < values >, 10
                                                         aerobus::polynomial < Ring, variable_name >::val < co-
aerobus::i32, 11
                                                                   effN >::coeff at< index, E >, 9
                                                         aerobus::polynomial < Ring, variable_name >::val < co-
     eq_v, 13
                                                                   effN >::coeff_at< index, std::enable_if_t<(index<
     pos_v, 13
aerobus::i32::val< x >, 31
                                                                   0 \mid | index > 0) > 0, 9
     eval, 32
                                                         aerobus::polynomial < Ring, variable name >::val < co-
     get. 32
                                                                   effN >::coeff at< index, std::enable if t<(index==0)>
aerobus::i64, 14
                                                                   >, 9
     add_t, 15
                                                         aerobus::polynomial < Ring, variable_name >::val < co-
                                                                   effN, coeffs >, 35
     div t, 15
     eq_t, 16
                                                              coeff_at_t, 36
                                                              eval, 36
     eq_v, 18
     gcd_t, 16
                                                              to_string, 36
                                                         aerobus::Quotient < Ring, X >, 26
     gt_t, 16
                                                         aerobus::Quotient< Ring, X >::val< V >, 37
     gt_v, 18
     It t, 16
                                                         aerobus::type list< Ts >, 28
     It v, 18
                                                              at. 29
     mod t, 17
                                                              concat, 29
                                                              insert, 29
     mul t, 17
                                                              push back, 29
     pos t, 17
     pos v, 18
                                                              push front, 30
     sub t, 17
                                                              remove, 30
aerobus::i64::val < x >, 32
                                                         aerobus::type_list< Ts >::pop_front, 25
     eval, 33
                                                         aerobus::type_list< Ts >::split< index >, 26
     get, 33
                                                         aerobus::type list<>, 30
aerobus::is prime< n >, 19
                                                         aerobus::zpz , 39
aerobus::IsEuclideanDomain, 7
                                                         aerobus::zpz<p>::val<math><x>, 37
aerobus::IsField, 7
                                                         at
aerobus::IsRing, 8
                                                              aerobus::type list< Ts >, 29
aerobus::polynomial < Ring, variable_name >, 20
                                                         coeff at t
     add_t, 21
                                                              aerobus::polynomial<
                                                                                        Ring,
                                                                                                  variable name
     derive_t, 22
                                                                   >::val< coeffN, coeffs >, 36
     div t, 22
                                                         concat
     eq_t, 22
                                                              aerobus::type_list< Ts >, 29
     gcd_t, 22
     gt_t, 23
                                                         derive t
     It t, 23
                                                              aerobus::polynomial < Ring, variable_name >, 22
     mod t, 23
                                                         div_t
     monomial t, 24
                                                              aerobus::i64, 15
     mul t, 24
                                                              aerobus::polynomial < Ring, variable_name >, 22
     pos t, 24
     simplify_t, 24
                                                         eq t
     sub t, 25
aerobus::polynomial< Ring, variable_name >::horner_evaluation<
                                                               \stackrel{"}{a}erobus::polynomial < Ring, variable name >, 22
```

126 INDEX

```
eq_v
                                                        to_string
     aerobus::i32, 13
                                                             aerobus::polynomial<
                                                                                       Ring,
                                                                                                 variable_name
     aerobus::i64, 18
                                                                  >::val< coeffN, coeffs >, 36
eval
     aerobus::i32::val< x >, 32
    aerobus::i64::val < x >, 33
                                        variable_name
     aerobus::polynomial<
                              Ring,
          >::val< coeffN, coeffs >, 36
gcd_t
     aerobus::i64, 16
     aerobus::polynomial < Ring, variable name >, 22
get
    aerobus::i32::val< x >, 32
     aerobus::i64::val < x >, 33
gt_t
     aerobus::i64, 16
     aerobus::polynomial < Ring, variable_name >, 23
gt_v
     aerobus::i64, 18
insert
    aerobus::type_list< Ts >, 29
lt t
     aerobus::i64, 16
     aerobus::polynomial < Ring, variable_name >, 23
lt_v
     aerobus::i64, 18
mod_t
    aerobus::i64, 17
     aerobus::polynomial < Ring, variable_name >, 23
monomial_t
     aerobus::polynomial < Ring, variable name >, 24
mul t
     aerobus::i64, 17
    aerobus::polynomial < Ring, variable_name >, 24
pos_t
     aerobus::i64, 17
    aerobus::polynomial < Ring, variable name >, 24
pos_v
     aerobus::i32, 13
    aerobus::i64, 18
push_back
     aerobus::type_list< Ts >, 29
push_front
    aerobus::type_list< Ts >, 30
remove
    aerobus::type_list< Ts >, 30
simplify t
     aerobus::polynomial < Ring, variable name >, 24
src/aerobus.h, 41
sub t
     aerobus::i64, 17
     aerobus::polynomial < Ring, variable_name >, 25
```