Aerobus

v1.2

Generated by Doxygen 1.9.8

Chapter 1

Concept Index

1.1 Concepts

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

Chapter 2

Class Index

2.1 Class List

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

Chapter 3

File Index

3.1 File List

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

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_v<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 >::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 >::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 >::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 **lt_t** = typename lt< 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

v	

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

  type for actual values
- 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 lt_t = typename lt< 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 lt_v = lt_t<v1, v2>::value

  strictly smaller operator (v1 < v2) - boolean value
- 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)

5.8.1 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

Template Parameters

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)

Template Parameters

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

Template Parameters

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

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

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

```
template<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 >::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 >::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

- static constexpr bool **value** = internal::_is_prime<n, 5>::value

  true iff n is prime

5.11.1 Detailed Description

template<size_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 > 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^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 >, 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>
requires IsEuclideanDomain<Ring>
struct aerobus::polynomial< Ring >

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

adds two polynomials

Template Parameters

v1	
v2	

5.12.2.2 derive_t

```
template<typename Ring >
template<typename v >
using aerobus::polynomial< Ring >::derive_t = typename derive_helper<v>::type
```

derivation operator

Template Parameters

v	

5.12.2.3 div_t

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

division operator

Template Parameters

v1	
v2	

5.12.2.4 eq_t

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

equality operator

Template Parameters

v1	
v2	

5.12.2.5 gcd_t

```
template<typename Ring >
```

```
template<typename v1 , typename v2 >
using aerobus::polynomial< Ring >::gcd_t = std::conditional_t< Ring::is_euclidean_domain,
typename make_unit<gcd_t<polynomial<Ring>, v1, v2> >::type, void>
```

greatest common divisor of two polynomials

Template Parameters

v1	
v2	

5.12.2.6 gt_t

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

strict greater operator

Template Parameters

v1	
v2	

5.12.2.7 lt_t

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

strict less operator

Template Parameters

v1	
v2	

5.12.2.8 mod_t

```
template<typename Ring >
template<typename v1 , typename v2 >
using aerobus::polynomial< Ring >::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 >
template<typename coeff , size_t deg>
using aerobus::polynomial< Ring >::monomial_t = typename monomial<coeff, deg>::type
```

monomial : coeff X^deg

Template Parameters

coeff	
deg	

5.12.2.10 mul_t

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

multiplication of two polynomials

Template Parameters

v1	
v2	

5.12.2.11 pos_t

```
template<typename Ring >
template<typename v >
using aerobus::polynomial< Ring >::pos_t = typename Ring::template pos_t<typename v::aN>
```

checks for positivity (an > 0)

Template Parameters

v	

5.12.2.12 simplify_t

```
template<typename Ring >
template<typename P >
using aerobus::polynomial< Ring >::simplify_t = typename simplify<P>::type
```

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

Template Parameters

P	

5.12.2.13 sub_t

```
template<typename Ring >
template<typename v1 , typename v2 >
using aerobus::polynomial< Ring >::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

Quotient ring by the principal ideal generated by 'X' With i32 as Ring and i32::val<2> as X, Quotient is Z/2Z.

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

Classes

- struct val

    projection values in the quotient ring

Public Types

- using **zero** = val< typename Ring::zero >

    zero value
- using **one** = val< typename Ring::one >

    one
- template<typename v1 , typename v2 >
  using add_t = val< typename Ring::template add_t< typename v1::type, typename v2::type > >

    addition operator
- template<typename v1 , typename v2 >
  using mul_t = val< typename Ring::template mul_t< typename v1::type, typename v2::type > >

    substraction operator
- template<typename v1 , typename v2 >
  using div_t = val< typename Ring::template div_t< typename v1::type, typename v2::type > >

    division operator
- template<typename v1 , typename v2 >
  using mod_t = val< typename Ring::template mod_t< typename v1::type, typename v2::type > >

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

    equality operator (as type)
- template<typename v1 >
  using pos_t = std::true_type

    positivity operator always true
- template<auto x>
  using inject_constant_t = val< typename Ring::template inject_constant_t< x > >

    inject a 'constant' in quotient ring for example : QuotientRing<i32, i32::val<2>>::inject_constant_t<1>
- template<typename v >
  using inject_ring_t = val< v >

    projects a value of Ring onto the quotient for example : QuotientRing<i32, i32::val<2>>::inject_ring_t<i32::val<1>>

Static Public Attributes

- template<typename v1 , typename v2 >
  static constexpr bool eq_v = Ring::template eq_t<typename v1::type, typename v2::type>::value

    addition operator (as boolean value)
- template<typename v >
  static constexpr bool pos_v = pos_t<v>::value

    positivity operator always true
- static constexpr bool **is_euclidean_domain** = true

    quotien rings are euclidean domain

5.14.1 Detailed Description

template<typename Ring, typename X>
requires IsRing<Ring>
struct aerobus::Quotient< Ring, X >

Quotient ring by the principal ideal generated by 'X' With i32 as Ring and i32::val<2> as X, Quotient is Z/2Z.

Template Parameters

Ring	A ring type, such as 'i32', must satisfy the IsRing concept
X	a value in Ring, such as i32::val<2>

5.14.2 Member Typedef Documentation

5.14.2.1 add_t

```
template<typename Ring , typename X >
template<typename v1 , typename v2 >
using aerobus::Quotient< Ring, X >::add_t = val<typename Ring::template add_t<typename v1::type,
typename v2::type> >
```

addition operator

Template Parameters

v1	a value in quotient ring
v2	a value in quotient ring

5.14.2.2 div_t

```
template<typename Ring , typename X >
template<typename v1 , typename v2 >
using aerobus::Quotient< Ring, X >::div_t = val<typename Ring::template div_t<typename v1::type,
typename v2::type> >
```

division operator

Template Parameters

v1	a value in quotient ring
v2	a value in quotient ring

5.14.2.3 eq_t

```
template<typename Ring , typename X >
template<typename v1 , typename v2 >
```

```
using aerobus::Quotient< Ring, X >::eq_t = typename Ring::template eq_t<typename v1::type,
typename v2::type>
```

equality operator (as type)

Template Parameters

v1	a value in quotient ring
v2	a value in quotient ring

5.14.2.4 inject_constant_t

```
template<typename Ring , typename X >
template<auto x>
using aerobus::Quotient< Ring, X >::inject_constant_t = val<typename Ring::template inject_constant_t<x>
>
```

inject a 'constant' in quotient ring for example : QuotientRing<i32, i32::val<2>>::inject_constant_t<1>

Template Parameters

x	a 'constant' from Ring point of view

5.14.2.5 inject_ring_t

```
template<typename Ring , typename X >
template<typename v >
using aerobus::Quotient< Ring, X >::inject_ring_t = val<v>
```

projects a value of Ring onto the quotient for example : QuotientRing<i32, i32::val<2>>::inject_ring_t<i32::val<1>>

Template Parameters

v	a value in Ring

5.14.2.6 mod_t

```
template<typename Ring , typename X >
template<typename v1 , typename v2 >
using aerobus::Quotient< Ring, X >::mod_t = val<typename Ring::template mod_t<typename v1::type,
typename v2::type> >
```

modulus operator

Template Parameters

v1	a value in quotient ring
v2	a value in quotient ring

5.14.2.7 mul_t

```
template<typename Ring , typename X >
template<typename v1 , typename v2 >
using aerobus::Quotient< Ring, X >::mul_t = val<typename Ring::template mul_t<typename v1::type,
typename v2::type> >
```

substraction operator

Template Parameters

v1	a value in quotient ring
v2	a value in quotient ring

5.14.2.8 pos_t

```
template<typename Ring , typename X >
template<typename v1 >
using aerobus::Quotient< Ring, X >::pos_t = std::true_type
```

positivity operator always true

Template Parameters

v1	a value in quotient ring

5.14.3 Member Data Documentation

5.14.3.1 eq_v

```
template<typename Ring , typename X >
template<typename v1 , typename v2 >
constexpr bool aerobus::Quotient< Ring, X >::eq_v = Ring::template eq_t<typename v1::type,
typename v2::type>::value  [static], [constexpr]
```

addition operator (as boolean value)

Template Parameters

v1	a value in quotient ring
v2	a value in quotient ring

5.14.3.2 pos_v

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

positivity operator always true

Template Parameters

v1	a value in quotient ring

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 for example type_list<int, double, float>

Template Parameters

...Ts	types to store and manipulate at compile time

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

index	

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

U	

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

Template Parameters

index	
T	

5.16.2.4 push_back

```
template<typename... 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

T	

5.16.2.5 push_front

```
template<typename... Ts>
template<typename T >
using aerobus::type_list< Ts >::push_front = type_list<T, Ts...>
```

Adds T to front of the list.

Template Parameters

T	

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

index	

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

- src/aerobus.h

5.17 aerobus::type_list<> Struct Reference

specialization for empty type list

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

Public Types

- template< typename 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

5.17.1 Detailed Description

specialization for empty type list

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 **ring_type** = i32

  Enclosing ring type.
- 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

Template Parameters

x	an actual integer

5.18.2 Member Function Documentation

5.18.2.1 eval()

```
template<int32_t x>
template<typename valueRing >
static constexpr valueRing aerobus::i32::val< x >::eval (
             const valueRing & v )  [inline], [static], [constexpr]
```

cast x into valueRing

Template Parameters

valueRing	double for example

5.18.2.2 get()

```
template<int32_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 **ring_type** = i64

  enclosing ring type
- 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

  actual value

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<int64_t x>
template<typename valueRing >
static constexpr valueRing aerobus::i64::val< x >::eval (
             const valueRing & v )  [inline], [static], [constexpr]
```

cast value in valueRing

Template Parameters

valueRing	(double for example)

5.19.2.2 get()

```
template<int64_t x>
template<typename valueType >
static constexpr valueType aerobus::i64::val< x >::get ( )  [inline], [static], [constexpr]
```

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 >::val< coeffN, coeffs > Struct Template Reference

values (seen as types) in polynomial ring

`#include <aerobus.h>`

Public Types

- using **ring_type** = polynomial< Ring >

  enclosing ring type
- using **aN** = coeffN

  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
- template<typename 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>
template<typename coeffN, typename... coeffs>
struct aerobus::polynomial< Ring >::val< coeffN, coeffs >

values (seen as types) in polynomial ring

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 >
template<typename coeffN , typename...  coeffs>
template<size_t index>
using aerobus::polynomial< Ring >::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()

```
template<typename Ring >
template<typename coeffN , typename...  coeffs>
template<typename valueRing >
static constexpr valueRing aerobus::polynomial< Ring >::val< coeffN, coeffs >::eval (
            const valueRing & x )  [inline], [static], [constexpr]
```

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 >
template<typename coeffN , typename... coeffs>
static std::string aerobus::polynomial< Ring >::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

projection values in the quotient ring

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

Public Types

- using **type** = abs_t< typename Ring::template mod_t< V, X > >

5.21.1 Detailed Description

template<typename Ring, typename X>
template<typename V>
struct aerobus::Quotient< Ring, X >::val< V >

projection values in the quotient ring

Template Parameters

V	a value from 'Ring'

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

- src/aerobus.h

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

Public Types

- using **ring_type** = zpz< p >

  enclosing ring type
- 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

  actual value

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

- src/aerobus.h

5.23 aerobus::polynomial< Ring >::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 **ring_type** = polynomial< Ring >

  enclosing ring type
- 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

  degree
- static constexpr bool **is_zero_v** = is_zero_t::value

5.23.1 Detailed Description

template<typename Ring>
template<typename coeffN>
struct aerobus::polynomial< Ring >::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< p > 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 **lt_t** = typename lt< 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)

Static Public Attributes

- static constexpr bool **is_field** = is_prime<p>::value
- static constexpr bool **is_euclidean_domain** = true
- template<typename v1 , typename v2 >
  static constexpr bool **gt_v** = gt_t<v1, v2>::value

  strictly greater operator (booleanvalue)
- template<typename v1 , typename v2 >
  static constexpr bool **lt_v** = lt_t<v1, v2>::value

  strictly smaller operator (booleanvalue)
- template<typename v1 , typename v2 >
  static constexpr bool **eq_v** = eq_t<v1, v2>::value

  equality operator (booleanvalue)
- template<typename v >
  static constexpr bool **pos_v** = pos_t<v>::value

  positivity operator (boolean value)

5.24.1 Detailed Description

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

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

6.1 aerobus.h

```
// -*- lsst-c++ -*-
#ifndef __INC_AEROBUS__ // NOLINT
#define __INC_AEROBUS__

#include <cstdint>
#include <cstddef>
#include <cstring>
#include <type_traits>
#include <utility>
#include <algorithm>
#include <functional>
#include <string>
#include <concepts> // NOLINT
#include <array>


#ifdef _MSC_VER
#define ALIGNED(x) __declspec(align(x))
#define INLINED __forceinline
#else
#define ALIGNED(x) __attribute__((aligned(x)))
#define INLINED __attribute__((always_inline)) inline
#endif

// aligned allocation
namespace aerobus {
    template<typename T>
    T* aligned_malloc(size_t count, size_t alignment) {
        #ifdef _MSC_VER
        return static_cast<T*>(_aligned_malloc(count * sizeof(T), alignment));
        #else
        return static_cast<T*>(aligned_alloc(alignment, count * sizeof(T)));
        #endif
    }
}  // namespace aerobus

// concepts
namespace aerobus {
    template <typename R>
    concept 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>;
    };

    template <typename R>
    concept 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_v<typename R::one> == true;
        // typename R::template gt_t<typename R::one, typename R::zero>;
        R::is_euclidean_domain == true;
    };

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

// utilities
namespace aerobus {
    namespace internal {
        template<template<typename...> typename TT, typename T>
        struct is_instantiation_of : std::false_type { };

        template<template<typename...> typename TT, typename... Ts>
        struct is_instantiation_of<TT, TT<Ts...» : std::true_type { };

        template<template<typename...> typename TT, typename T>
        inline constexpr bool is_instantiation_of_v = is_instantiation_of<TT, T>::value;

        template <int64_t i, typename T, typename... Ts>
        struct type_at {
            static_assert(i < sizeof...(Ts) + 1, "index out of range");
            using type = typename type_at<i - 1, Ts...>::type;
        };

        template <typename T, typename... Ts> struct type_at<0, T, Ts...> {
            using type = T;
        };

        template <size_t i, typename... Ts>
        using type_at_t = typename type_at<i, Ts...>::type;


        template<size_t n, size_t i, typename E = void>
        struct _is_prime {};

        template<size_t i>
        struct _is_prime<0, i> {
            static constexpr bool value = false;
        };

        template<size_t i>
        struct _is_prime<1, i> {
            static constexpr bool value = false;
        };

        template<size_t i>
        struct _is_prime<2, i> {
            static constexpr bool value = true;
        };

        template<size_t i>
        struct _is_prime<3, i> {
            static constexpr bool value = true;
        };

        template<size_t i>
        struct _is_prime<5, i> {
            static constexpr bool value = true;
        };

        template<size_t i>
        struct _is_prime<7, i> {
            static constexpr bool value = true;
        };

        template<size_t n, size_t i>
        struct _is_prime<n, i, std::enable_if_t<(n != 2 && n % 2 == 0)» {
            static constexpr bool value = false;
        };

        template<size_t n, size_t i>
        struct _is_prime<n, i, std::enable_if_t<(n != 2 && n != 3 && n % 2 != 0 && n % 3 == 0)» {
            static constexpr bool value = false;
        };

        template<size_t n, size_t i>
        struct _is_prime<n, i, std::enable_if_t<(n >= 9 && i * i > n)» {
            static constexpr bool value = true;
        };

        template<size_t n, size_t i>
        struct _is_prime<n, i, std::enable_if_t<(
            n % i == 0 &&
            n >= 9 &&
            n % 3 != 0 &&
            n % 2 != 0 &&
            i * i > n)» {
            static constexpr bool value = true;
        };

        template<size_t n, size_t i>
        struct _is_prime<n, i, std::enable_if_t<(
            n % (i+2) == 0 &&
            n >= 9 &&
            n % 3 != 0 &&
            n % 2 != 0 &&
            i * i <= n)» {
            static constexpr bool value = true;
        };

        template<size_t n, size_t i>
        struct _is_prime<n, i, std::enable_if_t<(
                n % (i+2) != 0 &&
                n % i != 0 &&
                n >= 9 &&
                n % 3 != 0 &&
                n % 2 != 0 &&
                (i * i <= n))» {
            static constexpr bool value = _is_prime<n, i+6>::value;
        };

    }  // namespace internal

    template<size_t n>
    struct is_prime {
        static constexpr bool value = internal::_is_prime<n, 5>::value;
    };

    template<size_t n>
    static constexpr bool is_prime_v = is_prime<n>::value;

    namespace internal {
        template <std::size_t... Is>
        constexpr auto index_sequence_reverse(std::index_sequence<Is...> const&)
            -> decltype(std::index_sequence<sizeof...(Is) - 1U - Is...>{});

        template <std::size_t N>
        using make_index_sequence_reverse
            = decltype(index_sequence_reverse(std::make_index_sequence<N>{}));

        template<typename Ring, typename E = void>
        struct gcd;

        template<typename Ring>
        struct gcd<Ring, std::enable_if_t<Ring::is_euclidean_domain» {
            template<typename A, typename B, typename E = void>
            struct gcd_helper {};

            // B = 0, A > 0
            template<typename A, typename B>
            struct gcd_helper<A, B, std::enable_if_t<
                ((B::is_zero_t::value) &&
                    (Ring::template gt_t<A, typename Ring::zero>::value))» {
                using type = A;
            };

            // B = 0, A < 0
            template<typename A, typename B>
            struct gcd_helper<A, B, std::enable_if_t<
                ((B::is_zero_t::value) &&
                    !(Ring::template gt_t<A, typename Ring::zero>::value))» {
                using type = typename Ring::template sub_t<typename Ring::zero, A>;
            };

            // B != 0
            template<typename A, typename B>
            struct gcd_helper<A, B, std::enable_if_t<
                (!B::is_zero_t::value)
                » {
            private: // NOLINT
                // A / B
                using k = typename Ring::template div_t<A, B>;
                // A - (A/B)*B = A % B
                using m = typename Ring::template sub_t<A, typename Ring::template mul_t<k, B»;

            public:
                using type = typename gcd_helper<B, m>::type;
            };

            template<typename A, typename B>
            using type = typename gcd_helper<A, B>::type;
        };
    }  // namespace internal

    template<typename T, typename A, typename B>
    using gcd_t = typename internal::gcd<T>::template type<A, B>;

    template<typename val>
    requires IsEuclideanDomain<typename val::ring_type>
    using abs_t = std::conditional_t<
                    val::ring_type::template pos_v<val>,
                    val, typename val::ring_type::template sub_t<typename val::ring_type::zero, val»;
}  // namespace aerobus

namespace aerobus {
    template<typename Ring, typename X>
    requires IsRing<Ring>
    struct Quotient {
        template <typename V>
        struct val {
         public:
            using type = abs_t<typename Ring::template mod_t<V, X>>;
        };

        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, typename v2>
        static constexpr bool eq_v = Ring::template eq_t<typename v1::type, typename v2::type>::value;

        template<typename v1>
        using pos_t = std::true_type;

        template<typename v>
        static constexpr bool pos_v = pos_t<v>::value;

        static constexpr bool is_euclidean_domain = true;

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

        template<typename v>
        using inject_ring_t = val<v>;
    };
}  // namespace aerobus

// type_list
namespace aerobus {
    template <typename... Ts>
    struct type_list;

    namespace internal {
        template <typename T, typename... Us>
        struct pop_front_h {
            using tail = type_list<Us...>;
            using head = T;
        };

        template <size_t index, typename L1, typename L2>
        struct split_h {
         private:
            static_assert(index <= L2::length, "index ouf of bounds");
            using a = typename L2::pop_front::type;
            using b = typename L2::pop_front::tail;
            using c = typename L1::template push_back<a>;

         public:
            using head = typename split_h<index - 1, c, b>::head;
            using tail = typename split_h<index - 1, c, b>::tail;
        };

        template <typename L1, typename L2>
        struct split_h<0, L1, L2> {
            using head = L1;
            using tail = L2;
        };

        template <size_t index, typename L, typename T>
        struct insert_h {
            static_assert(index <= L::length, "index ouf of bounds");
            using s = typename L::template split<index>;
            using left = typename s::head;
            using right = typename s::tail;
            using ll = typename left::template push_back<T>;
            using type = typename ll::template concat<right>;
        };

        template <size_t index, typename L>
        struct remove_h {
            using s = typename L::template split<index>;
            using left = typename s::head;
            using right = typename s::tail;
            using rr = typename right::pop_front::tail;
            using type = typename left::template concat<rr>;
        };
    }  // namespace internal

    template <typename... Ts>
    struct type_list {
     private:
        template <typename T>
        struct concat_h;

        template <typename... Us>
        struct concat_h<type_list<Us...» {
            using type = type_list<Ts..., Us...>;
        };

     public:
        static constexpr size_t length = sizeof...(Ts);

        template <typename T>
        using push_front = type_list<T, Ts...>;

        template <size_t index>
        using at = internal::type_at_t<index, Ts...>;

        struct pop_front {
            using type = typename internal::pop_front_h<Ts...>::head;
            using tail = typename internal::pop_front_h<Ts...>::tail;
        };

        template <typename T>
        using push_back = type_list<Ts..., T>;

        template <typename U>
        using concat = typename concat_h<U>::type;

        template <size_t index>
        struct split {
         private:
            using inner = internal::split_h<index, type_list<>, type_list<Ts...»;

         public:
            using head = typename inner::head;
            using tail = typename inner::tail;
        };

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

        template <size_t index>
        using remove = typename internal::remove_h<index, type_list<Ts...»::type;
    };

    template <>
    struct type_list<> {
        static constexpr size_t length = 0;

        template <typename T>
        using push_front = type_list<T>;

        template <typename T>
        using push_back = type_list<T>;

        template <typename U>
        using concat = U;

        // TODO(jewave): assert index == 0
        template <typename T, size_t index>
        using insert = type_list<T>;
    };
}  // namespace aerobus

// i32
namespace aerobus {
    struct i32 {
        using inner_type = int32_t;
        template<int32_t x>
        struct val {
            using ring_type = i32;
            static constexpr int32_t v = x;

            template<typename valueType>
            static constexpr valueType get() { return static_cast<valueType>(x); }

            using is_zero_t = std::bool_constant<x == 0>;

            static std::string to_string() {
                return std::to_string(x);
            }

            template<typename valueRing>
            static constexpr valueRing eval(const valueRing& v) {
                return static_cast<valueRing>(x);
            }
        };

        using zero = val<0>;
        using one = val<1>;
        static constexpr bool is_field = false;
        static constexpr bool is_euclidean_domain = true;
        template<auto x>
        using inject_constant_t = val<static_cast<int32_t>(x)>;

        template<typename v>
        using inject_ring_t = v;

     private:
        template<typename v1, typename v2>
        struct add {
            using type = val<v1::v + v2::v>;
        };

        template<typename v1, typename v2>
        struct sub {
            using type = val<v1::v - v2::v>;
        };

        template<typename v1, typename v2>
        struct mul {
            using type = val<v1::v* v2::v>;
        };

        template<typename v1, typename v2>
        struct div {
            using type = val<v1::v / v2::v>;
        };

        template<typename v1, typename v2>
        struct remainder {
            using type = val<v1::v % v2::v>;
        };

        template<typename v1, typename v2>
        struct gt {
            using type = std::conditional_t<(v1::v > v2::v), std::true_type, std::false_type>;
        };

        template<typename v1, typename v2>
        struct lt {
            using type = std::conditional_t<(v1::v < v2::v), std::true_type, std::false_type>;
        };

        template<typename v1, typename v2>
        struct eq {
            using type = std::conditional_t<(v1::v == v2::v), std::true_type, std::false_type>;
        };

        template<typename v1>
        struct pos {
            using type = std::bool_constant<(v1::v > 0)>;
        };

     public:
        template<typename v1, typename v2>
        using add_t = typename add<v1, v2>::type;

        template<typename v1, typename v2>
        using sub_t = typename sub<v1, v2>::type;

        template<typename v1, typename v2>
        using mul_t = typename mul<v1, v2>::type;

        template<typename v1, typename v2>
        using div_t = typename div<v1, v2>::type;

        template<typename v1, typename v2>
        using mod_t = typename remainder<v1, v2>::type;

        template<typename v1, typename v2>
        using gt_t = typename gt<v1, v2>::type;

        template<typename v1, typename v2>
        using lt_t = typename lt<v1, v2>::type;

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

        template<typename v1, typename v2>
        static constexpr bool eq_v = eq_t<v1, v2>::value;

        template<typename v1, typename v2>
        using gcd_t = gcd_t<i32, v1, v2>;

        template<typename v>
        using pos_t = typename pos<v>::type;

        template<typename v>
        static constexpr bool pos_v = pos_t<v>::value;
    };
}  // namespace aerobus

// i64
namespace aerobus {
    struct i64 {
        using inner_type = int64_t;
        template<int64_t x>
        struct val {
            using ring_type = i64;
            static constexpr int64_t v = x;

            template<typename valueType>
            static constexpr valueType get() { return static_cast<valueType>(x); }

            using is_zero_t = std::bool_constant<x == 0>;

            static std::string to_string() {
                return std::to_string(x);
            }

            template<typename valueRing>
            static constexpr valueRing eval(const valueRing& v) {
                return static_cast<valueRing>(x);
            }
        };

        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>;
        using one = val<1>;
        static constexpr bool is_field = false;
        static constexpr bool is_euclidean_domain = true;

     private:
        template<typename v1, typename v2>
        struct add {
            using type = val<v1::v + v2::v>;
        };

        template<typename v1, typename v2>
        struct sub {
            using type = val<v1::v - v2::v>;
        };

        template<typename v1, typename v2>
        struct mul {
            using type = val<v1::v* v2::v>;
        };

        template<typename v1, typename v2>
        struct div {
            using type = val<v1::v / v2::v>;
        };

        template<typename v1, typename v2>
        struct remainder {
            using type = val<v1::v% v2::v>;
        };

        template<typename v1, typename v2>
        struct gt {
            using type = std::conditional_t<(v1::v > v2::v), std::true_type, std::false_type>;
        };

        template<typename v1, typename v2>
        struct lt {
            using type = std::conditional_t<(v1::v < v2::v), std::true_type, std::false_type>;
        };

        template<typename v1, typename v2>
        struct eq {
            using type = std::conditional_t<(v1::v == v2::v), std::true_type, std::false_type>;
        };

        template<typename v>
        struct pos {
            using type = std::bool_constant<(v::v > 0)>;
        };

     public:
        template<typename v1, typename v2>
        using add_t = typename add<v1, v2>::type;

        template<typename v1, typename v2>
        using sub_t = typename sub<v1, v2>::type;

        template<typename v1, typename v2>
        using mul_t = typename mul<v1, v2>::type;

        template<typename v1, typename v2>
        using div_t = typename div<v1, v2>::type;

        template<typename v1, typename v2>
        using mod_t = typename remainder<v1, v2>::type;

        template<typename v1, typename v2>
        using gt_t = typename gt<v1, v2>::type;

        template<typename v1, typename v2>
        static constexpr bool gt_v = gt_t<v1, v2>::value;

        template<typename v1, typename v2>
        using lt_t = typename lt<v1, v2>::type;

        template<typename v1, typename v2>
        static constexpr bool lt_v = lt_t<v1, v2>::value;

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

        template<typename v1, typename v2>
        static constexpr bool eq_v = eq_t<v1, v2>::value;

        template<typename v1, typename v2>
        using gcd_t = gcd_t<i64, v1, v2>;

        template<typename v>
        using pos_t = typename pos<v>::type;

        template<typename v>
        static constexpr bool pos_v = pos_t<v>::value;
    };
}  // namespace aerobus

// z/pz
namespace aerobus {
    template<int32_t p>
    struct zpz {
        using inner_type = int32_t;
        template<int32_t x>
        struct val {
            using ring_type = zpz<p>;
            static constexpr int32_t v = x % p;

            template<typename valueType>
            static constexpr valueType get() { return static_cast<valueType>(x % p); }
```

```
           using is_zero_t = std::bool_constant<x% p == 0>;
           static std::string to_string() {
               return std::to_string(x % p);
           }

           template<typename valueRing>
           static constexpr valueRing eval(const valueRing& v) {
               return static_cast<valueRing>(x % p);
           }
       };

       template<auto x>
       using inject_constant_t = val<static_cast<int32_t>(x)>;

       using zero = val<0>;
       using one = val<1>;
       static constexpr bool is_field = is_prime<p>::value;
       static constexpr bool is_euclidean_domain = true;

    private:
       template<typename v1, typename v2>
       struct add {
           using type = val<(v1::v + v2::v) % p>;
       };

       template<typename v1, typename v2>
       struct sub {
           using type = val<(v1::v - v2::v) % p>;
       };

       template<typename v1, typename v2>
       struct mul {
           using type = val<(v1::v* v2::v) % p>;
       };

       template<typename v1, typename v2>
       struct div {
           using type = val<(v1::v% p) / (v2::v % p)>;
       };

       template<typename v1, typename v2>
       struct remainder {
           using type = val<(v1::v% v2::v) % p>;
       };

       template<typename v1, typename v2>
       struct gt {
           using type = std::conditional_t<(v1::v% p > v2::v% p), std::true_type, std::false_type>;
       };

       template<typename v1, typename v2>
       struct lt {
           using type = std::conditional_t<(v1::v% p < v2::v% p), std::true_type, std::false_type>;
       };

       template<typename v1, typename v2>
       struct eq {
           using type = std::conditional_t<(v1::v% p == v2::v % p), std::true_type, std::false_type>;
       };

       template<typename v1>
       struct pos {
           using type = std::bool_constant<(v1::v > 0)>;
       };

    public:
       template<typename v1, typename v2>
       using add_t = typename add<v1, v2>::type;

       template<typename v1, typename v2>
       using sub_t = typename sub<v1, v2>::type;

       template<typename v1, typename v2>
       using mul_t = typename mul<v1, v2>::type;

       template<typename v1, typename v2>
       using div_t = typename div<v1, v2>::type;

       template<typename v1, typename v2>
       using mod_t = typename remainder<v1, v2>::type;

       template<typename v1, typename v2>
       using gt_t = typename gt<v1, v2>::type;

       template<typename v1, typename v2>
       static constexpr bool gt_v = gt_t<v1, v2>::value;
```

```

        template<typename v1, typename v2>
        using lt_t = typename lt<v1, v2>::type;

        template<typename v1, typename v2>
        static constexpr bool lt_v = lt_t<v1, v2>::value;

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

        template<typename v1, typename v2>
        static constexpr bool eq_v = eq_t<v1, v2>::value;

        template<typename v1, typename v2>
        using gcd_t = gcd_t<i32, v1, v2>;

        template<typename v1>
        using pos_t = typename pos<v1>::type;

        template<typename v>
        static constexpr bool pos_v = pos_t<v>::value;
    };
}  // namespace aerobus

// polynomial
namespace aerobus {
    // coeffN x^N + ...
    template<typename Ring>
    requires IsEuclideanDomain<Ring>
    struct polynomial {
        static constexpr bool is_field = false;
        static constexpr bool is_euclidean_domain = Ring::is_euclidean_domain;

        template<typename coeffN, typename... coeffs>
        struct val {
            using ring_type = polynomial<Ring>;
            static constexpr size_t degree = sizeof...(coeffs);
            using aN = coeffN;
            using strip = val<coeffs...>;
            using is_zero_t = std::bool_constant<(degree == 0) && (aN::is_zero_t::value)>;
            static constexpr bool is_zero_v = is_zero_t::value;

         private:
            template<size_t index, typename E = void>
            struct coeff_at {};

            template<size_t index>
            struct coeff_at<index, std::enable_if_t<(index >= 0 && index <= sizeof...(coeffs))» {
                using type = internal::type_at_t<sizeof...(coeffs) - index, coeffN, coeffs...>;
            };

            template<size_t index>
            struct coeff_at<index, std::enable_if_t<(index < 0 || index > sizeof...(coeffs))» {
                using type = typename Ring::zero;
            };

         public:
            template<size_t index>
            using coeff_at_t = typename coeff_at<index>::type;

            static std::string to_string() {
                return string_helper<coeffN, coeffs...>::func();
            }

            template<typename valueRing>
            static constexpr valueRing eval(const valueRing& x) {
                return horner_evaluation<valueRing, val>
                        ::template inner<0, degree + 1>
                        ::func(static_cast<valueRing>(0), x);
            }
        };

        template<typename coeffN>
        struct val<coeffN> {
            using ring_type = polynomial<Ring>;
            static constexpr size_t degree = 0;
            using aN = coeffN;
            using strip = val<coeffN>;
            using is_zero_t = std::bool_constant<aN::is_zero_t::value>;

            static constexpr bool is_zero_v = is_zero_t::value;

            template<size_t index, typename E = void>
            struct coeff_at {};

            template<size_t index>
            struct coeff_at<index, std::enable_if_t<(index == 0)» {
```

```
            using type = aN;
        };

        template<size_t index>
        struct coeff_at<index, std::enable_if_t<(index < 0 || index > 0)» {
            using type = typename Ring::zero;
        };

        template<size_t index>
        using coeff_at_t = typename coeff_at<index>::type;

        static std::string to_string() {
            return string_helper<coeffN>::func();
        }

        template<typename valueRing>
        static constexpr valueRing eval(const valueRing& x) {
            return static_cast<valueRing>(aN::template get<valueRing>());
        }
    };

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

 private:
    template<typename P, typename E = void>
    struct simplify;

    template <typename P1, typename P2, typename I>
    struct add_low;

    template<typename P1, typename P2>
    struct add {
        using type = typename simplify<typename add_low<
        P1,
        P2,
        internal::make_index_sequence_reverse<
        std::max(P1::degree, P2::degree) + 1
        »::type>::type;
    };

    template <typename P1, typename P2, typename I>
    struct sub_low;

    template <typename P1, typename P2, typename I>
    struct mul_low;

    template<typename v1, typename v2>
    struct mul {
            using type = typename mul_low<
                v1,
                v2,
                internal::make_index_sequence_reverse<
                v1::degree + v2::degree + 1
                »::type;
    };

    template<typename coeff, size_t deg>
    struct monomial;

    template<typename v, typename E = void>
    struct derive_helper {};

    template<typename v>
    struct derive_helper<v, std::enable_if_t<v::degree == 0» {
        using type = zero;
    };

    template<typename v>
    struct derive_helper<v, std::enable_if_t<v::degree != 0» {
        using type = typename add<
            typename derive_helper<typename simplify<typename v::strip>::type>::type,
            typename monomial<
                typename Ring::template mul_t<
                    typename v::aN,
                    typename Ring::template inject_constant_t<(v::degree)>
                >,
                v::degree - 1
            >::type
        >::type;
    };

    template<typename v1, typename v2, typename E = void>
    struct eq_helper {};

    template<typename v1, typename v2>
```

```
        struct eq_helper<v1, v2, std::enable_if_t<v1::degree != v2::degree» {
            using type = std::false_type;
        };


        template<typename v1, typename v2>
        struct eq_helper<v1, v2, std::enable_if_t<
            v1::degree == v2::degree &&
            (v1::degree != 0 || v2::degree != 0) &&
            std::is_same<
            typename Ring::template eq_t<typename v1::aN, typename v2::aN>,
            std::false_type
            >::value
        >
        > {
            using type = std::false_type;
        };

        template<typename v1, typename v2>
        struct eq_helper<v1, v2, std::enable_if_t<
            v1::degree == v2::degree &&
            (v1::degree != 0 || v2::degree != 0) &&
            std::is_same<
            typename Ring::template eq_t<typename v1::aN, typename v2::aN>,
            std::true_type
            >::value
        » {
            using type = typename eq_helper<typename v1::strip, typename v2::strip>::type;
        };

        template<typename v1, typename v2>
        struct eq_helper<v1, v2, std::enable_if_t<
            v1::degree == v2::degree &&
            (v1::degree == 0)
        » {
            using type = typename Ring::template eq_t<typename v1::aN, typename v2::aN>;
        };

        template<typename v1, typename v2, typename E = void>
        struct lt_helper {};

        template<typename v1, typename v2>
        struct lt_helper<v1, v2, std::enable_if_t<(v1::degree < v2::degree)» {
            using type = std::true_type;
        };

        template<typename v1, typename v2>
        struct lt_helper<v1, v2, std::enable_if_t<(v1::degree == v2::degree)» {
            using type = typename Ring::template lt_t<typename v1::aN, typename v2::aN>;
        };

        template<typename v1, typename v2>
        struct lt_helper<v1, v2, std::enable_if_t<(v1::degree > v2::degree)» {
            using type = std::false_type;
        };

        template<typename v1, typename v2, typename E = void>
        struct gt_helper {};

        template<typename v1, typename v2>
        struct gt_helper<v1, v2, std::enable_if_t<(v1::degree > v2::degree)» {
            using type = std::true_type;
        };

        template<typename v1, typename v2>
        struct gt_helper<v1, v2, std::enable_if_t<(v1::degree == v2::degree)» {
            using type = std::false_type;
        };

        template<typename v1, typename v2>
        struct gt_helper<v1, v2, std::enable_if_t<(v1::degree < v2::degree)» {
            using type = std::false_type;
        };

        // when high power is zero : strip
        template<typename P>
        struct simplify<P, std::enable_if_t<
            std::is_same<
            typename Ring::zero,
            typename P::aN
            >::value && (P::degree > 0)
        » {
            using type = typename simplify<typename P::strip>::type;
        };

        // otherwise : do nothing
        template<typename P>
        struct simplify<P, std::enable_if_t<
            !std::is_same<
            typename Ring::zero,
            typename P::aN
            >::value && (P::degree > 0)
        » {
            using type = P;
        };

        // do not simplify constants
        template<typename P>
        struct simplify<P, std::enable_if_t<P::degree == 0» {
            using type = P;
        };

        // addition at
        template<typename P1, typename P2, size_t index>
        struct add_at {
            using type =
                typename Ring::template add_t<
                    typename P1::template coeff_at_t<index>,
                    typename P2::template coeff_at_t<index>>;
        };

        template<typename P1, typename P2, size_t index>
        using add_at_t = typename add_at<P1, P2, index>::type;

        template<typename P1, typename P2, std::size_t... I>
        struct add_low<P1, P2, std::index_sequence<I...» {
            using type = val<add_at_t<P1, P2, I>...>;
        };

        // substraction at
        template<typename P1, typename P2, size_t index>
        struct sub_at {
            using type =
                typename Ring::template sub_t<
                    typename P1::template coeff_at_t<index>,
                    typename P2::template coeff_at_t<index>>;
        };

        template<typename P1, typename P2, size_t index>
        using sub_at_t = typename sub_at<P1, P2, index>::type;

        template<typename P1, typename P2, std::size_t... I>
        struct sub_low<P1, P2, std::index_sequence<I...» {
            using type = val<sub_at_t<P1, P2, I>...>;
        };

        template<typename P1, typename P2>
        struct sub {
            using type = typename simplify<typename sub_low<
            P1,
            P2,
            internal::make_index_sequence_reverse<
            std::max(P1::degree, P2::degree) + 1
            »::type>::type;
        };

        // multiplication at
        template<typename v1, typename v2, size_t k, size_t index, size_t stop>
        struct mul_at_loop_helper {
            using type = typename Ring::template add_t<
                typename Ring::template mul_t<
                typename v1::template coeff_at_t<index>,
                typename v2::template coeff_at_t<k - index>
                >,
                typename mul_at_loop_helper<v1, v2, k, index + 1, stop>::type
            >;
        };

        template<typename v1, typename v2, size_t k, size_t stop>
        struct mul_at_loop_helper<v1, v2, k, stop, stop> {
            using type = typename Ring::template mul_t<
                typename v1::template coeff_at_t<stop>,
                typename v2::template coeff_at_t<0>>;
        };

        template <typename v1, typename v2, size_t k, typename E = void>
        struct mul_at {};

        template<typename v1, typename v2, size_t k>
        struct mul_at<v1, v2, k, std::enable_if_t<(k < 0) || (k > v1::degree + v2::degree)» {
            using type = typename Ring::zero;
        };

        template<typename v1, typename v2, size_t k>
```

```
    struct mul_at<v1, v2, k, std::enable_if_t<(k >= 0) && (k <= v1::degree + v2::degree)>> {
        using type = typename mul_at_loop_helper<v1, v2, k, 0, k>::type;
    };

    template<typename P1, typename P2, size_t index>
    using mul_at_t = typename mul_at<P1, P2, index>::type;

    template<typename P1, typename P2, std::size_t... I>
    struct mul_low<P1, P2, std::index_sequence<I...>> {
        using type = val<mul_at_t<P1, P2, I>...>;
    };

    // division helper
    template< typename A, typename B, typename Q, typename R, typename E = void>
    struct div_helper {};

    template<typename A, typename B, typename Q, typename R>
    struct div_helper<A, B, Q, R, std::enable_if_t<
        (R::degree < B::degree) ||
        (R::degree == 0 && std::is_same<typename R::aN, typename Ring::zero>::value)>> {
        using q_type = Q;
        using mod_type = R;
        using gcd_type = B;
    };

    template<typename A, typename B, typename Q, typename R>
    struct div_helper<A, B, Q, R, std::enable_if_t<
        (R::degree >= B::degree) &&
        !(R::degree == 0 && std::is_same<typename R::aN, typename Ring::zero>::value)>> {
     private:  // NOLINT
        using rN = typename R::aN;
        using bN = typename B::aN;
        using pT = typename monomial<typename Ring::template div_t<rN, bN>, R::degree - B::degree>::type;
        using rr = typename sub<R, typename mul<pT, B>::type>::type;
        using qq = typename add<Q, pT>::type;

     public:
        using q_type = typename div_helper<A, B, qq, rr>::q_type;
        using mod_type = typename div_helper<A, B, qq, rr>::mod_type;
        using gcd_type = rr;
    };

    template<typename A, typename B>
    struct div {
        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;
        using m_type = typename div_helper<A, B, zero, A>::mod_type;
    };

    template<typename P>
    struct make_unit {
        using type = typename div<P, val<typename P::aN>>::q_type;
    };

    template<typename coeff, size_t deg>
    struct monomial {
        using type = typename mul<X, typename monomial<coeff, deg - 1>::type>::type;
    };

    template<typename coeff>
    struct monomial<coeff, 0> {
        using type = val<coeff>;
    };

    template<typename valueRing, typename P>
    struct horner_evaluation {
        template<size_t index, size_t stop>
        struct inner {
            static constexpr valueRing func(const valueRing& accum, const valueRing& x) {
                constexpr valueRing coeff =
                    static_cast<valueRing>(P::template coeff_at_t<P::degree - index>::template get<valueRing>());
                return horner_evaluation<valueRing, P>::template inner<index + 1, stop>::func(x * accum + coeff, x);
            }
        };

        template<size_t stop>
        struct inner<stop, stop> {
            static constexpr valueRing func(const valueRing& accum, const valueRing& x) {
                return accum;
            }
        };
    };

    template<typename coeff, typename... coeffs>
```

```
   struct string_helper {
       static std::string func() {
           std::string tail = string_helper<coeffs...>::func();
           std::string result = "";
           if (Ring::template eq_t<coeff, typename Ring::zero>::value) {
               return tail;
           } else if (Ring::template eq_t<coeff, typename Ring::one>::value) {
               if (sizeof...(coeffs) == 1) {
                   result += "x";
               } else {
                   result += "x^" + std::to_string(sizeof...(coeffs));
               }
           } else {
               if (sizeof...(coeffs) == 1) {
                   result += coeff::to_string() + " x";
               } else {
                   result += coeff::to_string()
                           + " x^" + std::to_string(sizeof...(coeffs));
               }
           }

           if (!tail.empty()) {
               result += " + " + tail;
           }

           return result;
       }
   };

   template<typename coeff>
   struct string_helper<coeff> {
       static std::string func() {
           if (!std::is_same<coeff, typename Ring::zero>::value) {
               return coeff::to_string();
           } else {
               return "";
           }
       }
   };

public:
   template<typename P>
   using simplify_t = typename simplify<P>::type;

   template<typename v1, typename v2>
   using add_t = typename add<v1, v2>::type;

   template<typename v1, typename v2>
   using sub_t = typename sub<v1, v2>::type;

   template<typename v1, typename v2>
   using mul_t = typename mul<v1, v2>::type;

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

   template<typename v1, typename v2>
   using lt_t = typename lt_helper<v1, v2>::type;

   template<typename v1, typename v2>
   using gt_t = typename gt_helper<v1, v2>::type;

   template<typename v1, typename v2>
   using div_t = typename div<v1, v2>::q_type;

   template<typename v1, typename v2>
   using mod_t = typename div_helper<v1, v2, zero, v1>::mod_type;

   template<typename coeff, size_t deg>
   using monomial_t = typename monomial<coeff, deg>::type;

   template<typename v>
   using derive_t = typename derive_helper<v>::type;

   template<typename v>
   using pos_t = typename Ring::template pos_t<typename v::aN>;

   template<typename v>
   static constexpr bool pos_v = pos_t<v>::value;

   template<typename v1, typename v2>
   using gcd_t = std::conditional_t<
       Ring::is_euclidean_domain,
       typename make_unit<gcd_t<polynomial<Ring>, v1, v2»::type,
       void>;

   template<auto x>
```

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

        template<typename v>
        using inject_ring_t = val<v>;
    };
}  // namespace aerobus

// fraction field
namespace aerobus {
    namespace internal {
        template<typename Ring, typename E = void>
        requires IsEuclideanDomain<Ring>
        struct _FractionField {};

        template<typename Ring>
        requires IsEuclideanDomain<Ring>
        struct _FractionField<Ring, std::enable_if_t<Ring::is_euclidean_domain» {
            static constexpr bool is_field = true;
            static constexpr bool is_euclidean_domain = true;

         private:
            template<typename val1, typename val2, typename E = void>
            struct to_string_helper {};

            template<typename val1, typename val2>
            struct to_string_helper <val1, val2,
                std::enable_if_t<
                Ring::template eq_t<
                val2, typename Ring::one
                >::value
                >
            > {
                static std::string func() {
                    return val1::to_string();
                }
            };

            template<typename val1, typename val2>
            struct to_string_helper<val1, val2,
                std::enable_if_t<
                !Ring::template eq_t<
                val2,
                typename Ring::one
                >::value
                >
            > {
                static std::string func() {
                    return "(" + val1::to_string() + ") / (" + val2::to_string() + ")";
                }
            };

         public:
            template<typename val1, typename val2>
            struct val {
                using x = val1;
                using y = val2;
                using is_zero_t = typename val1::is_zero_t;
                static constexpr bool is_zero_v = val1::is_zero_t::value;

                using ring_type = Ring;
                using field_type = _FractionField<Ring>;

                 static constexpr bool is_integer = std::is_same_v<val2, typename Ring::one>;

                template<typename valueType>
                static constexpr valueType get() { return static_cast<valueType>(x::v) / static_cast<valueType>(y::v); }

                static std::string to_string() {
                    return to_string_helper<val1, val2>::func();
                }

                template<typename valueRing>
                static constexpr valueRing eval(const valueRing& v) {
                    return x::eval(v) / y::eval(v);
                }
            };

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

            template<typename v>
            using inject_t = val<v, typename Ring::one>;

            template<auto x>
            using inject_constant_t = val<typename Ring::template inject_constant_t<x>, typename Ring::one>;

            template<typename v>
            using inject_ring_t = val<typename Ring::template inject_ring_t<v>, typename Ring::one>;

            using ring_type = Ring;

         private:
            template<typename v, typename E = void>
            struct simplify {};

            // x = 0
            template<typename v>
            struct simplify<v, std::enable_if_t<v::x::is_zero_t::value» {
                using type = typename _FractionField<Ring>::zero;
            };

            // x != 0
            template<typename v>
            struct simplify<v, std::enable_if_t<!v::x::is_zero_t::value» {
             private:
                using _gcd = typename Ring::template gcd_t<typename v::x, typename v::y>;
                using newx = typename Ring::template div_t<typename v::x, _gcd>;
                using newy = typename Ring::template div_t<typename v::y, _gcd>;

                using posx = std::conditional_t<
                                    !Ring::template pos_v<newy>,
                                    typename Ring::template sub_t<typename Ring::zero, newx>,
                                    newx>;
                using posy = std::conditional_t<
                                    !Ring::template pos_v<newy>,
                                    typename Ring::template sub_t<typename Ring::zero, newy>,
                                    newy>;
             public:
                using type = typename _FractionField<Ring>::template val<posx, posy>;
            };

         public:
            template<typename v>
            using simplify_t = typename simplify<v>::type;

         private:
            template<typename v1, typename v2>
            struct add {
             private:
                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>;
                using dividend = typename Ring::template add_t<a, b>;
                using diviser = typename Ring::template mul_t<typename v1::y, typename v2::y>;
                using g = typename Ring::template gcd_t<dividend, diviser>;

             public:
                using type = typename _FractionField<Ring>::template simplify_t<val<dividend,
diviser»;
            };

            template<typename v>
            struct pos {
                using type = std::conditional_t<
                    (Ring::template pos_v<typename v::x> && Ring::template pos_v<typename v::y>) ||
                    (!Ring::template pos_v<typename v::x> && !Ring::template pos_v<typename v::y>),
                    std::true_type,
                    std::false_type>;
            };

            template<typename v1, typename v2>
            struct sub {
             private:
                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>;
                using dividend = typename Ring::template sub_t<a, b>;
                using diviser = typename Ring::template mul_t<typename v1::y, typename v2::y>;
                using g = typename Ring::template gcd_t<dividend, diviser>;

             public:
                using type = typename _FractionField<Ring>::template simplify_t<val<dividend,
diviser»;
            };

            template<typename v1, typename v2>
            struct mul {
             private:
                using a = typename Ring::template mul_t<typename v1::x, typename v2::x>;
                using b = typename Ring::template mul_t<typename v1::y, typename v2::y>;

             public:
                using type = typename _FractionField<Ring>::template simplify_t<val<a, b»;
            };

            template<typename v1, typename v2, typename E = void>
            struct div {};

            template<typename v1, typename v2>
            struct div<v1, v2, std::enable_if_t<!std::is_same<v2, typename
_FractionField<Ring>::zero>::value»  {
             private:
                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>;

             public:
                using type = typename _FractionField<Ring>::template simplify_t<val<a, b»;
            };

            template<typename v1, typename v2>
            struct div<v1, v2, std::enable_if_t<
                std::is_same<zero, v1>::value && std::is_same<v2, zero>::value»  {
                using type = one;
            };

            template<typename v1, typename v2>
            struct eq {
                using type = std::conditional_t<
                        std::is_same<typename simplify_t<v1>::x, typename simplify_t<v2>::x>::value &&
                        std::is_same<typename simplify_t<v1>::y, typename simplify_t<v2>::y>::value,
                    std::true_type,
                    std::false_type>;
            };

            template<typename TL, typename E = void>
            struct vadd {};

            template<typename TL>
            struct vadd<TL, std::enable_if_t<(TL::length > 1)» {
                using head = typename TL::pop_front::type;
                using tail = typename TL::pop_front::tail;
                using type = typename add<head, typename vadd<tail>::type>::type;
            };

            template<typename TL>
            struct vadd<TL, std::enable_if_t<(TL::length == 1)» {
                using type = typename TL::template at<0>;
            };

            template<typename... vals>
            struct vmul {};

            template<typename v1, typename... vals>
            struct vmul<v1, vals...> {
                using type = typename mul<v1, typename vmul<vals...>::type>::type;
            };

            template<typename v1>
            struct vmul<v1> {
                using type = v1;
            };


            template<typename v1, typename v2, typename E = void>
            struct gt;

            template<typename v1, typename v2>
            struct gt<v1, v2, std::enable_if_t<
                (eq<v1, v2>::type::value)
                » {
                using type = std::false_type;
            };

            template<typename v1, typename v2>
            struct gt<v1, v2, std::enable_if_t<
                (!eq<v1, v2>::type::value) &&
                (!pos<v1>::type::value) && (!pos<v2>::type::value)
                » {
                using type = typename gt<
                    typename sub<zero, v1>::type, typename sub<zero, v2>::type
                >::type;
            };

            template<typename v1, typename v2>
            struct gt<v1, v2, std::enable_if_t<
                (!eq<v1, v2>::type::value) &&
                (pos<v1>::type::value) && (!pos<v2>::type::value)
                » {
                using type = std::true_type;
            };

            template<typename v1, typename v2>
            struct gt<v1, v2, std::enable_if_t<
                (!eq<v1, v2>::type::value) &&
                (!pos<v1>::type::value) && (pos<v2>::type::value)
                » {
                using type = std::false_type;
            };

            template<typename v1, typename v2>
            struct gt<v1, v2, std::enable_if_t<
                (!eq<v1, v2>::type::value) &&
                (pos<v1>::type::value) && (pos<v2>::type::value)
                » {
                using type = typename Ring::template gt_t<
                    typename Ring::template mul_t<v1::x, v2::y>,
                    typename Ring::template mul_t<v2::y, v2::x>
                >;
            };

         public:
            template<typename v1, typename v2>
            using add_t = typename add<v1, v2>::type;
            template<typename v1, typename v2>
            using mod_t = zero;
            template<typename v1, typename v2>
            using gcd_t = v1;
            template<typename... vs>
            using vadd_t = typename vadd<vs...>::type;
            template<typename... vs>
            using vmul_t = typename vmul<vs...>::type;
            template<typename v1, typename v2>
            using sub_t = typename sub<v1, v2>::type;
            template<typename v1, typename v2>
            using mul_t = typename mul<v1, v2>::type;
            template<typename v1, typename v2>
            using div_t = typename div<v1, v2>::type;
            template<typename v1, typename v2>
            using eq_t = typename eq<v1, v2>::type;
            template<typename v1, typename v2>
            static constexpr bool eq_v = eq<v1, v2>::type::value;
            template<typename v1, typename v2>
            using gt_t = typename gt<v1, v2>::type;
            template<typename v1, typename v2>
            static constexpr bool gt_v = gt<v1, v2>::type::value;
            template<typename v1>
            using pos_t = typename pos<v1>::type;
            template<typename v>
            static constexpr bool pos_v = pos_t<v>::value;
        };

        template<typename Ring, typename E = void>
        requires IsEuclideanDomain<Ring>
        struct FractionFieldImpl {};

        // fraction field of a field is the field itself
        template<typename Field>
        requires IsEuclideanDomain<Field>
        struct FractionFieldImpl<Field, std::enable_if_t<Field::is_field» {
            using type = Field;
            template<typename v>
            using inject_t = v;
        };

        // fraction field of a ring is the actual fraction field
        template<typename Ring>
        requires IsEuclideanDomain<Ring>
        struct FractionFieldImpl<Ring, std::enable_if_t<!Ring::is_field» {
            using type = _FractionField<Ring>;
        };
    }  // namespace internal

    template<typename Ring>
    requires IsEuclideanDomain<Ring>
    using FractionField = typename internal::FractionFieldImpl<Ring>::type;
}  // namespace aerobus

// short names for common types
namespace aerobus {
    using q32 = FractionField<i32>;
    using fpq32 = FractionField<polynomial<q32»;
    using q64 = FractionField<i64>;
    using pi64 = polynomial<i64>;
    using pq64 = polynomial<q64>;
    using fpq64 = FractionField<polynomial<q64»;
    template<typename Ring, typename v1, typename v2>
    using makefraction_t = typename FractionField<Ring>::template val<v1, v2>;

    template<typename Ring, typename v1, typename v2>
    using addfractions_t = typename FractionField<Ring>::template add_t<v1, v2>;
    template<typename Ring, typename v1, typename v2>
    using mulfractions_t = typename FractionField<Ring>::template mul_t<v1, v2>;
}  // namespace aerobus

// taylor series and common integers (factorial, bernouilli...) appearing in taylor coefficients
namespace aerobus {
    namespace internal {
        template<typename T, size_t x, typename E = void>
        struct factorial {};

        template<typename T, size_t x>
        struct factorial<T, x, std::enable_if_t<(x > 0)» {
        private:
            template<typename, size_t, typename>
            friend struct factorial;
        public:
            using type = typename T::template mul_t<typename T::template val<x>, typename factorial<T, x - 1>::type>;
            static constexpr typename T::inner_type value = type::template get<typename T::inner_type>();
        };

        template<typename T>
        struct factorial<T, 0> {
         public:
            using type = typename T::one;
            static constexpr typename T::inner_type value = type::template get<typename T::inner_type>();
        };
    }  // namespace internal

    template<typename T, size_t i>
    using factorial_t = typename internal::factorial<T, i>::type;

    template<typename T, size_t i>
    inline constexpr typename T::inner_type factorial_v = internal::factorial<T, i>::value;

    namespace internal {
        template<typename T, size_t k, size_t n, typename E = void>
        struct combination_helper {};

        template<typename T, size_t k, size_t n>
        struct combination_helper<T, k, n, std::enable_if_t<(n >= 0 && k <= (n / 2) && k > 0)» {
            using type = typename FractionField<T>::template mul_t<
                typename combination_helper<T, k - 1, n - 1>::type,
                makefraction_t<T, typename T::template val<n>, typename T::template val<k»>;
        };

        template<typename T, size_t k, size_t n>
        struct combination_helper<T, k, n, std::enable_if_t<(n >= 0 && k > (n / 2) && k > 0)» {
            using type = typename combination_helper<T, n - k, n>::type;
        };

        template<typename T, size_t n>
        struct combination_helper<T, 0, n> {
            using type = typename FractionField<T>::one;
        };

        template<typename T, size_t k, size_t n>
        struct combination {
            using type = typename internal::combination_helper<T, k, n>::type::x;
            static constexpr typename T::inner_type value =
                        internal::combination_helper<T, k, n>::type::template get<typename T::inner_type>();
        };
    }  // namespace internal

    template<typename T, size_t k, size_t n>
    using combination_t = typename internal::combination<T, k, n>::type;

    template<typename T, size_t k, size_t n>
    inline constexpr typename T::inner_type combination_v = internal::combination<T, k, n>::value;

    namespace internal {
        template<typename T, size_t m>
        struct bernouilli;

        template<typename T, typename accum, size_t k, size_t m>
        struct bernouilli_helper {
            using type = typename bernouilli_helper<
                T,
                addfractions_t<T,
                    accum,
                    mulfractions_t<T,
                        makefraction_t<T,
                            combination_t<T, k, m + 1>,
                            typename T::one>,
                        typename bernouilli<T, k>::type
                    >
                >,
                k + 1,
                m>::type;
        };

        template<typename T, typename accum, size_t m>
        struct bernouilli_helper<T, accum, m, m> {
            using type = accum;
        };



        template<typename T, size_t m>
        struct bernouilli {
            using type = typename FractionField<T>::template mul_t<
                typename internal::bernouilli_helper<T, typename FractionField<T>::zero, 0, m>::type,
                makefraction_t<T,
                typename T::template val<static_cast<typename T::inner_type>(-1)>,
                typename T::template val<static_cast<typename T::inner_type>(m + 1)>
                >
            >;

            template<typename floatType>
            static constexpr floatType value = type::template get<floatType>();
        };

        template<typename T>
        struct bernouilli<T, 0> {
            using type = typename FractionField<T>::one;

            template<typename floatType>
            static constexpr floatType value = type::template get<floatType>();
        };
    }  // namespace internal

    template<typename T, size_t n>
    using bernouilli_t = typename internal::bernouilli<T, n>::type;

    template<typename FloatType, typename T, size_t n >
    inline constexpr FloatType bernouilli_v = internal::bernouilli<T, n>::template value<FloatType>;

    namespace internal {
        template<typename T, int k, typename E = void>
        struct alternate {};

        template<typename T, int k>
        struct alternate<T, k, std::enable_if_t<k % 2 == 0» {
            using type = typename T::one;
            static constexpr typename T::inner_type value = type::template get<typename
T::inner_type>();
        };

        template<typename T, int k>
        struct alternate<T, k, std::enable_if_t<k % 2 != 0» {
            using type = typename T::template sub_t<typename T::zero, typename T::one>;
            static constexpr typename T::inner_type value = type::template get<typename
T::inner_type>();
        };
    }  // namespace internal

    template<typename T, int k>
    using alternate_t = typename internal::alternate<T, k>::type;

    namespace internal {
        template<typename T, int n, int k, typename E = void>
        struct stirling_helper {};

        template<typename T>
        struct stirling_helper<T, 0, 0> {
            using type = typename T::one;
        };

        template<typename T, int n>
        struct stirling_helper<T, n, 0, std::enable_if_t<(n > 0)» {
            using type = typename T::zero;
        };

        template<typename T, int n>
        struct stirling_helper<T, 0, n, std::enable_if_t<(n > 0)» {
            using type = typename T::zero;
        };

        template<typename T, int n, int k>
        struct stirling_helper<T, n, k, std::enable_if_t<(k > 0) && (n > 0)» {
            using type = typename T::template sub_t<
                            typename stirling_helper<T, n-1, k-1>::type,
                            typename T::template mul_t<
                                typename T::template inject_constant_t<n-1>,
                                typename stirling_helper<T, n-1, k>::type
                            »;
        };
    }  // namespace internal

    template<typename T, int n, int k>
    using stirling_signed_t = typename internal::stirling_helper<T, n, k>::type;

    template<typename T, int n, int k>
    using stirling_unsigned_t = abs_t<typename internal::stirling_helper<T, n, k>::type>;

    template<typename T, int n, int k>
    static constexpr typename T::inner_type stirling_signed_v = stirling_signed_t<T, n, k>::v;


    template<typename T, int n, int k>
    static constexpr typename T::inner_type stirling_unsigned_v = stirling_unsigned_t<T, n, k>::v;

    template<typename T, size_t k>
    inline constexpr typename T::inner_type alternate_v = internal::alternate<T, k>::value;

    namespace internal {
        template<typename T, auto p, auto n, typename E = void>
        struct pow {};

        template<typename T, auto p, auto n>
        struct pow<T, p, n, std::enable_if_t<(n > 0 && n % 2 == 0)» {
            using type = typename T::template mul_t<
                typename pow<T, p, n/2>::type,
                typename pow<T, p, n/2>::type
            >;
        };

        template<typename T, auto p, auto n>
        struct pow<T, p, n, std::enable_if_t<(n % 2 == 1)» {
            using type = typename T::template mul_t<
                typename T::template inject_constant_t<p>,
                typename T::template mul_t<
                    typename pow<T, p, n/2>::type,
                    typename pow<T, p, n/2>::type
                >
            >;
        };

        template<typename T, auto p>
        struct pow<T, p, 0> { using type = typename T::one; };
    }  // namespace internal

    template<typename T, auto p, auto n>
    using pow_t = typename internal::pow<T, p, n>::type;

    template<typename T, auto p, auto n>
    static constexpr typename T::inner_type pow_v = internal::pow<T, p, n>::type::v;

    namespace internal {
        template<typename, template<typename, size_t> typename, class>
        struct make_taylor_impl;

        template<typename T, template<typename, size_t> typename coeff_at, size_t... Is>
        struct make_taylor_impl<T, coeff_at, std::integer_sequence<size_t, Is...» {
            using type = typename polynomial<FractionField<T»::template val<typename coeff_at<T, Is>::type...>;
        };
    }

    template<typename T, template<typename, size_t index> typename coeff_at, size_t deg>
    using taylor = typename internal::make_taylor_impl<
        T,
        coeff_at,
        internal::make_index_sequence_reverse<deg + 1»::type;

    namespace internal {
        template<typename T, size_t i>
        struct exp_coeff {
            using type = makefraction_t<T, typename T::one, factorial_t<T, i»;
        };

        template<typename T, size_t i, typename E = void>
        struct sin_coeff_helper {};

        template<typename T, size_t i>
        struct sin_coeff_helper<T, i, std::enable_if_t<(i & 1) == 0» {
            using type = typename FractionField<T>::zero;
        };

        template<typename T, size_t i>
        struct sin_coeff_helper<T, i, std::enable_if_t<(i & 1) == 1» {
            using type = makefraction_t<T, alternate_t<T, i / 2>, factorial_t<T, i»;
        };

        template<typename T, size_t i>
        struct sin_coeff {
            using type = typename sin_coeff_helper<T, i>::type;
        };

        template<typename T, size_t i, typename E = void>
        struct sh_coeff_helper {};

        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;
        };

        template<typename T, size_t i>
        struct sh_coeff_helper<T, i, std::enable_if_t<(i & 1) == 1» {
            using type = makefraction_t<T, typename T::one, factorial_t<T, i»;
        };

        template<typename T, size_t i>
        struct sh_coeff {
            using type = typename sh_coeff_helper<T, i>::type;
        };

        template<typename T, size_t i, typename E = void>
        struct cos_coeff_helper {};

        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;
        };

        template<typename T, size_t i>
        struct cos_coeff_helper<T, i, std::enable_if_t<(i & 1) == 0» {
            using type = makefraction_t<T, alternate_t<T, i / 2>, factorial_t<T, i»;
        };

        template<typename T, size_t i>
        struct cos_coeff {
            using type = typename cos_coeff_helper<T, i>::type;
        };

        template<typename T, size_t i, typename E = void>
        struct cosh_coeff_helper {};

        template<typename T, size_t i>
        struct cosh_coeff_helper<T, i, std::enable_if_t<(i & 1) == 1» {
            using type = typename FractionField<T>::zero;
        };

        template<typename T, size_t i>
        struct cosh_coeff_helper<T, i, std::enable_if_t<(i & 1) == 0» {
            using type = makefraction_t<T, typename T::one, factorial_t<T, i»;
        };

        template<typename T, size_t i>
        struct cosh_coeff {
            using type = typename cosh_coeff_helper<T, i>::type;
        };

        template<typename T, size_t i>
        struct geom_coeff { using type = typename FractionField<T>::one; };


        template<typename T, size_t i, typename E = void>
        struct atan_coeff_helper;

        template<typename T, size_t i>
        struct atan_coeff_helper<T, i, std::enable_if_t<(i & 1) == 1» {
            using type = makefraction_t<T, alternate_t<T, i / 2>, typename T::template val<i»;
        };

        template<typename T, size_t i>
        struct atan_coeff_helper<T, i, std::enable_if_t<(i & 1) == 0» {
            using type = typename FractionField<T>::zero;
        };

        template<typename T, size_t i>
        struct atan_coeff { using type = typename atan_coeff_helper<T, i>::type; };

        template<typename T, size_t i, typename E = void>
        struct asin_coeff_helper;

        template<typename T, size_t i>
        struct asin_coeff_helper<T, i, std::enable_if_t<(i & 1) == 1» {
            using type = makefraction_t<T,
                factorial_t<T, i - 1>,
                typename T::template mul_t<
                    typename T::template val<i>,
                    T::template mul_t<
                        pow_t<T, 4, i / 2>,
                        pow<T, factorial<T, i / 2>::value, 2
                    >
                >
                »;
        };

        template<typename T, size_t i>
        struct asin_coeff_helper<T, i, std::enable_if_t<(i & 1) == 0» {
            using type = typename FractionField<T>::zero;
        };

        template<typename T, size_t i>
        struct asin_coeff {
            using type = typename asin_coeff_helper<T, i>::type;
        };

        template<typename T, size_t i>
        struct lnp1_coeff {
            using type = makefraction_t<T,
                alternate_t<T, i + 1>,
                typename T::template val<i»;
        };

        template<typename T>
        struct lnp1_coeff<T, 0> { using type = typename FractionField<T>::zero; };

        template<typename T, size_t i, typename E = void>
        struct asinh_coeff_helper;

        template<typename T, size_t i>
        struct asinh_coeff_helper<T, i, std::enable_if_t<(i & 1) == 1» {
            using type = makefraction_t<T,
                typename T::template mul_t<
                    alternate_t<T, i / 2>,
                    factorial_t<T, i - 1>
                >,
                typename T::template mul_t<
                    T::template mul_t<
                        typename T::template val<i>,
                        pow_t<T, (factorial<T, i / 2>::value), 2>
                    >,
                    pow_t<T, 4, i / 2>
                >
            >;
        };

        template<typename T, size_t i>
        struct asinh_coeff_helper<T, i, std::enable_if_t<(i & 1) == 0» {
            using type = typename FractionField<T>::zero;
        };

        template<typename T, size_t i>
        struct asinh_coeff {
            using type = typename asinh_coeff_helper<T, i>::type;
        };

        template<typename T, size_t i, typename E = void>
        struct atanh_coeff_helper;

        template<typename T, size_t i>
        struct atanh_coeff_helper<T, i, std::enable_if_t<(i & 1) == 1» {
            // 1/i
            using type = typename FractionField<T>:: template val<
                typename T::one,
                typename T::template val<static_cast<typename T::inner_type>(i)»;
        };

        template<typename T, size_t i>
        struct atanh_coeff_helper<T, i, std::enable_if_t<(i & 1) == 0» {
            using type = typename FractionField<T>::zero;
        };

        template<typename T, size_t i>
        struct atanh_coeff {
            using type = typename asinh_coeff_helper<T, i>::type;
        };

        template<typename T, size_t i, typename E = void>
        struct tan_coeff_helper;

        template<typename T, size_t i>
        struct tan_coeff_helper<T, i, std::enable_if_t<(i % 2) == 0» {
            using type = typename FractionField<T>::zero;
        };

        template<typename T, size_t i>
        struct tan_coeff_helper<T, i, std::enable_if_t<(i % 2) != 0» {
        private:
            // 4^((i+1)/2)
            using _4p = typename FractionField<T>::template inject_t<pow_t<T, 4, (i + 1) / 2»;
            // 4^((i+1)/2) - 1
            using _4pm1 = typename FractionField<T>::template sub_t<_4p, typename FractionField<T>::one>;
            // (-1)^((i-1)/2)
            using altp = typename FractionField<T>::template inject_t<alternate_t<T, (i - 1) / 2»;
            using dividend = typename FractionField<T>::template mul_t<
                altp,
                FractionField<T>::template mul_t<
                _4p,
                FractionField<T>::template mul_t<
                _4pm1,
                bernouilli_t<T, (i + 1)>
                >
                >
            >;
        public:
            using type = typename FractionField<T>::template div_t<dividend,
                typename FractionField<T>::template inject_t<factorial_t<T, i + 1»>;
        };

        template<typename T, size_t i>
        struct tan_coeff {
            using type = typename tan_coeff_helper<T, i>::type;
        };

        template<typename T, size_t i, typename E = void>
        struct tanh_coeff_helper;

        template<typename T, size_t i>
        struct tanh_coeff_helper<T, i, std::enable_if_t<(i % 2) == 0» {
            using type = typename FractionField<T>::zero;
        };

        template<typename T, size_t i>
        struct tanh_coeff_helper<T, i, std::enable_if_t<(i % 2) != 0» {
        private:
            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 FractionField<T>::one>;
            using dividend =
                typename FractionField<T>::template mul_t<
                _4p,
                typename FractionField<T>::template mul_t<
                _4pm1,
                bernouilli_t<T, (i + 1)>
                >
                >::type;
        public:
            using type = typename FractionField<T>::template div_t<dividend,
                FractionField<T>::template inject_t<factorial_t<T, i + 1»>;
        };

        template<typename T, size_t i>
        struct tanh_coeff {
            using type = typename tanh_coeff_helper<T, i>::type;
        };
    }  // namespace internal

    template<typename T, size_t deg>
    using exp = taylor<T, internal::exp_coeff, deg>;

    template<typename T, size_t deg>
    using expm1 = typename polynomial<FractionField<T»::template sub_t<
        exp<T, deg>,
        typename polynomial<FractionField<T»::one>;

    template<typename T, size_t deg>
    using lnp1 = taylor<T, internal::lnp1_coeff, deg>;

    template<typename T, size_t deg>
    using atan = taylor<T, internal::atan_coeff, deg>;

    template<typename T, size_t deg>
    using sin = taylor<T, internal::sin_coeff, deg>;

    template<typename T, size_t deg>
    using sinh = taylor<T, internal::sh_coeff, deg>;

    template<typename T, size_t deg>
    using cosh = taylor<T, internal::cosh_coeff, deg>;

    template<typename T, size_t deg>
    using cos = taylor<T, internal::cos_coeff, deg>;

    template<typename T, size_t deg>
    using geometric_sum = taylor<T, internal::geom_coeff, deg>;

    template<typename T, size_t deg>
    using asin = taylor<T, internal::asin_coeff, deg>;

    template<typename T, size_t deg>
    using asinh = taylor<T, internal::asinh_coeff, deg>;

    template<typename T, size_t deg>
    using atanh = taylor<T, internal::atanh_coeff, deg>;

    template<typename T, size_t deg>
    using tan = taylor<T, internal::tan_coeff, deg>;

    template<typename T, size_t deg>
    using tanh = taylor<T, internal::tanh_coeff, deg>;
}  // namespace aerobus

// continued fractions
namespace aerobus {
    template<int64_t... values>
    struct ContinuedFraction {};

    template<int64_t a0>
    struct ContinuedFraction<a0> {
        using type = typename q64::template inject_constant_t<a0>;
        static constexpr double val = type::template get<double>();
    };

    template<int64_t a0, int64_t... rest>
    struct ContinuedFraction<a0, rest...> {
        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 constexpr double val = type::template get<double>();
    };

    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>;
    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 =
ContinuedFraction<1, 2, 2, 2, 2, 2, 2, 2, 2, 2, 2, 2, 2, 2, 2, 2, 2, 2, 2, 2, 2, 2>;
    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>;
// NOLINT
}  // namespace aerobus

// known polynomials
namespace aerobus {
    // CChebyshev
    namespace internal {
        template<int kind, int deg>
        struct chebyshev_helper {
            using type = typename pi64::template sub_t<
                typename pi64::template mul_t<
                typename pi64::template mul_t<
                pi64::inject_constant_t<2>,
                typename pi64::X
                >,
                typename chebyshev_helper<kind, deg - 1>::type
                >,
                typename chebyshev_helper<kind, deg - 2>::type
            >;
        };

        template<>
        struct chebyshev_helper<1, 0> {
            using type = typename pi64::one;
        };

        template<>
        struct chebyshev_helper<1, 1> {
            using type = typename pi64::X;
        };

        template<>
        struct chebyshev_helper<2, 0> {
            using type = typename pi64::one;
        };

        template<>
        struct chebyshev_helper<2, 1> {
            using type = typename pi64::template mul_t<
                typename pi64::inject_constant_t<2>,
                typename pi64::X>;
        };
    }  // namespace internal

    // Laguerre
    namespace internal {
        template<size_t deg>
        struct laguerre_helper {
         private:
            // Lk = (1 / k) * ((2 * k - 1 - x) * lkm1 - (k - 2)Lkm2)
            using lnm2 = typename laguerre_helper<deg - 2>::type;
            using lnm1 = typename laguerre_helper<deg - 1>::type;
            // -x + 2k-1
            using p = typename pq64::template val<
                typename q64::template inject_constant_t<-1>,
                typename q64::template inject_constant_t<2 * deg - 1»;
            // 1/n
            using factor = typename pq64::template inject_ring_t<
                q64::val<typename i64::one, typename i64::template inject_constant_t<deg»>;

         public:
            using type = typename pq64::template mul_t <
                factor,
                typename pq64::template sub_t<
                    typename pq64::template mul_t<
                        p,
                        lnm1
                    >,
                    typename pq64::template mul_t<
                        typename pq64::template inject_constant_t<deg-1>,
                        lnm2
                    >
                >
            >;

        };

        template<>
        struct laguerre_helper<0> {
            using type = typename pq64::one;
        };

        template<>
        struct laguerre_helper<1> {
            using type = typename pq64::template sub_t<typename pq64::one, typename pq64::X>;
        };
    }  // namespace internal

    namespace known_polynomials {
        enum hermite_kind {
            probabilist,
            physicist
        };
    }

    namespace internal {
        template<size_t deg, known_polynomials::hermite_kind kind>
        struct hermite_helper {};

        template<size_t deg>
        struct hermite_helper<deg, known_polynomials::hermite_kind::probabilist> {
         private:
            using hnm1 = typename hermite_helper<deg - 1,
known_polynomials::hermite_kind::probabilist>::type;
            using hnm2 = typename hermite_helper<deg - 2,
known_polynomials::hermite_kind::probabilist>::type;

         public:
            using type = typename pi64::template sub_t<
                typename pi64::template mul_t<typename pi64::X, hnm1>,
                typename pi64::template mul_t<
                    typename pi64::template inject_constant_t<deg - 1>,
                    hnm2
                >
            >;
        };

        template<size_t deg>
        struct hermite_helper<deg, known_polynomials::hermite_kind::physicist> {
         private:
            using hnm1 = typename hermite_helper<deg - 1, known_polynomials::hermite_kind::physicist>::type;
            using hnm2 = typename hermite_helper<deg - 2, known_polynomials::hermite_kind::physicist>::type;

         public:
            using type = typename pi64::template sub_t<
                // 2X Hn-1
                typename pi64::template mul_t<
                    typename pi64::val<typename i64::template inject_constant_t<2>,
                    typename i64::zero>, hnm1>,

                typename pi64::template mul_t<
                    typename pi64::template inject_constant_t<2*(deg - 1)>,
                    hnm2
                >
            >;
        };

        template<>
        struct hermite_helper<0, known_polynomials::hermite_kind::probabilist> {
            using type = typename pi64::one;
        };

        template<>
        struct hermite_helper<1, known_polynomials::hermite_kind::probabilist> {
            using type = typename pi64::X;
        };

        template<>
        struct hermite_helper<0, known_polynomials::hermite_kind::physicist> {
            using type = typename pi64::one;
        };

        template<>
        struct hermite_helper<1, known_polynomials::hermite_kind::physicist> {
            // 2X
            using type = typename pi64::template val<typename i64::template inject_constant_t<2>, typename i64::zero>;
        };
    }  // namespace internal

    namespace known_polynomials {
        template <size_t deg>
        using chebyshev_T = typename internal::chebyshev_helper<1, deg>::type;

        template <size_t deg>
        using chebyshev_U = typename internal::chebyshev_helper<2, deg>::type;

        template <size_t deg>
        using laguerre = typename internal::laguerre_helper<deg>::type;

        template <size_t deg>
        using hermite_prob = typename internal::hermite_helper<deg, hermite_kind::probabilist>::type;

        template <size_t deg>
        using hermite_phys = typename internal::hermite_helper<deg, hermite_kind::physicist>::type;
    }  // namespace known_polynomials
}  // namespace aerobus


#ifdef AEROBUS_CONWAY_IMPORTS
template<int p, int n>
struct ConwayPolynomial;

#define ZPZV ZPZ::template val
#define POLYV aerobus::polynomial<ZPZ>::template val
template<> struct ConwayPolynomial<2, 1> { using ZPZ = aerobus::zpz<2>; using type = POLYV<ZPZV<1>, ZPZV<1»; };  // NOLINT
template<> struct ConwayPolynomial<2, 2> { using ZPZ = aerobus::zpz<2>; using type = POLYV<ZPZV<1>, ZPZV<1>, ZPZV<1»; }; // NOLINT
template<> struct ConwayPolynomial<2, 3> { using ZPZ = aerobus::zpz<2>; using type = POLYV<ZPZV<1>, ZPZV<0>, ZPZV<1>, ZPZV<1»; };  // NOLINT
template<> struct ConwayPolynomial<2, 4> { using ZPZ = aerobus::zpz<2>; using type = POLYV<ZPZV<1>, ZPZV<0>, ZPZV<0>, ZPZV<1>, ZPZV<1»; };  // NOLINT
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
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
template<> struct ConwayPolynomial<2, 7> { using ZPZ = aerobus::zpz<2>; using type = POLYV<ZPZV<1>, ZPZV<0>, ZPZV<0>, ZPZV<0>, ZPZV<0>, ZPZV<0>, ZPZV<1>, ZPZV<1»; };  // NOLINT
```

```
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»; };  // NOLINT
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<1»; };  // NOLINT
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<0>, ZPZV<1>, ZPZV<1>, ZPZV<1>, ZPZV<1»; };  // NOLINT
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<1>, ZPZV<0>, ZPZV<1»; };  // NOLINT
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<1>, ZPZV<0>, ZPZV<1>, ZPZV<1»; };  // NOLINT
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<1>, ZPZV<1>, ZPZV<0>, ZPZV<1>, ZPZV<1»; };  // NOLINT
template<> struct ConwayPolynomial<2, 14> { using ZPZ = aerobus::zpz<2>; using type = POLYV<ZPZV<1>, ZPZV<0>, ZPZV<0>, ZPZV<0>, ZPZV<0>, ZPZV<0>, ZPZV<0>, ZPZV<1>, ZPZV<0>, ZPZV<1>, ZPZV<0>, ZPZV<1>, ZPZV<0>, ZPZV<0>, ZPZV<1»; };  // NOLINT
template<> struct ConwayPolynomial<2, 15> { 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<0>, ZPZV<1>, ZPZV<0>, ZPZV<1»; };  // NOLINT
template<> struct ConwayPolynomial<2, 16> { 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<0>, ZPZV<1>, ZPZV<0>, ZPZV<1>, ZPZV<1>, ZPZV<0>, ZPZV<1»; };  // NOLINT
template<> struct ConwayPolynomial<2, 17> { 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<0>, ZPZV<0>, ZPZV<0>, ZPZV<0>, ZPZV<1>, ZPZV<0>, ZPZV<0>, ZPZV<1»; };  // NOLINT
template<> struct ConwayPolynomial<2, 18> { using ZPZ = aerobus::zpz<2>; using type = POLYV<ZPZV<1>, ZPZV<0>, ZPZV<0>, ZPZV<0>, ZPZV<0>, ZPZV<0>, ZPZV<1>, ZPZV<0>, ZPZV<1>, ZPZV<0>, ZPZV<0>, ZPZV<0>, ZPZV<0>, ZPZV<0>, ZPZV<0>, ZPZV<0>, ZPZV<0>, ZPZV<1>, ZPZV<1»; };  // NOLINT
template<> struct ConwayPolynomial<2, 19> { 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<0>, ZPZV<0>, ZPZV<0>, ZPZV<0>, ZPZV<1>, ZPZV<0>, ZPZV<0>, ZPZV<1>, ZPZV<1>, ZPZV<1»; };  // NOLINT
template<> struct ConwayPolynomial<2, 20> { 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<0>, ZPZV<1>, ZPZV<1>, ZPZV<1>, ZPZV<1>, ZPZV<0>, ZPZV<0>, ZPZV<1>, ZPZV<1»; };  // NOLINT
template<> struct ConwayPolynomial<3, 1> { using ZPZ = aerobus::zpz<3>; using type = POLYV<ZPZV<1>, ZPZV<1»; };  // NOLINT
template<> struct ConwayPolynomial<3, 2> { using ZPZ = aerobus::zpz<3>; using type = POLYV<ZPZV<1>, ZPZV<2>, ZPZV<2»; };  // NOLINT
template<> struct ConwayPolynomial<3, 3> { using ZPZ = aerobus::zpz<3>; using type = POLYV<ZPZV<1>, ZPZV<0>, ZPZV<2>, ZPZV<1»; };  // NOLINT
template<> struct ConwayPolynomial<3, 4> { using ZPZ = aerobus::zpz<3>; using type = POLYV<ZPZV<1>, ZPZV<2>, ZPZV<0>, ZPZV<0>, ZPZV<2»; };  // NOLINT
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
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»; };  // NOLINT
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
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
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<1>, ZPZV<1»; };  // NOLINT
template<> struct ConwayPolynomial<3, 10> { using ZPZ = aerobus::zpz<3>; using type = POLYV<ZPZV<1>, ZPZV<0>, ZPZV<0>, ZPZV<2>, ZPZV<2>, ZPZV<2>, ZPZV<0>, ZPZV<0>, ZPZV<1>, ZPZV<2»; };  // NOLINT
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<2>, ZPZV<0>, ZPZV<1»; };  // NOLINT
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<0>, ZPZV<1>, ZPZV<0>, ZPZV<2»; };  // NOLINT
template<> struct ConwayPolynomial<3, 13> { 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<0>, ZPZV<0>, ZPZV<2>, ZPZV<1»; };  // NOLINT
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<1>, ZPZV<0>, ZPZV<2>, ZPZV<1>, ZPZV<0>, ZPZV<2»; };  // NOLINT
template<> struct ConwayPolynomial<3, 15> { using ZPZ = aerobus::zpz<3>; using type = POLYV<ZPZV<1>, ZPZV<0>, ZPZV<0>, ZPZV<0>, ZPZV<0>, ZPZV<0>, ZPZV<0>, ZPZV<2>, ZPZV<0>, ZPZV<0>, ZPZV<1>, ZPZV<0>, ZPZV<0>, ZPZV<2>, ZPZV<1>, ZPZV<1»; };  // NOLINT
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<2>, ZPZV<2>, ZPZV<0>, ZPZV<2>, ZPZV<2>, ZPZV<2>, ZPZV<1>, ZPZV<2»; };  // NOLINT
template<> struct ConwayPolynomial<3, 17> { 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<0>, ZPZV<0>, ZPZV<0>, ZPZV<0>, ZPZV<0>, ZPZV<0>, ZPZV<2>, ZPZV<1»; };  // NOLINT
template<> struct ConwayPolynomial<3, 18> { 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<1>, ZPZV<0>, ZPZV<2>, ZPZV<0>, ZPZV<2>, ZPZV<1>, ZPZV<2>, ZPZV<0>, ZPZV<2>, ZPZV<0>, ZPZV<2»; };  // NOLINT
template<> struct ConwayPolynomial<3, 19> { 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<0>, ZPZV<0>, ZPZV<0>, ZPZV<0>, ZPZV<0>, ZPZV<0>, ZPZV<0>, ZPZV<2>, ZPZV<0>, ZPZV<1»; };  // NOLINT
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<2>, ZPZV<0>, ZPZV<1>, ZPZV<1>, ZPZV<1>,
ZPZV<1>, ZPZV<0>, ZPZV<0>, ZPZV<2>, ZPZV<2>, ZPZV<2>, ZPZV<0>, ZPZV<1>, ZPZV<2»; };  // NOLINT
template<> struct ConwayPolynomial<5, 1> { using ZPZ = aerobus::zpz<5>; using type = POLYV<ZPZV<1>, ZPZV<3»; };  // NOLINT
template<> struct ConwayPolynomial<5, 2> { using ZPZ = aerobus::zpz<5>; using type = POLYV<ZPZV<1>, ZPZV<4>, ZPZV<2»; };  // NOLINT
template<> struct ConwayPolynomial<5, 3> { using ZPZ = aerobus::zpz<5>; using type = POLYV<ZPZV<1>, ZPZV<0>, ZPZV<3>, ZPZV<3»; };  // NOLINT
template<> struct ConwayPolynomial<5, 4> { using ZPZ = aerobus::zpz<5>; using type = POLYV<ZPZV<1>, ZPZV<0>, ZPZV<4>, ZPZV<4>, ZPZV<2»; };  // NOLINT
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
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
template<> struct ConwayPolynomial<5, 7> { using ZPZ = aerobus::zpz<5>; using type = POLYV<ZPZV<1>, ZPZV<0>, ZPZV<0>, ZPZV<0>, ZPZV<0>, ZPZV<0>, ZPZV<3>, ZPZV<3»; };  // NOLINT
template<> struct ConwayPolynomial<5, 8> { using ZPZ = aerobus::zpz<5>; using type = POLYV<ZPZV<1>, ZPZV<0>, ZPZV<0>, ZPZV<0>, ZPZV<1>, ZPZV<0>, ZPZV<3>, ZPZV<4>, ZPZV<2»; };  // NOLINT
template<> struct ConwayPolynomial<5, 9> { using ZPZ = aerobus::zpz<5>; using type = POLYV<ZPZV<1>, ZPZV<0>, ZPZV<0>, ZPZV<0>, ZPZV<0>, ZPZV<0>, ZPZV<2>, ZPZV<0>, ZPZV<1>, ZPZV<3»; };  // NOLINT
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»; };  // NOLINT
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<0>, ZPZV<0>, ZPZV<3>, ZPZV<3»; };  // NOLINT
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<3>, ZPZV<2>, ZPZV<2»; };  // NOLINT
template<> struct ConwayPolynomial<5, 13> { 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<0>, ZPZV<4>, ZPZV<3>, ZPZV<3»; };  // NOLINT
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<0>, ZPZV<4>, ZPZV<4>, ZPZV<2>, ZPZV<3>, ZPZV<0>, ZPZV<1>, ZPZV<2»; };  // NOLINT
template<> struct ConwayPolynomial<5, 15> { 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<2>, ZPZV<0>, ZPZV<3>, ZPZV<3>, ZPZV<4>, ZPZV<3»; };  // NOLINT
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<1>, ZPZV<4>, ZPZV<4>, ZPZV<4>, ZPZV<2>, ZPZV<4>, ZPZV<4>, ZPZV<1>, ZPZV<2»; };  // NOLINT
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<0>, ZPZV<0>, ZPZV<0>, ZPZV<0>, ZPZV<0>, ZPZV<0>, ZPZV<0>, ZPZV<0>, ZPZV<0>, ZPZV<3>, ZPZV<2>, ZPZV<3»; };  // NOLINT
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
template<> struct ConwayPolynomial<5, 19> { 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<0>, ZPZV<0>, ZPZV<0>, ZPZV<0>, ZPZV<0>, ZPZV<0>, ZPZV<1>, ZPZV<0>, ZPZV<2>, ZPZV<3»; };  // NOLINT
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<3>, ZPZV<0>, ZPZV<4>, ZPZV<3>, ZPZV<2>, ZPZV<0>, ZPZV<3>, ZPZV<0>, ZPZV<0>, ZPZV<4>, ZPZV<0>, ZPZV<1>, ZPZV<2»; };  // NOLINT
template<> struct ConwayPolynomial<7, 1> { using ZPZ = aerobus::zpz<7>; using type = POLYV<ZPZV<1>, ZPZV<4»; };  // NOLINT
template<> struct ConwayPolynomial<7, 2> { using ZPZ = aerobus::zpz<7>; using type = POLYV<ZPZV<1>, ZPZV<6>, ZPZV<3»; };  // NOLINT
template<> struct ConwayPolynomial<7, 3> { using ZPZ = aerobus::zpz<7>; using type = POLYV<ZPZV<1>, ZPZV<6>, ZPZV<0>, ZPZV<4»; };  // NOLINT
template<> struct ConwayPolynomial<7, 4> { using ZPZ = aerobus::zpz<7>; using type = POLYV<ZPZV<1>, ZPZV<0>, ZPZV<5>, ZPZV<4>, ZPZV<3»; };  // NOLINT
template<> struct ConwayPolynomial<7, 5> { using ZPZ = aerobus::zpz<7>; using type = POLYV<ZPZV<1>, ZPZV<0>, ZPZV<0>, ZPZV<0>, ZPZV<1>, ZPZV<4»; };  // NOLINT
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
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<6>, ZPZV<4»; };  // NOLINT
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
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<4»; };  // NOLINT
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»; };  // NOLINT
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<0>, ZPZV<1>, ZPZV<4»; };  // NOLINT
template<> struct ConwayPolynomial<7, 12> { using ZPZ = aerobus::zpz<7>; using type = POLYV<ZPZV<1>, ZPZV<0>, ZPZV<0>, ZPZV<0>, ZPZV<2>, ZPZV<5>, ZPZV<3>, ZPZV<2>, ZPZV<4>, ZPZV<0>, ZPZV<5>, ZPZV<0>, ZPZV<3»; };  // NOLINT
template<> struct ConwayPolynomial<7, 13> { 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<0>, ZPZV<6>, ZPZV<0>, ZPZV<4»; };  // NOLINT
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<0>, ZPZV<6>, ZPZV<2>, ZPZV<0>, ZPZV<3>, ZPZV<6>, ZPZV<3»; };  // NOLINT
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<0>, ZPZV<5>, ZPZV<6>, ZPZV<6>,
ZPZV<4>, ZPZV<1>, ZPZV<2>, ZPZV<4»; };  // NOLINT
template<> struct ConwayPolynomial<7, 16> { 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<4>, ZPZV<5>, ZPZV<3>, ZPZV<4>,
ZPZV<1>, ZPZV<6>, ZPZV<2>, ZPZV<4>, ZPZV<3»; };  // NOLINT
template<> struct ConwayPolynomial<7, 17> { 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<0>, ZPZV<0>,
ZPZV<0>, ZPZV<0>, ZPZV<0>, ZPZV<0>, ZPZV<1>, ZPZV<4»; };  // NOLINT
template<> struct ConwayPolynomial<7, 18> { using ZPZ = aerobus::zpz<7>; using type = POLYV<ZPZV<1>,
ZPZV<0>, ZPZV<0>, ZPZV<0>, ZPZV<0>, ZPZV<0>, ZPZV<1>, ZPZV<2>, ZPZV<6>, ZPZV<1>, ZPZV<6>, ZPZV<5>,
ZPZV<1>, ZPZV<3>, ZPZV<0>, ZPZV<0>, ZPZV<6>, ZPZV<2>, ZPZV<3»; };  // NOLINT
template<> struct ConwayPolynomial<7, 19> { 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<0>, ZPZV<0>,
ZPZV<0>, ZPZV<0>, ZPZV<0>, ZPZV<0>, ZPZV<0>, ZPZV<5>, ZPZV<0>, ZPZV<4»; };  // NOLINT
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<1>, ZPZV<6>, ZPZV<2>, ZPZV<5>,
ZPZV<2>, ZPZV<3>, ZPZV<1>, ZPZV<3>, ZPZV<0>, ZPZV<3>, ZPZV<0>, ZPZV<1>, ZPZV<3»; };  // NOLINT
template<> struct ConwayPolynomial<11, 1> { using ZPZ = aerobus::zpz<11>; using type = POLYV<ZPZV<1>,
ZPZV<9»; };  // NOLINT
template<> struct ConwayPolynomial<11, 2> { using ZPZ = aerobus::zpz<11>; using type = POLYV<ZPZV<1>,
ZPZV<7>, ZPZV<2»; };  // NOLINT
template<> struct ConwayPolynomial<11, 3> { using ZPZ = aerobus::zpz<11>; using type = POLYV<ZPZV<1>,
ZPZV<0>, ZPZV<2>, ZPZV<9»; };  // NOLINT
template<> struct ConwayPolynomial<11, 4> { using ZPZ = aerobus::zpz<11>; using type = POLYV<ZPZV<1>,
ZPZV<0>, ZPZV<8>, ZPZV<10>, ZPZV<2»; };  // NOLINT
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
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
template<> struct ConwayPolynomial<11, 7> { using ZPZ = aerobus::zpz<11>; using type = POLYV<ZPZV<1>,
ZPZV<0>, ZPZV<0>, ZPZV<0>, ZPZV<0>, ZPZV<0>, ZPZV<4>, ZPZV<9»; };  // NOLINT
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>, ZPZV<1>, ZPZV<7>, ZPZV<2»; };  // NOLINT
template<> struct ConwayPolynomial<11, 9> { using ZPZ = aerobus::zpz<11>; using type = POLYV<ZPZV<1>,
ZPZV<0>, ZPZV<0>, ZPZV<0>, ZPZV<0>, ZPZV<0>, ZPZV<0>, ZPZV<9>, ZPZV<8>, ZPZV<9»; };  // NOLINT
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<10>, ZPZV<6>, ZPZV<6>, ZPZV<2»; };  //
NOLINT
template<> struct ConwayPolynomial<11, 11> { 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<10>, ZPZV<9»; };
// NOLINT
template<> struct ConwayPolynomial<11, 12> { using ZPZ = aerobus::zpz<11>; using type = POLYV<ZPZV<1>,
ZPZV<0>, ZPZV<0>, ZPZV<0>, ZPZV<1>, ZPZV<1>, ZPZV<4>, ZPZV<2>, ZPZV<5>, ZPZV<5>, ZPZV<6>, ZPZV<5>,
ZPZV<2»; };  // NOLINT
template<> struct ConwayPolynomial<11, 13> { 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<0>, ZPZV<0>,
ZPZV<7>, ZPZV<9»; };  // NOLINT
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<9>, ZPZV<6>, ZPZV<4>, ZPZV<8>,
ZPZV<6>, ZPZV<10>, ZPZV<2»; };  // NOLINT
template<> struct ConwayPolynomial<11, 15> { 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<10>, ZPZV<7>, ZPZV<0>,
ZPZV<5>, ZPZV<0>, ZPZV<0>, ZPZV<9»; };  // NOLINT
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<1>, ZPZV<10>, ZPZV<1>, ZPZV<3>,
ZPZV<5>, ZPZV<3>, ZPZV<10>, ZPZV<9>, ZPZV<2»; };  // NOLINT
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<0>, ZPZV<0>, ZPZV<0>, ZPZV<0>, ZPZV<0>,
ZPZV<0>, ZPZV<0>, ZPZV<0>, ZPZV<0>, ZPZV<4>, ZPZV<9»; };  // NOLINT
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<8>, ZPZV<10>, ZPZV<8>, ZPZV<3>, ZPZV<9>,
ZPZV<1>, ZPZV<0>, ZPZV<3>, ZPZV<9>, ZPZV<8>, ZPZV<2>, ZPZV<2»; };  // NOLINT
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<0>, ZPZV<0>, ZPZV<0>, ZPZV<0>,
ZPZV<0>, ZPZV<0>, ZPZV<0>, ZPZV<0>, ZPZV<0>, ZPZV<8>, ZPZV<2>, ZPZV<9»; };  // NOLINT
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<1>, ZPZV<10>, ZPZV<9>, ZPZV<1>,
ZPZV<5>, ZPZV<7>, ZPZV<2>, ZPZV<4>, ZPZV<5>, ZPZV<5>, ZPZV<6>, ZPZV<5>, ZPZV<2»; };  // NOLINT
template<> struct ConwayPolynomial<13, 1> { using ZPZ = aerobus::zpz<13>; using type = POLYV<ZPZV<1>,
ZPZV<11»; };  // NOLINT
template<> struct ConwayPolynomial<13, 2> { using ZPZ = aerobus::zpz<13>; using type = POLYV<ZPZV<1>,
ZPZV<12>, ZPZV<2»; };  // NOLINT
template<> struct ConwayPolynomial<13, 3> { using ZPZ = aerobus::zpz<13>; using type = POLYV<ZPZV<1>,
ZPZV<0>, ZPZV<2>, ZPZV<11»; };  // NOLINT
template<> struct ConwayPolynomial<13, 4> { using ZPZ = aerobus::zpz<13>; using type = POLYV<ZPZV<1>,
ZPZV<0>, ZPZV<3>, ZPZV<12>, ZPZV<2»; };  // NOLINT
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
template<> struct ConwayPolynomial<13, 6> { using ZPZ = aerobus::zpz<13>; using type = POLYV<ZPZV<1>,
ZPZV<0>, ZPZV<10>, ZPZV<11>, ZPZV<11>, ZPZV<2»; };  // NOLINT
template<> struct ConwayPolynomial<13, 7> { using ZPZ = aerobus::zpz<13>; using type = POLYV<ZPZV<1>,
ZPZV<0>, ZPZV<0>, ZPZV<0>, ZPZV<0>, ZPZV<0>, ZPZV<3>, ZPZV<11»; };  // NOLINT
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<3>, ZPZV<2»; };  // NOLINT
template<> struct ConwayPolynomial<13, 9> { using ZPZ = aerobus::zpz<13>; using type = POLYV<ZPZV<1>,
ZPZV<0>, ZPZV<0>, ZPZV<0>, ZPZV<0>, ZPZV<12>, ZPZV<8>, ZPZV<12>, ZPZV<12>, ZPZV<11»; };  // NOLINT
template<> struct ConwayPolynomial<13, 10> { using ZPZ = aerobus::zpz<13>; using type = POLYV<ZPZV<1>,
ZPZV<0>, ZPZV<0>, ZPZV<0>, ZPZV<0>, ZPZV<7>, ZPZV<5>, ZPZV<8>, ZPZV<1>, ZPZV<1>, ZPZV<2»; };  //
```

```
NOLINT
template<> struct ConwayPolynomial<13, 11> { 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<3>, ZPZV<11»; }; // NOLINT
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
template<> struct ConwayPolynomial<13, 13> { 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<0>, ZPZV<0>, ZPZV<12>, ZPZV<11»; };  // NOLINT
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<4>, ZPZV<0>, ZPZV<6>, ZPZV<11>, ZPZV<7>, ZPZV<10>, ZPZV<10>, ZPZV<2»; };  // NOLINT
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<2>, ZPZV<12>, ZPZV<2>, ZPZV<11>, ZPZV<10>, ZPZV<11>, ZPZV<8>, ZPZV<11»; };  // NOLINT
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<3>, ZPZV<12>, ZPZV<8>, ZPZV<2>, ZPZV<12>, ZPZV<9>, ZPZV<12>, ZPZV<6>, ZPZV<2»; };  // NOLINT
template<> struct ConwayPolynomial<13, 17> { 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<0>, ZPZV<0>, ZPZV<0>, ZPZV<0>, ZPZV<0>, ZPZV<10>, ZPZV<6>, ZPZV<11»; };  // NOLINT
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<10>, ZPZV<4>, ZPZV<11>, ZPZV<11>, ZPZV<9>, ZPZV<5>, ZPZV<3>, ZPZV<5>, ZPZV<6>, ZPZV<0>, ZPZV<9>, ZPZV<2»; };  // NOLINT
template<> struct ConwayPolynomial<13, 19> { 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<0>, ZPZV<0>, ZPZV<0>, ZPZV<0>, ZPZV<0>, ZPZV<0>, ZPZV<0>, ZPZV<0>, ZPZV<9>, ZPZV<11»; };  // NOLINT
template<> struct ConwayPolynomial<13, 20> { 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<12>, ZPZV<9>, ZPZV<0>, ZPZV<7>, ZPZV<8>, ZPZV<7>, ZPZV<4>, ZPZV<0>, ZPZV<4>, ZPZV<8>, ZPZV<11>, ZPZV<2»; };  // NOLINT
template<> struct ConwayPolynomial<17, 1> { using ZPZ = aerobus::zpz<17>; using type = POLYV<ZPZV<1>, ZPZV<14»; };  // NOLINT
template<> struct ConwayPolynomial<17, 2> { using ZPZ = aerobus::zpz<17>; using type = POLYV<ZPZV<1>, ZPZV<16>, ZPZV<3»; };  // NOLINT
template<> struct ConwayPolynomial<17, 3> { using ZPZ = aerobus::zpz<17>; using type = POLYV<ZPZV<1>, ZPZV<0>, ZPZV<1>, ZPZV<14»; };  // NOLINT
template<> struct ConwayPolynomial<17, 4> { using ZPZ = aerobus::zpz<17>; using type = POLYV<ZPZV<1>, ZPZV<0>, ZPZV<7>, ZPZV<10>, ZPZV<3»; };  // NOLINT
template<> struct ConwayPolynomial<17, 5> { using ZPZ = aerobus::zpz<17>; using type = POLYV<ZPZV<1>, ZPZV<0>, ZPZV<0>, ZPZV<0>, ZPZV<1>, ZPZV<14»; };  // NOLINT
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
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
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
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<7>, ZPZV<8>, ZPZV<14»; };  // NOLINT
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»; };  // NOLINT
template<> struct ConwayPolynomial<17, 11> { 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<5>, ZPZV<14»; }; // NOLINT
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<13>, ZPZV<6>, ZPZV<14>, ZPZV<9>, ZPZV<3»; };  // NOLINT
template<> struct ConwayPolynomial<17, 13> { 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<0>, ZPZV<0>, ZPZV<15>, ZPZV<14»; };  // NOLINT
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
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<4>, ZPZV<4>, ZPZV<16>, ZPZV<6>, ZPZV<14>, ZPZV<14>, ZPZV<14»; };  // NOLINT
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<1>, ZPZV<13>, ZPZV<5>, ZPZV<2>, ZPZV<12>, ZPZV<13>, ZPZV<12>, ZPZV<1>, ZPZV<3»; };  // NOLINT
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<0>, ZPZV<0>, ZPZV<0>, ZPZV<0>, ZPZV<0>, ZPZV<0>, ZPZV<0>, ZPZV<16>, ZPZV<14»; };  // NOLINT
template<> struct ConwayPolynomial<17, 18> { using ZPZ = aerobus::zpz<17>; using type = POLYV<ZPZV<1>, ZPZV<0>, ZPZV<0>, ZPZV<0>, ZPZV<0>, ZPZV<0>, ZPZV<1>, ZPZV<0>, ZPZV<9>, ZPZV<16>, ZPZV<7>, ZPZV<1>, ZPZV<0>, ZPZV<9>, ZPZV<11>, ZPZV<13>, ZPZV<13>, ZPZV<9>, ZPZV<3»; };  // NOLINT
template<> struct ConwayPolynomial<17, 19> { 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<0>, ZPZV<0>, ZPZV<0>, ZPZV<0>, ZPZV<0>, ZPZV<0>, ZPZV<0>, ZPZV<0>, ZPZV<11>, ZPZV<14»; };  // NOLINT
template<> struct ConwayPolynomial<17, 20> { 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<1>, ZPZV<5>, ZPZV<16>, ZPZV<14>, ZPZV<13>, ZPZV<3>, ZPZV<14>, ZPZV<9>, ZPZV<1>, ZPZV<13>, ZPZV<2>, ZPZV<5>, ZPZV<3»; };  // NOLINT
template<> struct ConwayPolynomial<19, 1> { using ZPZ = aerobus::zpz<19>; using type = POLYV<ZPZV<1>, ZPZV<17»; };  // NOLINT
template<> struct ConwayPolynomial<19, 2> { using ZPZ = aerobus::zpz<19>; using type = POLYV<ZPZV<1>, ZPZV<18>, ZPZV<2»; };  // NOLINT
template<> struct ConwayPolynomial<19, 3> { using ZPZ = aerobus::zpz<19>; using type = POLYV<ZPZV<1>,
ZPZV<0>, ZPZV<4>, ZPZV<17»; };  // NOLINT
template<> struct ConwayPolynomial<19, 4> { using ZPZ = aerobus::zpz<19>; using type = POLYV<ZPZV<1>,
ZPZV<0>, ZPZV<2>, ZPZV<11>, ZPZV<2»; };  // NOLINT
template<> struct ConwayPolynomial<19, 5> { using ZPZ = aerobus::zpz<19>; using type = POLYV<ZPZV<1>,
ZPZV<0>, ZPZV<0>, ZPZV<0>, ZPZV<5>, ZPZV<17»; };  // NOLINT
template<> struct ConwayPolynomial<19, 6> { using ZPZ = aerobus::zpz<19>; using type = POLYV<ZPZV<1>,
ZPZV<0>, ZPZV<0>, ZPZV<17>, ZPZV<17>, ZPZV<6>, ZPZV<2»; };  // NOLINT
template<> struct ConwayPolynomial<19, 7> { using ZPZ = aerobus::zpz<19>; using type = POLYV<ZPZV<1>,
ZPZV<0>, ZPZV<0>, ZPZV<0>, ZPZV<0>, ZPZV<0>, ZPZV<6>, ZPZV<17»; };  // NOLINT
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
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<11>, ZPZV<14>, ZPZV<16>, ZPZV<17»; };  // NOLINT
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<2»; };  //
NOLINT
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<8>, ZPZV<17»; };
// NOLINT
template<> struct ConwayPolynomial<19, 12> { using ZPZ = aerobus::zpz<19>; using type = POLYV<ZPZV<1>,
ZPZV<0>, ZPZV<0>, ZPZV<0>, ZPZV<0>, ZPZV<3>, ZPZV<2>, ZPZV<18>, ZPZV<2>, ZPZV<9>, ZPZV<16>, ZPZV<7>,
ZPZV<2»; };  // NOLINT
template<> struct ConwayPolynomial<19, 13> { 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<0>, ZPZV<0>,
ZPZV<11>, ZPZV<17»; };  // NOLINT
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<11>, ZPZV<11>, ZPZV<11>, ZPZV<1>, ZPZV<5>,
ZPZV<16>, ZPZV<7>, ZPZV<2»; };  // NOLINT
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<1>, ZPZV<10>, ZPZV<11>, ZPZV<13>,
ZPZV<15>, ZPZV<14>, ZPZV<0>, ZPZV<17»; };  // NOLINT
template<> struct ConwayPolynomial<19, 16> { 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<12>, ZPZV<13>, ZPZV<0>,
ZPZV<15>, ZPZV<9>, ZPZV<6>, ZPZV<14>, ZPZV<2»; };  // NOLINT
template<> struct ConwayPolynomial<19, 17> { 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<0>, ZPZV<0>,
ZPZV<0>, ZPZV<0>, ZPZV<0>, ZPZV<0>, ZPZV<2>, ZPZV<17»; };  // NOLINT
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<10>, ZPZV<9>, ZPZV<7>, ZPZV<17>, ZPZV<5>,
ZPZV<0>, ZPZV<16>, ZPZV<5>, ZPZV<7>, ZPZV<3>, ZPZV<14>, ZPZV<2»; };  // NOLINT
template<> struct ConwayPolynomial<19, 19> { 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<0>, ZPZV<0>,
ZPZV<0>, ZPZV<0>, ZPZV<0>, ZPZV<0>, ZPZV<0>, ZPZV<0>, ZPZV<18>, ZPZV<17»; };  // NOLINT
template<> struct ConwayPolynomial<19, 20> { 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<16>, ZPZV<13>, ZPZV<0>,
ZPZV<4>, ZPZV<7>, ZPZV<8>, ZPZV<6>, ZPZV<0>, ZPZV<3>, ZPZV<6>, ZPZV<11>, ZPZV<2»; };  // NOLINT
template<> struct ConwayPolynomial<23, 1> { using ZPZ = aerobus::zpz<23>; using type = POLYV<ZPZV<1>,
ZPZV<18»; };  // NOLINT
template<> struct ConwayPolynomial<23, 2> { using ZPZ = aerobus::zpz<23>; using type = POLYV<ZPZV<1>,
ZPZV<21>, ZPZV<5»; };  // NOLINT
template<> struct ConwayPolynomial<23, 3> { using ZPZ = aerobus::zpz<23>; using type = POLYV<ZPZV<1>,
ZPZV<0>, ZPZV<2>, ZPZV<18»; };  // NOLINT
template<> struct ConwayPolynomial<23, 4> { using ZPZ = aerobus::zpz<23>; using type = POLYV<ZPZV<1>,
ZPZV<0>, ZPZV<3>, ZPZV<19>, ZPZV<5»; };  // NOLINT
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
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
template<> struct ConwayPolynomial<23, 7> { using ZPZ = aerobus::zpz<23>; using type = POLYV<ZPZV<1>,
ZPZV<0>, ZPZV<0>, ZPZV<0>, ZPZV<0>, ZPZV<0>, ZPZV<21>, ZPZV<18»; };  // NOLINT
template<> struct ConwayPolynomial<23, 8> { using ZPZ = aerobus::zpz<23>; using type = POLYV<ZPZV<1>,
ZPZV<0>, ZPZV<0>, ZPZV<0>, ZPZV<3>, ZPZV<20>, ZPZV<5>, ZPZV<3>, ZPZV<5»; };  // NOLINT
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<3>, ZPZV<8>, ZPZV<9>, ZPZV<18»; };  // NOLINT
template<> struct ConwayPolynomial<23, 10> { using ZPZ = aerobus::zpz<23>; using type = POLYV<ZPZV<1>,
ZPZV<0>, ZPZV<0>, ZPZV<0>, ZPZV<0>, ZPZV<17>, ZPZV<5>, ZPZV<15>, ZPZV<6>, ZPZV<1>, ZPZV<5»; };  //
NOLINT
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<22>, ZPZV<7>, ZPZV<18»;
};  // NOLINT
template<> struct ConwayPolynomial<23, 12> { using ZPZ = aerobus::zpz<23>; using type = POLYV<ZPZV<1>,
ZPZV<0>, ZPZV<0>, ZPZV<0>, ZPZV<0>, ZPZV<21>, ZPZV<21>, ZPZV<15>, ZPZV<14>, ZPZV<12>, ZPZV<18>,
ZPZV<12>, ZPZV<5»; };  // NOLINT
template<> struct ConwayPolynomial<23, 13> { 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<0>, ZPZV<0>,
ZPZV<9>, ZPZV<18»; };  // NOLINT
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<5>, ZPZV<16>, ZPZV<1>, ZPZV<18>, ZPZV<19>,
ZPZV<1>, ZPZV<22>, ZPZV<5»; };  // NOLINT
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<2>, ZPZV<8>, ZPZV<15>,
ZPZV<9>, ZPZV<7>, ZPZV<18>, ZPZV<18»; };  // NOLINT
template<> struct ConwayPolynomial<23, 16> { 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<19>, ZPZV<19>, ZPZV<16>,
ZPZV<13>, ZPZV<1>, ZPZV<14>, ZPZV<17>, ZPZV<5»; };  // NOLINT
template<> struct ConwayPolynomial<23, 17> { 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<0>, ZPZV<0>,
ZPZV<0>, ZPZV<0>, ZPZV<0>, ZPZV<0>, ZPZV<20>, ZPZV<18»; };  // NOLINT
template<> struct ConwayPolynomial<23, 18> { using ZPZ = aerobus::zpz<23>; using type = POLYV<ZPZV<1>, ZPZV<0>, ZPZV<0>, ZPZV<0>, ZPZV<0>, ZPZV<0>, ZPZV<1>, ZPZV<18>, ZPZV<2>, ZPZV<1>, ZPZV<18>, ZPZV<3>, ZPZV<16>, ZPZV<21>, ZPZV<0>, ZPZV<11>, ZPZV<3>, ZPZV<19>, ZPZV<5»; };  // NOLINT
template<> struct ConwayPolynomial<23, 19> { 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<0>, ZPZV<0>, ZPZV<0>, ZPZV<0>, ZPZV<0>, ZPZV<0>, ZPZV<0>, ZPZV<0>, ZPZV<5>, ZPZV<18»; };  // NOLINT
template<> struct ConwayPolynomial<29, 1> { using ZPZ = aerobus::zpz<29>; using type = POLYV<ZPZV<1>, ZPZV<27»; };  // NOLINT
template<> struct ConwayPolynomial<29, 2> { using ZPZ = aerobus::zpz<29>; using type = POLYV<ZPZV<1>, ZPZV<24>, ZPZV<2»; };  // NOLINT
template<> struct ConwayPolynomial<29, 3> { using ZPZ = aerobus::zpz<29>; using type = POLYV<ZPZV<1>, ZPZV<0>, ZPZV<2>, ZPZV<27»; };  // NOLINT
template<> struct ConwayPolynomial<29, 4> { using ZPZ = aerobus::zpz<29>; using type = POLYV<ZPZV<1>, ZPZV<0>, ZPZV<2>, ZPZV<15>, ZPZV<2»; };  // NOLINT
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
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
template<> struct ConwayPolynomial<29, 7> { using ZPZ = aerobus::zpz<29>; using type = POLYV<ZPZV<1>, ZPZV<0>, ZPZV<0>, ZPZV<0>, ZPZV<0>, ZPZV<0>, ZPZV<2>, ZPZV<27»; };  // NOLINT
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
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<4>, ZPZV<22>, ZPZV<22>, ZPZV<27»; };  // NOLINT
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<22>, ZPZV<2»; };  // NOLINT
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<28>, ZPZV<8>, ZPZV<27»; };  // NOLINT
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
template<> struct ConwayPolynomial<29, 13> { 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<7>, ZPZV<27»; };  // NOLINT
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<1>, ZPZV<3>, ZPZV<14>, ZPZV<10>, ZPZV<21>, ZPZV<18>, ZPZV<27>, ZPZV<5>, ZPZV<2»; };  // NOLINT
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<13>, ZPZV<14>, ZPZV<8>, ZPZV<1>, ZPZV<12>, ZPZV<26>, ZPZV<27»; };  // NOLINT
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<6>, ZPZV<27>, ZPZV<2>, ZPZV<18>, ZPZV<23>, ZPZV<1>, ZPZV<27>, ZPZV<10>, ZPZV<2»; };  // NOLINT
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<0>, ZPZV<0>, ZPZV<0>, ZPZV<0>, ZPZV<0>, ZPZV<0>, ZPZV<0>, ZPZV<2>, ZPZV<27»; };  // NOLINT
template<> struct ConwayPolynomial<29, 18> { using ZPZ = aerobus::zpz<29>; using type = POLYV<ZPZV<1>, 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
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<0>, ZPZV<0>, ZPZV<0>, ZPZV<0>, ZPZV<0>, ZPZV<0>, ZPZV<4>, ZPZV<27»; };  // NOLINT
template<> struct ConwayPolynomial<31, 1> { using ZPZ = aerobus::zpz<31>; using type = POLYV<ZPZV<1>, ZPZV<28»; };  // NOLINT
template<> struct ConwayPolynomial<31, 2> { using ZPZ = aerobus::zpz<31>; using type = POLYV<ZPZV<1>, ZPZV<29>, ZPZV<3»; };  // NOLINT
template<> struct ConwayPolynomial<31, 3> { using ZPZ = aerobus::zpz<31>; using type = POLYV<ZPZV<1>, ZPZV<0>, ZPZV<1>, ZPZV<28»; };  // NOLINT
template<> struct ConwayPolynomial<31, 4> { using ZPZ = aerobus::zpz<31>; using type = POLYV<ZPZV<1>, ZPZV<0>, ZPZV<3>, ZPZV<16>, ZPZV<3»; };  // NOLINT
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
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
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
template<> struct ConwayPolynomial<31, 8> { using ZPZ = aerobus::zpz<31>; using type = POLYV<ZPZV<1>, ZPZV<0>, ZPZV<0>, ZPZV<0>, ZPZV<0>, ZPZV<25>, ZPZV<12>, ZPZV<24>, ZPZV<3»; };  // NOLINT
template<> struct ConwayPolynomial<31, 9> { using ZPZ = aerobus::zpz<31>; using type = POLYV<ZPZV<1>, ZPZV<0>, ZPZV<0>, ZPZV<0>, ZPZV<0>, ZPZV<0>, ZPZV<4>, ZPZV<20>, ZPZV<29>, ZPZV<28»; };  // NOLINT
template<> struct ConwayPolynomial<31, 10> { 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<3»; };  // NOLINT
template<> struct ConwayPolynomial<31, 11> { 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<20>, ZPZV<28»; };  // NOLINT
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<9>, ZPZV<25>, ZPZV<12>, ZPZV<3»; };  // NOLINT
template<> struct ConwayPolynomial<31, 13> { 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<0>, ZPZV<0>, ZPZV<6>, ZPZV<28»; };  // NOLINT
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<0>, ZPZV<10>, ZPZV<5>, ZPZV<1>, ZPZV<1>, ZPZV<18>,
ZPZV<18>, ZPZV<6>, ZPZV<3»; };  // NOLINT
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<30>, ZPZV<29>, ZPZV<12>, ZPZV<13>, ZPZV<23>, ZPZV<25>, ZPZV<28»; };  // NOLINT
template<> struct ConwayPolynomial<31, 16> { 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<28>, ZPZV<24>, ZPZV<26>, ZPZV<28>, ZPZV<11>, ZPZV<19>, ZPZV<27>, ZPZV<3»; };  // NOLINT
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<0>, ZPZV<0>, ZPZV<0>, ZPZV<0>, ZPZV<0>, ZPZV<0>, ZPZV<0>, ZPZV<0>, ZPZV<10>, ZPZV<28»; };  // NOLINT
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<2>, ZPZV<7>, ZPZV<12>, ZPZV<11>, ZPZV<25>, ZPZV<25>, ZPZV<10>, ZPZV<6>, ZPZV<3»; };  // NOLINT
template<> struct ConwayPolynomial<31, 19> { 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<0>, ZPZV<0>, ZPZV<0>, ZPZV<0>, ZPZV<0>, ZPZV<0>, ZPZV<0>, ZPZV<0>, ZPZV<7>, ZPZV<28»; };  // NOLINT
template<> struct ConwayPolynomial<37, 1> { using ZPZ = aerobus::zpz<37>; using type = POLYV<ZPZV<1>, ZPZV<35»; };  // NOLINT
template<> struct ConwayPolynomial<37, 2> { using ZPZ = aerobus::zpz<37>; using type = POLYV<ZPZV<1>, ZPZV<33>, ZPZV<2»; };  // NOLINT
template<> struct ConwayPolynomial<37, 3> { using ZPZ = aerobus::zpz<37>; using type = POLYV<ZPZV<1>, ZPZV<0>, ZPZV<6>, ZPZV<35»; };  // NOLINT
template<> struct ConwayPolynomial<37, 4> { using ZPZ = aerobus::zpz<37>; using type = POLYV<ZPZV<1>, ZPZV<0>, ZPZV<6>, ZPZV<24>, ZPZV<2»; };  // NOLINT
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
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
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
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
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<20>, ZPZV<32>, ZPZV<35»; };  // NOLINT
template<> struct ConwayPolynomial<37, 10> { using ZPZ = aerobus::zpz<37>; using type = POLYV<ZPZV<1>, ZPZV<0>, ZPZV<0>, ZPZV<0>, ZPZV<0>, ZPZV<8>, ZPZV<29>, ZPZV<18>, ZPZV<11>, ZPZV<4>, ZPZV<2»; };  // NOLINT
template<> struct ConwayPolynomial<37, 11> { 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<0>, ZPZV<0>, ZPZV<2>, ZPZV<35»; };  // NOLINT
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<18>, ZPZV<33>, ZPZV<2»; };  // NOLINT
template<> struct ConwayPolynomial<37, 13> { 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<0>, ZPZV<0>, ZPZV<0>, ZPZV<0>, ZPZV<6>, ZPZV<35»; };  // NOLINT
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<9>, ZPZV<2»; };  // NOLINT
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<1>, ZPZV<31>, ZPZV<28>, ZPZV<27>, ZPZV<13>, ZPZV<34>, ZPZV<33>, ZPZV<35»; };  // NOLINT
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<0>, ZPZV<0>, ZPZV<0>, ZPZV<0>, ZPZV<0>, ZPZV<0>, ZPZV<0>, ZPZV<0>, ZPZV<0>, ZPZV<3>, ZPZV<35»; };  // NOLINT
template<> struct ConwayPolynomial<37, 18> { using ZPZ = aerobus::zpz<37>; using type = POLYV<ZPZV<1>, ZPZV<0>, ZPZV<0>, ZPZV<0>, ZPZV<0>, ZPZV<0>, ZPZV<1>, ZPZV<8>, ZPZV<19>, ZPZV<15>, ZPZV<1>, ZPZV<22>, ZPZV<20>, ZPZV<12>, ZPZV<32>, ZPZV<14>, ZPZV<27>, ZPZV<20>, ZPZV<2»; };  // NOLINT
template<> struct ConwayPolynomial<37, 19> { 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<0>, ZPZV<0>, ZPZV<0>, ZPZV<0>, ZPZV<0>, ZPZV<0>, ZPZV<0>, ZPZV<0>, ZPZV<0>, ZPZV<36>, ZPZV<23>, ZPZV<35»; };  // NOLINT
template<> struct ConwayPolynomial<41, 1> { using ZPZ = aerobus::zpz<41>; using type = POLYV<ZPZV<1>, ZPZV<35»; };  // NOLINT
template<> struct ConwayPolynomial<41, 2> { using ZPZ = aerobus::zpz<41>; using type = POLYV<ZPZV<1>, ZPZV<38>, ZPZV<6»; };  // NOLINT
template<> struct ConwayPolynomial<41, 3> { using ZPZ = aerobus::zpz<41>; using type = POLYV<ZPZV<1>, ZPZV<0>, ZPZV<1>, ZPZV<35»; };  // NOLINT
template<> struct ConwayPolynomial<41, 4> { using ZPZ = aerobus::zpz<41>; using type = POLYV<ZPZV<1>, ZPZV<0>, ZPZV<0>, ZPZV<23>, ZPZV<6»; };  // NOLINT
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
template<> struct ConwayPolynomial<41, 6> { using ZPZ = aerobus::zpz<41>; using type = POLYV<ZPZV<1>, ZPZV<0>, ZPZV<4>, ZPZV<33>, ZPZV<39>, ZPZV<6>, ZPZV<6»; };  // NOLINT
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<35»; };  // NOLINT
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
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<4>, ZPZV<31>, ZPZV<5>, ZPZV<35»; };  // NOLINT
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<31>, ZPZV<8>, ZPZV<20>, ZPZV<30>, ZPZV<6»; };  // NOLINT
template<> struct ConwayPolynomial<41, 11> { 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<20>, ZPZV<35»; };  // NOLINT
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<13>, ZPZV<34>, ZPZV<24>, ZPZV<21>,
ZPZV<27>, ZPZV<6»; };  // NOLINT
template<> struct ConwayPolynomial<41, 13> { 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<0>, ZPZV<0>,
ZPZV<13>, ZPZV<35»; };  // NOLINT
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<0>, ZPZV<12>, ZPZV<15>, ZPZV<4>, ZPZV<27>, ZPZV<11>,
ZPZV<39>, ZPZV<10>, ZPZV<6»; };  // NOLINT
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<29>, ZPZV<16>, ZPZV<2>,
ZPZV<35>, ZPZV<10>, ZPZV<21>, ZPZV<35»; };  // NOLINT
template<> struct ConwayPolynomial<41, 17> { 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<0>, ZPZV<0>,
ZPZV<0>, ZPZV<0>, ZPZV<0>, ZPZV<0>, ZPZV<4>, ZPZV<35»; };  // NOLINT
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<1>, ZPZV<7>, ZPZV<20>, ZPZV<23>, ZPZV<35>,
ZPZV<38>, ZPZV<24>, ZPZV<12>, ZPZV<29>, ZPZV<10>, ZPZV<6>, ZPZV<6»; };  // NOLINT
template<> struct ConwayPolynomial<41, 19> { 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<0>, ZPZV<0>,
ZPZV<0>, ZPZV<0>, ZPZV<0>, ZPZV<0>, ZPZV<0>, ZPZV<0>, ZPZV<10>, ZPZV<35»; };  // NOLINT
template<> struct ConwayPolynomial<43, 1> { using ZPZ = aerobus::zpz<43>; using type = POLYV<ZPZV<1>,
ZPZV<40»; };  // NOLINT
template<> struct ConwayPolynomial<43, 2> { using ZPZ = aerobus::zpz<43>; using type = POLYV<ZPZV<1>,
ZPZV<42>, ZPZV<3»; };  // NOLINT
template<> struct ConwayPolynomial<43, 3> { using ZPZ = aerobus::zpz<43>; using type = POLYV<ZPZV<1>,
ZPZV<0>, ZPZV<1>, ZPZV<40»; };  // NOLINT
template<> struct ConwayPolynomial<43, 4> { using ZPZ = aerobus::zpz<43>; using type = POLYV<ZPZV<1>,
ZPZV<0>, ZPZV<5>, ZPZV<42>, ZPZV<3»; };  // NOLINT
template<> struct ConwayPolynomial<43, 5> { using ZPZ = aerobus::zpz<43>; using type = POLYV<ZPZV<1>,
ZPZV<0>, ZPZV<0>, ZPZV<0>, ZPZV<8>, ZPZV<40»; };  // NOLINT
template<> struct ConwayPolynomial<43, 6> { using ZPZ = aerobus::zpz<43>; using type = POLYV<ZPZV<1>,
ZPZV<0>, ZPZV<0>, ZPZV<19>, ZPZV<28>, ZPZV<21>, ZPZV<3»; };  // NOLINT
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
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»; };  // NOLINT
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<12>, ZPZV<39>, ZPZV<1>, ZPZV<40»; };  // NOLINT
template<> struct ConwayPolynomial<43, 10> { using ZPZ = aerobus::zpz<43>; using type = POLYV<ZPZV<1>,
ZPZV<0>, ZPZV<0>, ZPZV<0>, ZPZV<3>, ZPZV<26>, ZPZV<36>, ZPZV<5>, ZPZV<27>, ZPZV<24>, ZPZV<3»; };  //
NOLINT
template<> struct ConwayPolynomial<43, 11> { 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<0>, ZPZV<7>, ZPZV<40»; };
// NOLINT
template<> struct ConwayPolynomial<43, 12> { using ZPZ = aerobus::zpz<43>; using type = POLYV<ZPZV<1>,
ZPZV<0>, ZPZV<0>, ZPZV<0>, ZPZV<0>, ZPZV<34>, ZPZV<27>, ZPZV<16>, ZPZV<17>, ZPZV<6>, ZPZV<23>,
ZPZV<38>, ZPZV<3»; };  // NOLINT
template<> struct ConwayPolynomial<43, 13> { 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<0>, ZPZV<0>, ZPZV<0>,
ZPZV<4>, ZPZV<40»; };  // NOLINT
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
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<2>, ZPZV<37>, ZPZV<22>, ZPZV<42>,
ZPZV<4>, ZPZV<15>, ZPZV<37>, ZPZV<40»; };  // NOLINT
template<> struct ConwayPolynomial<43, 17> { 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<0>, ZPZV<0>, ZPZV<0>,
ZPZV<0>, ZPZV<0>, ZPZV<0>, ZPZV<0>, ZPZV<36>, ZPZV<40»; };  // NOLINT
template<> struct ConwayPolynomial<43, 18> { using ZPZ = aerobus::zpz<43>; using type = POLYV<ZPZV<1>,
ZPZV<0>, ZPZV<0>, ZPZV<0>, ZPZV<0>, ZPZV<0>, ZPZV<1>, ZPZV<3>, ZPZV<28>, ZPZV<41>, ZPZV<24>, ZPZV<7>,
ZPZV<24>, ZPZV<29>, ZPZV<16>, ZPZV<34>, ZPZV<37>, ZPZV<18>, ZPZV<3»; };  // NOLINT
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<0>, ZPZV<0>, ZPZV<0>,
ZPZV<0>, ZPZV<0>, ZPZV<0>, ZPZV<0>, ZPZV<0>, ZPZV<0>, ZPZV<30>, ZPZV<40»; };  // NOLINT
template<> struct ConwayPolynomial<47, 1> { using ZPZ = aerobus::zpz<47>; using type = POLYV<ZPZV<1>,
ZPZV<42»; };  // NOLINT
template<> struct ConwayPolynomial<47, 2> { using ZPZ = aerobus::zpz<47>; using type = POLYV<ZPZV<1>,
ZPZV<45>, ZPZV<5»; };  // NOLINT
template<> struct ConwayPolynomial<47, 3> { using ZPZ = aerobus::zpz<47>; using type = POLYV<ZPZV<1>,
ZPZV<0>, ZPZV<3>, ZPZV<42»; };  // NOLINT
template<> struct ConwayPolynomial<47, 4> { using ZPZ = aerobus::zpz<47>; using type = POLYV<ZPZV<1>,
ZPZV<0>, ZPZV<8>, ZPZV<40>, ZPZV<5»; };  // NOLINT
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
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
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<12>, ZPZV<42»; };  // NOLINT
template<> struct ConwayPolynomial<47, 8> { using ZPZ = aerobus::zpz<47>; using type = POLYV<ZPZV<1>,
ZPZV<0>, ZPZV<0>, ZPZV<0>, ZPZV<1>, ZPZV<29>, ZPZV<19>, ZPZV<3>, ZPZV<5»; };  // NOLINT
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»; };  // NOLINT
template<> struct ConwayPolynomial<47, 10> { using ZPZ = aerobus::zpz<47>; using type = POLYV<ZPZV<1>,
ZPZV<0>, ZPZV<0>, ZPZV<0>, ZPZV<1>, ZPZV<42>, ZPZV<14>, ZPZV<18>, ZPZV<45>, ZPZV<45>, ZPZV<5»; };  //
NOLINT
template<> struct ConwayPolynomial<47, 11> { 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<0>, ZPZV<6>, ZPZV<42»; };
```

```
// NOLINT
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<35>, ZPZV<12>, ZPZV<46>, ZPZV<14>,
ZPZV<9>, ZPZV<5»; };  // NOLINT
template<> struct ConwayPolynomial<47, 13> { 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<0>, ZPZV<0>, ZPZV<0>,
ZPZV<5>, ZPZV<42»; };  // NOLINT
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<36>, ZPZV<20>, ZPZV<30>, ZPZV<17>,
ZPZV<24>, ZPZV<9>, ZPZV<32>, ZPZV<5»; };  // NOLINT
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<0>, ZPZV<43>, ZPZV<31>, ZPZV<14>,
ZPZV<42>, ZPZV<13>, ZPZV<17>, ZPZV<42»; };  // NOLINT
template<> struct ConwayPolynomial<47, 17> { 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<0>, ZPZV<0>, ZPZV<0>,
ZPZV<0>, ZPZV<0>, ZPZV<0>, ZPZV<0>, ZPZV<16>, ZPZV<42»; };  // NOLINT
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<0>, ZPZV<6>, ZPZV<41>, ZPZV<42>, ZPZV<26>, ZPZV<44>,
ZPZV<24>, ZPZV<22>, ZPZV<11>, ZPZV<5>, ZPZV<45>, ZPZV<33>, ZPZV<5»; };  // NOLINT
template<> struct ConwayPolynomial<47, 19> { 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<0>, ZPZV<0>, ZPZV<0>,
ZPZV<0>, ZPZV<0>, ZPZV<0>, ZPZV<0>, ZPZV<0>, ZPZV<0>, ZPZV<35>, ZPZV<42»; };  // NOLINT
template<> struct ConwayPolynomial<53, 1> { using ZPZ = aerobus::zpz<53>; using type = POLYV<ZPZV<1>,
ZPZV<51»; };  // NOLINT
template<> struct ConwayPolynomial<53, 2> { using ZPZ = aerobus::zpz<53>; using type = POLYV<ZPZV<1>,
ZPZV<49>, ZPZV<2»; };  // NOLINT
template<> struct ConwayPolynomial<53, 3> { using ZPZ = aerobus::zpz<53>; using type = POLYV<ZPZV<1>,
ZPZV<0>, ZPZV<3>, ZPZV<51»; };  // NOLINT
template<> struct ConwayPolynomial<53, 4> { using ZPZ = aerobus::zpz<53>; using type = POLYV<ZPZV<1>,
ZPZV<0>, ZPZV<9>, ZPZV<38>, ZPZV<2»; };  // NOLINT
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
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
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<9>, ZPZV<51»; };  // NOLINT
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
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<13>, ZPZV<5>, ZPZV<51»; };  // NOLINT
template<> struct ConwayPolynomial<53, 10> { using ZPZ = aerobus::zpz<53>; using type = POLYV<ZPZV<1>,
ZPZV<0>, ZPZV<0>, ZPZV<0>, ZPZV<1>, ZPZV<0>, ZPZV<1>, ZPZV<27>, ZPZV<15>, ZPZV<29>, ZPZV<2»; };  //
NOLINT
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<15>, ZPZV<51»;
};  // NOLINT
template<> struct ConwayPolynomial<53, 12> { using ZPZ = aerobus::zpz<53>; using type = POLYV<ZPZV<1>,
ZPZV<0>, ZPZV<0>, ZPZV<0>, ZPZV<2>, ZPZV<34>, ZPZV<4>, ZPZV<13>, ZPZV<10>, ZPZV<42>, ZPZV<34>,
ZPZV<41>, ZPZV<2»; };  // NOLINT
template<> struct ConwayPolynomial<53, 13> { 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<0>, ZPZV<52>,
ZPZV<28>, ZPZV<51»; };  // NOLINT
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<0>, ZPZV<37>,
ZPZV<12>, ZPZV<23>, ZPZV<2»; };  // NOLINT
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<0>, ZPZV<22>, ZPZV<31>, ZPZV<15>,
ZPZV<11>, ZPZV<20>, ZPZV<4>, ZPZV<51»; };  // NOLINT
template<> struct ConwayPolynomial<53, 17> { 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<0>, ZPZV<0>,
ZPZV<0>, ZPZV<0>, ZPZV<0>, ZPZV<0>, ZPZV<12>, ZPZV<51»; };  // NOLINT
template<> struct ConwayPolynomial<53, 18> { using ZPZ = aerobus::zpz<53>; using type = POLYV<ZPZV<1>,
ZPZV<0>, ZPZV<0>, ZPZV<0>, ZPZV<0>, ZPZV<0>, ZPZV<0>, ZPZV<52>, ZPZV<31>, ZPZV<51>, ZPZV<27>, ZPZV<0>,
ZPZV<39>, ZPZV<44>, ZPZV<6>, ZPZV<8>, ZPZV<16>, ZPZV<11>, ZPZV<2»; };  // NOLINT
template<> struct ConwayPolynomial<53, 19> { 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<0>, ZPZV<0>,
ZPZV<0>, ZPZV<0>, ZPZV<0>, ZPZV<0>, ZPZV<0>, ZPZV<0>, ZPZV<11>, ZPZV<51»; };  // NOLINT
template<> struct ConwayPolynomial<59, 1> { using ZPZ = aerobus::zpz<59>; using type = POLYV<ZPZV<1>,
ZPZV<57»; };  // NOLINT
template<> struct ConwayPolynomial<59, 2> { using ZPZ = aerobus::zpz<59>; using type = POLYV<ZPZV<1>,
ZPZV<58>, ZPZV<2»; };  // NOLINT
template<> struct ConwayPolynomial<59, 3> { using ZPZ = aerobus::zpz<59>; using type = POLYV<ZPZV<1>,
ZPZV<0>, ZPZV<5>, ZPZV<57»; };  // NOLINT
template<> struct ConwayPolynomial<59, 4> { using ZPZ = aerobus::zpz<59>; using type = POLYV<ZPZV<1>,
ZPZV<0>, ZPZV<2>, ZPZV<40>, ZPZV<2»; };  // NOLINT
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
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
template<> struct ConwayPolynomial<59, 7> { using ZPZ = aerobus::zpz<59>; using type = POLYV<ZPZV<1>,
ZPZV<0>, ZPZV<0>, ZPZV<0>, ZPZV<0>, ZPZV<0>, ZPZV<10>, ZPZV<57»; };  // NOLINT
template<> struct ConwayPolynomial<59, 8> { using ZPZ = aerobus::zpz<59>; using type = POLYV<ZPZV<1>,
ZPZV<0>, ZPZV<0>, ZPZV<0>, ZPZV<16>, ZPZV<32>, ZPZV<2>, ZPZV<50>, ZPZV<2»; };  // NOLINT
template<> struct ConwayPolynomial<59, 9> { using ZPZ = aerobus::zpz<59>; using type = POLYV<ZPZV<1>,
ZPZV<0>, ZPZV<0>, ZPZV<0>, ZPZV<0>, ZPZV<0>, ZPZV<1>, ZPZV<32>, ZPZV<47>, ZPZV<57»; };  // NOLINT
template<> struct ConwayPolynomial<59, 10> { using ZPZ = aerobus::zpz<59>; using type = POLYV<ZPZV<1>,
ZPZV<0>, ZPZV<0>, ZPZV<0>, ZPZV<1>, ZPZV<28>, ZPZV<25>, ZPZV<4>, ZPZV<39>, ZPZV<15>, ZPZV<2»; };  //
```

```
NOLINT
template<> struct ConwayPolynomial<59, 11> { 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<6>, ZPZV<57»; }; // NOLINT
template<> struct ConwayPolynomial<59, 12> { using ZPZ = aerobus::zpz<59>; using type = POLYV<ZPZV<1>, ZPZV<0>, ZPZV<0>, ZPZV<0>, ZPZV<0>, ZPZV<39>, ZPZV<25>, ZPZV<51>, ZPZV<21>, ZPZV<38>, ZPZV<8>, ZPZV<1>, ZPZV<2»; };  // NOLINT
template<> struct ConwayPolynomial<59, 13> { 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<0>, ZPZV<0>, ZPZV<3>, ZPZV<57»; };  // NOLINT
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<11>, ZPZV<13>, ZPZV<25>, ZPZV<32>, ZPZV<26>, ZPZV<2»; };  // NOLINT
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<57>, ZPZV<24>, ZPZV<23>, ZPZV<13>, ZPZV<39>, ZPZV<58>, ZPZV<57»; };  // NOLINT
template<> struct ConwayPolynomial<59, 17> { 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<0>, ZPZV<0>, ZPZV<0>, ZPZV<0>, ZPZV<0>, ZPZV<0>, ZPZV<9>, ZPZV<57»; };  // NOLINT
template<> struct ConwayPolynomial<59, 18> { using ZPZ = aerobus::zpz<59>; using type = POLYV<ZPZV<1>, ZPZV<0>, ZPZV<0>, ZPZV<0>, ZPZV<0>, ZPZV<0>, ZPZV<1>, ZPZV<37>, ZPZV<38>, ZPZV<27>, ZPZV<11>, ZPZV<14>, ZPZV<7>, ZPZV<44>, ZPZV<16>, ZPZV<47>, ZPZV<34>, ZPZV<32>, ZPZV<2»; };  // NOLINT
template<> struct ConwayPolynomial<59, 19> { 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<0>, ZPZV<0>, ZPZV<0>, ZPZV<0>, ZPZV<0>, ZPZV<0>, ZPZV<0>, ZPZV<0>, ZPZV<11>, ZPZV<57»; };  // NOLINT
template<> struct ConwayPolynomial<61, 1> { using ZPZ = aerobus::zpz<61>; using type = POLYV<ZPZV<1>, ZPZV<59»; };  // NOLINT
template<> struct ConwayPolynomial<61, 2> { using ZPZ = aerobus::zpz<61>; using type = POLYV<ZPZV<1>, ZPZV<60>, ZPZV<2»; };  // NOLINT
template<> struct ConwayPolynomial<61, 3> { using ZPZ = aerobus::zpz<61>; using type = POLYV<ZPZV<1>, ZPZV<0>, ZPZV<7>, ZPZV<59»; };  // NOLINT
template<> struct ConwayPolynomial<61, 4> { using ZPZ = aerobus::zpz<61>; using type = POLYV<ZPZV<1>, ZPZV<0>, ZPZV<3>, ZPZV<40>, ZPZV<2»; };  // NOLINT
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»; };  // NOLINT
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
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
template<> struct ConwayPolynomial<61, 8> { using ZPZ = aerobus::zpz<61>; using type = POLYV<ZPZV<1>, ZPZV<0>, ZPZV<0>, ZPZV<0>, ZPZV<0>, ZPZV<57>, ZPZV<1>, ZPZV<56>, ZPZV<2»; };  // NOLINT
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<9>, ZPZV<50>, ZPZV<18>, ZPZV<59»; };  // NOLINT
template<> struct ConwayPolynomial<61, 10> { using ZPZ = aerobus::zpz<61>; using type = POLYV<ZPZV<1>, ZPZV<0>, ZPZV<0>, ZPZV<0>, ZPZV<0>, ZPZV<28>, ZPZV<15>, ZPZV<44>, ZPZV<16>, ZPZV<6>, ZPZV<2»; };  // NOLINT
template<> struct ConwayPolynomial<61, 11> { 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<18>, ZPZV<59»; };  // NOLINT
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
template<> struct ConwayPolynomial<61, 13> { 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<0>, ZPZV<0>, ZPZV<3>, ZPZV<59»; };  // NOLINT
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<1>, ZPZV<48>, ZPZV<26>, ZPZV<11>, ZPZV<8>, ZPZV<30>, ZPZV<54>, ZPZV<48>, ZPZV<2»; };  // NOLINT
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<39>, ZPZV<35>, ZPZV<44>, ZPZV<25>, ZPZV<23>, ZPZV<51>, ZPZV<59»; };  // NOLINT
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<0>, ZPZV<0>, ZPZV<0>, ZPZV<0>, ZPZV<0>, ZPZV<0>, ZPZV<0>, ZPZV<10>, ZPZV<59»; };  // NOLINT
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<13>, ZPZV<36>, ZPZV<4>, ZPZV<32>, ZPZV<57>, ZPZV<42>, ZPZV<25>, ZPZV<25>, ZPZV<52>, ZPZV<2»; };  // NOLINT
template<> struct ConwayPolynomial<61, 19> { 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<0>, ZPZV<0>, ZPZV<0>, ZPZV<0>, ZPZV<0>, ZPZV<0>, ZPZV<0>, ZPZV<0>, ZPZV<2>, ZPZV<59»; };  // NOLINT
template<> struct ConwayPolynomial<67, 1> { using ZPZ = aerobus::zpz<67>; using type = POLYV<ZPZV<1>, ZPZV<65»; };  // NOLINT
template<> struct ConwayPolynomial<67, 2> { using ZPZ = aerobus::zpz<67>; using type = POLYV<ZPZV<1>, ZPZV<63>, ZPZV<2»; };  // NOLINT
template<> struct ConwayPolynomial<67, 3> { using ZPZ = aerobus::zpz<67>; using type = POLYV<ZPZV<1>, ZPZV<0>, ZPZV<6>, ZPZV<65»; };  // NOLINT
template<> struct ConwayPolynomial<67, 4> { using ZPZ = aerobus::zpz<67>; using type = POLYV<ZPZV<1>, ZPZV<0>, ZPZV<8>, ZPZV<54>, ZPZV<2»; };  // NOLINT
template<> struct ConwayPolynomial<67, 5> { using ZPZ = aerobus::zpz<67>; using type = POLYV<ZPZV<1>, ZPZV<0>, ZPZV<0>, ZPZV<0>, ZPZV<2>, ZPZV<65»; };  // NOLINT
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
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
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<2»; };  // NOLINT
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<25>, ZPZV<49>, ZPZV<55>, ZPZV<65»; };  // NOLINT
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»; };  // NOLINT
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<9>, ZPZV<65»; };  // NOLINT
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<21>, ZPZV<27>, ZPZV<2»; };  // NOLINT
template<> struct ConwayPolynomial<67, 13> { 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<0>, ZPZV<0>, ZPZV<22>, ZPZV<65»; };  // NOLINT
template<> struct ConwayPolynomial<67, 14> { using ZPZ = aerobus::zpz<67>; using type = POLYV<ZPZV<1>, ZPZV<0>, ZPZV<0>, ZPZV<0>, ZPZV<0>, ZPZV<0>, ZPZV<0>, ZPZV<17>, ZPZV<22>, ZPZV<5>, ZPZV<56>, ZPZV<0>, ZPZV<1>, ZPZV<37>, ZPZV<2»; };  // NOLINT
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<1>, ZPZV<1>, ZPZV<52>, ZPZV<41>, ZPZV<20>, ZPZV<21>, ZPZV<46>, ZPZV<65»; };  // NOLINT
template<> struct ConwayPolynomial<67, 17> { 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<0>, ZPZV<0>, ZPZV<0>, ZPZV<0>, ZPZV<0>, ZPZV<0>, ZPZV<5>, ZPZV<65»; };  // NOLINT
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<1>, ZPZV<63>, ZPZV<52>, ZPZV<18>, ZPZV<33>, ZPZV<55>, ZPZV<28>, ZPZV<29>, ZPZV<51>, ZPZV<6>, ZPZV<59>, ZPZV<13>, ZPZV<2»; };  // NOLINT
template<> struct ConwayPolynomial<67, 19> { 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<0>, ZPZV<0>, ZPZV<0>, ZPZV<0>, ZPZV<0>, ZPZV<0>, ZPZV<0>, ZPZV<0>, ZPZV<18>, ZPZV<65»; };  // NOLINT
template<> struct ConwayPolynomial<71, 1> { using ZPZ = aerobus::zpz<71>; using type = POLYV<ZPZV<1>, ZPZV<64»; };  // NOLINT
template<> struct ConwayPolynomial<71, 2> { using ZPZ = aerobus::zpz<71>; using type = POLYV<ZPZV<1>, ZPZV<69>, ZPZV<7»; };  // NOLINT
template<> struct ConwayPolynomial<71, 3> { using ZPZ = aerobus::zpz<71>; using type = POLYV<ZPZV<1>, ZPZV<0>, ZPZV<4>, ZPZV<64»; };  // NOLINT
template<> struct ConwayPolynomial<71, 4> { using ZPZ = aerobus::zpz<71>; using type = POLYV<ZPZV<1>, ZPZV<0>, ZPZV<4>, ZPZV<41>, ZPZV<7»; };  // NOLINT
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
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
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
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
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<62>, ZPZV<64»; };  // NOLINT
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»; };  // NOLINT
template<> struct ConwayPolynomial<71, 11> { 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<48>, ZPZV<64»; };  // NOLINT
template<> struct ConwayPolynomial<71, 12> { using ZPZ = aerobus::zpz<71>; using type = POLYV<ZPZV<1>, ZPZV<0>, ZPZV<0>, ZPZV<0>, ZPZV<0>, ZPZV<12>, ZPZV<28>, ZPZV<29>, ZPZV<55>, ZPZV<21>, ZPZV<58>, ZPZV<23>, ZPZV<7»; };  // NOLINT
template<> struct ConwayPolynomial<71, 13> { 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<0>, ZPZV<0>, ZPZV<27>, ZPZV<64»; };  // NOLINT
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<28>, ZPZV<32>, ZPZV<18>, ZPZV<52>, ZPZV<67>, ZPZV<49>, ZPZV<64»; };  // NOLINT
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<0>, ZPZV<0>, ZPZV<0>, ZPZV<0>, ZPZV<0>, ZPZV<0>, ZPZV<0>, ZPZV<0>, ZPZV<0>, ZPZV<3>, ZPZV<64»; };  // NOLINT
template<> struct ConwayPolynomial<71, 19> { 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<0>, ZPZV<0>, ZPZV<0>, ZPZV<0>, ZPZV<0>, ZPZV<0>, ZPZV<0>, ZPZV<0>, ZPZV<4>, ZPZV<64»; };  // NOLINT
template<> struct ConwayPolynomial<73, 1> { using ZPZ = aerobus::zpz<73>; using type = POLYV<ZPZV<1>, ZPZV<68»; };  // NOLINT
template<> struct ConwayPolynomial<73, 2> { using ZPZ = aerobus::zpz<73>; using type = POLYV<ZPZV<1>, ZPZV<70>, ZPZV<5»; };  // NOLINT
template<> struct ConwayPolynomial<73, 3> { using ZPZ = aerobus::zpz<73>; using type = POLYV<ZPZV<1>, ZPZV<0>, ZPZV<2>, ZPZV<68»; };  // NOLINT
template<> struct ConwayPolynomial<73, 4> { using ZPZ = aerobus::zpz<73>; using type = POLYV<ZPZV<1>, ZPZV<0>, ZPZV<16>, ZPZV<56>, ZPZV<5»; };  // NOLINT
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
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
template<> struct ConwayPolynomial<73, 7> { using ZPZ = aerobus::zpz<73>; using type = POLYV<ZPZV<1>, ZPZV<0>, ZPZV<0>, ZPZV<0>, ZPZV<0>, ZPZV<0>, ZPZV<10>, ZPZV<68»; };  // NOLINT
template<> struct ConwayPolynomial<73, 8> { using ZPZ = aerobus::zpz<73>; using type = POLYV<ZPZV<1>, ZPZV<0>, ZPZV<0>, ZPZV<0>, ZPZV<3>, ZPZV<53>, ZPZV<39>, ZPZV<18>, ZPZV<5»; };  // NOLINT
template<> struct ConwayPolynomial<73, 9> { using ZPZ = aerobus::zpz<73>; using type = POLYV<ZPZV<1>, ZPZV<0>, ZPZV<0>, ZPZV<0>, ZPZV<0>, ZPZV<0>, ZPZV<0>, ZPZV<72>, ZPZV<15>, ZPZV<68»; };  // NOLINT
template<> struct ConwayPolynomial<73, 10> { using ZPZ = aerobus::zpz<73>; using type = POLYV<ZPZV<1>, ZPZV<0>, ZPZV<0>, ZPZV<0>, ZPZV<2>, ZPZV<15>, ZPZV<23>, ZPZV<33>, ZPZV<32>, ZPZV<69>, ZPZV<5»; };  //
```

```
NOLINT
template<> struct ConwayPolynomial<73, 11> { 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<5>, ZPZV<68>»; }; // NOLINT
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
template<> struct ConwayPolynomial<73, 13> { 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<0>, ZPZV<0>, ZPZV<7>, ZPZV<68>»; };  // NOLINT
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<10>, ZPZV<33>, ZPZV<57>, ZPZV<57>, ZPZV<62>, ZPZV<68>»; };  // NOLINT
template<> struct ConwayPolynomial<73, 17> { 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<0>, ZPZV<0>, ZPZV<0>, ZPZV<0>, ZPZV<0>, ZPZV<0>, ZPZV<8>, ZPZV<68>»; };  // NOLINT
template<> struct ConwayPolynomial<73, 19> { 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<0>, ZPZV<0>, ZPZV<0>, ZPZV<0>, ZPZV<0>, ZPZV<0>, ZPZV<0>, ZPZV<0>, ZPZV<25>, ZPZV<68>»; };  // NOLINT
template<> struct ConwayPolynomial<79, 1> { using ZPZ = aerobus::zpz<79>; using type = POLYV<ZPZV<1>, ZPZV<76>»; };  // NOLINT
template<> struct ConwayPolynomial<79, 2> { using ZPZ = aerobus::zpz<79>; using type = POLYV<ZPZV<1>, ZPZV<78>, ZPZV<3>»; };  // NOLINT
template<> struct ConwayPolynomial<79, 3> { using ZPZ = aerobus::zpz<79>; using type = POLYV<ZPZV<1>, ZPZV<0>, ZPZV<9>, ZPZV<76>»; };  // NOLINT
template<> struct ConwayPolynomial<79, 4> { using ZPZ = aerobus::zpz<79>; using type = POLYV<ZPZV<1>, ZPZV<0>, ZPZV<2>, ZPZV<66>, ZPZV<3>»; };  // NOLINT
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
template<> struct ConwayPolynomial<79, 6> { using ZPZ = aerobus::zpz<79>; using type = POLYV<ZPZV<1>, ZPZV<0>, ZPZV<0>, ZPZV<19>, ZPZV<28>, ZPZV<68>, ZPZV<3>»; };  // NOLINT
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
template<> struct ConwayPolynomial<79, 8> { using ZPZ = aerobus::zpz<79>; using type = POLYV<ZPZV<1>, ZPZV<0>, ZPZV<0>, ZPZV<0>, ZPZV<0>, ZPZV<60>, ZPZV<59>, ZPZV<48>, ZPZV<3>»; };  // NOLINT
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<19>, ZPZV<76>»; };  // NOLINT
template<> struct ConwayPolynomial<79, 10> { using ZPZ = aerobus::zpz<79>; using type = POLYV<ZPZV<1>, ZPZV<0>, ZPZV<0>, ZPZV<0>, ZPZV<4>, ZPZV<44>, ZPZV<51>, ZPZV<1>, ZPZV<30>, ZPZV<42>, ZPZV<3>»; };  // NOLINT
template<> struct ConwayPolynomial<79, 11> { using ZPZ = aerobus::zpz<79>; 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<76>»; }; // NOLINT
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
template<> struct ConwayPolynomial<79, 13> { using ZPZ = aerobus::zpz<79>; 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<78>, ZPZV<4>, ZPZV<76>»; };  // NOLINT
template<> struct ConwayPolynomial<79, 17> { using ZPZ = aerobus::zpz<79>; 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<25>, ZPZV<76>»; };  // NOLINT
template<> struct ConwayPolynomial<79, 19> { using ZPZ = aerobus::zpz<79>; 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<25>, ZPZV<76>»; };  // NOLINT
template<> struct ConwayPolynomial<83, 1> { using ZPZ = aerobus::zpz<83>; using type = POLYV<ZPZV<1>, ZPZV<81>»; };  // NOLINT
template<> struct ConwayPolynomial<83, 2> { using ZPZ = aerobus::zpz<83>; using type = POLYV<ZPZV<1>, ZPZV<82>, ZPZV<2>»; };  // NOLINT
template<> struct ConwayPolynomial<83, 3> { using ZPZ = aerobus::zpz<83>; using type = POLYV<ZPZV<1>, ZPZV<0>, ZPZV<3>, ZPZV<81>»; };  // NOLINT
template<> struct ConwayPolynomial<83, 4> { using ZPZ = aerobus::zpz<83>; using type = POLYV<ZPZV<1>, ZPZV<0>, ZPZV<4>, ZPZV<42>, ZPZV<2>»; };  // NOLINT
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
template<> struct ConwayPolynomial<83, 6> { using ZPZ = aerobus::zpz<83>; using type = POLYV<ZPZV<1>, ZPZV<0>, ZPZV<1>, ZPZV<76>, ZPZV<32>, ZPZV<17>, ZPZV<2>»; };  // NOLINT
template<> struct ConwayPolynomial<83, 7> { using ZPZ = aerobus::zpz<83>; using type = POLYV<ZPZV<1>, ZPZV<0>, ZPZV<0>, ZPZV<0>, ZPZV<0>, ZPZV<0>, ZPZV<3>, ZPZV<81>»; };  // NOLINT
template<> struct ConwayPolynomial<83, 8> { using ZPZ = aerobus::zpz<83>; using type = POLYV<ZPZV<1>, ZPZV<0>, ZPZV<0>, ZPZV<0>, ZPZV<1>, ZPZV<65>, ZPZV<23>, ZPZV<42>, ZPZV<2>»; };  // NOLINT
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<1>, ZPZV<24>, ZPZV<18>, ZPZV<81>»; };  // NOLINT
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<0>, ZPZV<73>, ZPZV<0>, ZPZV<53>, ZPZV<2>»; };  // NOLINT
template<> struct ConwayPolynomial<83, 11> { using ZPZ = aerobus::zpz<83>; 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>, ZPZV<81>»; };  // NOLINT
template<> struct ConwayPolynomial<83, 12> { using ZPZ = aerobus::zpz<83>; using type = POLYV<ZPZV<1>, ZPZV<0>, ZPZV<0>, ZPZV<0>, ZPZV<0>, ZPZV<35>, ZPZV<12>, ZPZV<31>, ZPZV<19>, ZPZV<65>, ZPZV<55>, ZPZV<75>, ZPZV<2>»; };  // NOLINT
template<> struct ConwayPolynomial<83, 13> { using ZPZ = aerobus::zpz<83>; 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<15>, ZPZV<81>»; };  // NOLINT
template<> struct ConwayPolynomial<83, 17> { using ZPZ = aerobus::zpz<83>; 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<81»; };  // NOLINT
template<> struct ConwayPolynomial<83, 19> { using ZPZ = aerobus::zpz<83>; 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<47>, ZPZV<81»; };  // NOLINT
template<> struct ConwayPolynomial<89, 1> { using ZPZ = aerobus::zpz<89>; using type = POLYV<ZPZV<1>,
ZPZV<86»; };  // NOLINT
template<> struct ConwayPolynomial<89, 2> { using ZPZ = aerobus::zpz<89>; using type = POLYV<ZPZV<1>,
ZPZV<82>, ZPZV<3»; };  // NOLINT
template<> struct ConwayPolynomial<89, 3> { using ZPZ = aerobus::zpz<89>; using type = POLYV<ZPZV<1>,
ZPZV<0>, ZPZV<3>, ZPZV<86»; };  // NOLINT
template<> struct ConwayPolynomial<89, 4> { using ZPZ = aerobus::zpz<89>; using type = POLYV<ZPZV<1>,
ZPZV<0>, ZPZV<4>, ZPZV<72>, ZPZV<3»; };  // NOLINT
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
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
template<> struct ConwayPolynomial<89, 7> { using ZPZ = aerobus::zpz<89>; using type = POLYV<ZPZV<1>,
ZPZV<0>, ZPZV<0>, ZPZV<0>, ZPZV<0>, ZPZV<0>, ZPZV<7>, ZPZV<86»; };  // NOLINT
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
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<12>, ZPZV<6>, ZPZV<86»; };  // NOLINT
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»; };  //
NOLINT
template<> struct ConwayPolynomial<89, 11> { using ZPZ = aerobus::zpz<89>; using type = POLYV<ZPZV<1>,
ZPZV<0>, ZPZV<0>, ZPZV<0>, ZPZV<0>, ZPZV<0>, ZPZV<0>, ZPZV<0>, ZPZV<0>, ZPZV<88>, ZPZV<26>, ZPZV<86»;
};  // NOLINT
template<> struct ConwayPolynomial<89, 12> { using ZPZ = aerobus::zpz<89>; using type = POLYV<ZPZV<1>,
ZPZV<0>, ZPZV<0>, ZPZV<2>, ZPZV<85>, ZPZV<15>, ZPZV<44>, ZPZV<51>, ZPZV<8>, ZPZV<70>,
ZPZV<52>, ZPZV<3»; };  // NOLINT
template<> struct ConwayPolynomial<89, 13> { using ZPZ = aerobus::zpz<89>; 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<17>, ZPZV<86»; };  // NOLINT
template<> struct ConwayPolynomial<89, 17> { using ZPZ = aerobus::zpz<89>; 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<20>, ZPZV<86»; };  // NOLINT
template<> struct ConwayPolynomial<89, 19> { using ZPZ = aerobus::zpz<89>; 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<34>, ZPZV<86»; };  // NOLINT
template<> struct ConwayPolynomial<97, 1> { using ZPZ = aerobus::zpz<97>; using type = POLYV<ZPZV<1>,
ZPZV<92»; };  // NOLINT
template<> struct ConwayPolynomial<97, 2> { using ZPZ = aerobus::zpz<97>; using type = POLYV<ZPZV<1>,
ZPZV<96>, ZPZV<5»; };  // NOLINT
template<> struct ConwayPolynomial<97, 3> { using ZPZ = aerobus::zpz<97>; using type = POLYV<ZPZV<1>,
ZPZV<0>, ZPZV<9>, ZPZV<92»; };  // NOLINT
template<> struct ConwayPolynomial<97, 4> { using ZPZ = aerobus::zpz<97>; using type = POLYV<ZPZV<1>,
ZPZV<0>, ZPZV<6>, ZPZV<80>, ZPZV<5»; };  // NOLINT
template<> struct ConwayPolynomial<97, 5> { using ZPZ = aerobus::zpz<97>; using type = POLYV<ZPZV<1>,
ZPZV<0>, ZPZV<0>, ZPZV<0>, ZPZV<3>, ZPZV<92»; };  // NOLINT
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
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
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
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<12>, ZPZV<59>, ZPZV<7>, ZPZV<92»; };  // NOLINT
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»; };  //
NOLINT
template<> struct ConwayPolynomial<97, 11> { 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<0>, ZPZV<0>, ZPZV<5>, ZPZV<92»; };
// NOLINT
template<> struct ConwayPolynomial<97, 12> { using ZPZ = aerobus::zpz<97>; using type = POLYV<ZPZV<1>,
ZPZV<0>, ZPZV<0>, ZPZV<0>, ZPZV<0>, ZPZV<30>, ZPZV<59>, ZPZV<81>, ZPZV<0>, ZPZV<86>, ZPZV<78>,
ZPZV<94>, ZPZV<5»; };  // NOLINT
template<> struct ConwayPolynomial<97, 13> { 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<0>, ZPZV<0>, ZPZV<0>, ZPZV<0>,
ZPZV<3>, ZPZV<92»; };  // NOLINT
template<> struct ConwayPolynomial<97, 17> { 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<0>, ZPZV<0>, ZPZV<0>, ZPZV<0>,
ZPZV<0>, ZPZV<0>, ZPZV<0>, ZPZV<0>, ZPZV<5>, ZPZV<92»; };  // NOLINT
template<> struct ConwayPolynomial<97, 19> { 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<0>, ZPZV<0>, ZPZV<0>, ZPZV<0>,
ZPZV<0>, ZPZV<0>, ZPZV<0>, ZPZV<0>, ZPZV<0>, ZPZV<0>, ZPZV<15>, ZPZV<92»; };  // NOLINT
template<> struct ConwayPolynomial<101, 1> { using ZPZ = aerobus::zpz<101>; using type =
POLYV<ZPZV<1>, ZPZV<99»; };  // NOLINT
template<> struct ConwayPolynomial<101, 2> { using ZPZ = aerobus::zpz<101>; using type =
POLYV<ZPZV<1>, ZPZV<97>, ZPZV<2»; };  // NOLINT
template<> struct ConwayPolynomial<101, 3> { using ZPZ = aerobus::zpz<101>; using type =
POLYV<ZPZV<1>, ZPZV<0>, ZPZV<3>, ZPZV<99»; };  // NOLINT
template<> struct ConwayPolynomial<101, 4> { using ZPZ = aerobus::zpz<101>; using type =
POLYV<ZPZV<1>, ZPZV<0>, ZPZV<1>, ZPZV<78>, ZPZV<2»; };  // NOLINT
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
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
template<> struct ConwayPolynomial<101, 7> { using ZPZ = aerobus::zpz<101>; using type =
POLYV<ZPZV<1>, ZPZV<0>, ZPZV<0>, ZPZV<0>, ZPZV<0>, ZPZV<0>, ZPZV<6>, ZPZV<99»; };  // NOLINT
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
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<64>, ZPZV<47>, ZPZV<99»; };
// NOLINT
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
template<> struct ConwayPolynomial<101, 11> { 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<31>, ZPZV<99»; };  // NOLINT
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<21>, ZPZV<2»; };  // NOLINT
template<> struct ConwayPolynomial<101, 13> { 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<7>, ZPZV<99»; };  // NOLINT
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<0>, ZPZV<0>, ZPZV<0>, ZPZV<0>, ZPZV<31>, ZPZV<99»; };  // NOLINT
template<> struct ConwayPolynomial<101, 19> { 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<0>, ZPZV<0>, ZPZV<0>, ZPZV<0>, ZPZV<0>, ZPZV<0>, ZPZV<24>, ZPZV<99»; };  //
NOLINT
template<> struct ConwayPolynomial<103, 1> { using ZPZ = aerobus::zpz<103>; using type =
POLYV<ZPZV<1>, ZPZV<98»; };  // NOLINT
template<> struct ConwayPolynomial<103, 2> { using ZPZ = aerobus::zpz<103>; using type =
POLYV<ZPZV<1>, ZPZV<102>, ZPZV<5»; };  // NOLINT
template<> struct ConwayPolynomial<103, 3> { using ZPZ = aerobus::zpz<103>; using type =
POLYV<ZPZV<1>, ZPZV<0>, ZPZV<2>, ZPZV<98»; };  // NOLINT
template<> struct ConwayPolynomial<103, 4> { using ZPZ = aerobus::zpz<103>; using type =
POLYV<ZPZV<1>, ZPZV<0>, ZPZV<2>, ZPZV<88>, ZPZV<5»; };  // NOLINT
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
template<> struct ConwayPolynomial<103, 6> { using ZPZ = aerobus::zpz<103>; using type =
POLYV<ZPZV<1>, ZPZV<0>, ZPZV<0>, ZPZV<96>, ZPZV<9>, ZPZV<30>, ZPZV<5»; };  // NOLINT
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»; };  // NOLINT
template<> struct ConwayPolynomial<103, 8> { using ZPZ = aerobus::zpz<103>; using type =
POLYV<ZPZV<1>, ZPZV<0>, ZPZV<0>, ZPZV<0>, ZPZV<1>, ZPZV<70>, ZPZV<71>, ZPZV<49>, ZPZV<5»; };  //
NOLINT
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<51>, ZPZV<98»; };
// NOLINT
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
template<> struct ConwayPolynomial<103, 11> { 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<5>, ZPZV<98»; };  // NOLINT
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»; };  // NOLINT
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<5>, ZPZV<98»; };  // NOLINT
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<0>, ZPZV<0>,
ZPZV<0>, ZPZV<0>, ZPZV<0>, ZPZV<0>, ZPZV<0>, ZPZV<102>, ZPZV<8>, ZPZV<98»; };  // NOLINT
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>, ZPZV<0>, ZPZV<0>, ZPZV<0>, ZPZV<0>, ZPZV<0>, ZPZV<0>, ZPZV<0>, ZPZV<2>, ZPZV<98»; };  //
NOLINT
template<> struct ConwayPolynomial<107, 1> { using ZPZ = aerobus::zpz<107>; using type =
POLYV<ZPZV<1>, ZPZV<105»; };  // NOLINT
template<> struct ConwayPolynomial<107, 2> { using ZPZ = aerobus::zpz<107>; using type =
POLYV<ZPZV<1>, ZPZV<103>, ZPZV<2»; };  // NOLINT
template<> struct ConwayPolynomial<107, 3> { using ZPZ = aerobus::zpz<107>; using type =
POLYV<ZPZV<1>, ZPZV<0>, ZPZV<5>, ZPZV<105»; };  // NOLINT
template<> struct ConwayPolynomial<107, 4> { using ZPZ = aerobus::zpz<107>; using type =
POLYV<ZPZV<1>, ZPZV<0>, ZPZV<13>, ZPZV<79>, ZPZV<2»; };  // NOLINT
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
template<> struct ConwayPolynomial<107, 6> { using ZPZ = aerobus::zpz<107>; using type =
POLYV<ZPZV<1>, ZPZV<0>, ZPZV<1>, ZPZV<52>, ZPZV<22>, ZPZV<79>, ZPZV<2»; };  // NOLINT
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<105»; };  // NOLINT
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<2»; };  //
NOLINT
template<> struct ConwayPolynomial<107, 9> { using ZPZ = aerobus::zpz<107>; using type =
POLYV<ZPZV<1>, ZPZV<0>, ZPZV<0>, ZPZV<0>, ZPZV<0>, ZPZV<0>, ZPZV<3>, ZPZV<3>, ZPZV<66>, ZPZV<105»; };
// NOLINT
```

```
template<> struct ConwayPolynomial<107, 10> { using ZPZ = aerobus::zpz<107>; using type = POLYV<ZPZV<1>, ZPZV<0>, ZPZV<0>, ZPZV<0>, ZPZV<2>, ZPZV<94>, ZPZV<61>, ZPZV<83>, ZPZV<83>, ZPZV<95>, ZPZV<2>»; };  // NOLINT
template<> struct ConwayPolynomial<107, 11> { 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<8>, ZPZV<105>»; };  // NOLINT
template<> struct ConwayPolynomial<107, 12> { using ZPZ = aerobus::zpz<107>; using type = POLYV<ZPZV<1>, ZPZV<0>, ZPZV<0>, ZPZV<0>, ZPZV<0>, ZPZV<37>, ZPZV<48>, ZPZV<6>, ZPZV<0>, ZPZV<61>, ZPZV<42>, ZPZV<57>, ZPZV<2>»; };  // NOLINT
template<> struct ConwayPolynomial<107, 13> { 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<0>, ZPZV<0>, ZPZV<4>, ZPZV<105>»; };  // NOLINT
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<0>, ZPZV<0>, ZPZV<0>, ZPZV<0>, ZPZV<0>, ZPZV<0>, ZPZV<2>, ZPZV<105>»; };  // NOLINT
template<> struct ConwayPolynomial<107, 19> { 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<0>, ZPZV<0>, ZPZV<0>, ZPZV<0>, ZPZV<0>, ZPZV<0>, ZPZV<0>, ZPZV<0>, ZPZV<24>, ZPZV<105>»; };  // NOLINT
template<> struct ConwayPolynomial<109, 1> { using ZPZ = aerobus::zpz<109>; using type = POLYV<ZPZV<1>, ZPZV<103>»; };  // NOLINT
template<> struct ConwayPolynomial<109, 2> { using ZPZ = aerobus::zpz<109>; using type = POLYV<ZPZV<1>, ZPZV<108>, ZPZV<6>»; };  // NOLINT
template<> struct ConwayPolynomial<109, 3> { using ZPZ = aerobus::zpz<109>; using type = POLYV<ZPZV<1>, ZPZV<0>, ZPZV<1>, ZPZV<103>»; };  // NOLINT
template<> struct ConwayPolynomial<109, 4> { using ZPZ = aerobus::zpz<109>; using type = POLYV<ZPZV<1>, ZPZV<0>, ZPZV<11>, ZPZV<98>, ZPZV<6>»; };  // NOLINT
template<> struct ConwayPolynomial<109, 5> { using ZPZ = aerobus::zpz<109>; using type = POLYV<ZPZV<1>, ZPZV<0>, ZPZV<0>, ZPZV<0>, ZPZV<4>, ZPZV<103>»; };  // NOLINT
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<6>»; };  // NOLINT
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<103>»; };  // NOLINT
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
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<93>, ZPZV<87>, ZPZV<103>»; };  // NOLINT
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
template<> struct ConwayPolynomial<109, 11> { 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<0>, ZPZV<0>, ZPZV<11>, ZPZV<103>»; };  // NOLINT
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<8>, ZPZV<65>, ZPZV<103>, ZPZV<28>, ZPZV<6>»; };  // NOLINT
template<> struct ConwayPolynomial<109, 13> { 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<0>, ZPZV<0>, ZPZV<0>, ZPZV<0>, ZPZV<1>, ZPZV<103>»; };  // NOLINT
template<> struct ConwayPolynomial<109, 17> { 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<0>, ZPZV<0>, ZPZV<0>, ZPZV<0>, ZPZV<0>, ZPZV<0>, ZPZV<0>, ZPZV<0>, ZPZV<1>, ZPZV<103>»; };  // NOLINT
template<> struct ConwayPolynomial<109, 19> { 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<0>, ZPZV<0>, ZPZV<0>, ZPZV<0>, ZPZV<0>, ZPZV<0>, ZPZV<0>, ZPZV<0>, ZPZV<0>, ZPZV<0>, ZPZV<15>, ZPZV<103>»; };  // NOLINT
template<> struct ConwayPolynomial<113, 1> { using ZPZ = aerobus::zpz<113>; using type = POLYV<ZPZV<1>, ZPZV<110>»; };  // NOLINT
template<> struct ConwayPolynomial<113, 2> { using ZPZ = aerobus::zpz<113>; using type = POLYV<ZPZV<1>, ZPZV<101>, ZPZV<3>»; };  // NOLINT
template<> struct ConwayPolynomial<113, 3> { using ZPZ = aerobus::zpz<113>; using type = POLYV<ZPZV<1>, ZPZV<0>, ZPZV<8>, ZPZV<110>»; };  // NOLINT
template<> struct ConwayPolynomial<113, 4> { using ZPZ = aerobus::zpz<113>; using type = POLYV<ZPZV<1>, ZPZV<0>, ZPZV<0>, ZPZV<62>, ZPZV<3>»; };  // NOLINT
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
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
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<110>»; };  // NOLINT
template<> struct ConwayPolynomial<113, 8> { using ZPZ = aerobus::zpz<113>; using type = POLYV<ZPZV<1>, ZPZV<0>, ZPZV<0>, ZPZV<0>, ZPZV<3>, ZPZV<98>, ZPZV<38>, ZPZV<28>, ZPZV<3>»; };  // NOLINT
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<87>, ZPZV<71>, ZPZV<110>»; };  // NOLINT
template<> struct ConwayPolynomial<113, 10> { using ZPZ = aerobus::zpz<113>; using type = POLYV<ZPZV<1>, ZPZV<0>, ZPZV<0>, ZPZV<0>, ZPZV<0>, ZPZV<108>, ZPZV<57>, ZPZV<45>, ZPZV<83>, ZPZV<56>, ZPZV<3>»; };  // NOLINT
template<> struct ConwayPolynomial<113, 11> { 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<3>, ZPZV<110>»; };  // NOLINT
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<4>, ZPZV<98>, ZPZV<56>, ZPZV<10>, ZPZV<27>, ZPZV<3>»; };  // NOLINT
```

```
template<> struct ConwayPolynomial<113, 13> { 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<0>, ZPZV<4>, ZPZV<110»; };  // NOLINT
template<> struct ConwayPolynomial<113, 17> { 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<0>, ZPZV<0>, ZPZV<0>, ZPZV<0>, ZPZV<0>, ZPZV<4>, ZPZV<110»; };  // NOLINT
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<0>, ZPZV<0>, ZPZV<0>, ZPZV<0>, ZPZV<0>, ZPZV<0>, ZPZV<0>, ZPZV<22>, ZPZV<110»; };  //
NOLINT
template<> struct ConwayPolynomial<127, 1> { using ZPZ = aerobus::zpz<127>; using type =
POLYV<ZPZV<1>, ZPZV<124»; };  // NOLINT
template<> struct ConwayPolynomial<127, 2> { using ZPZ = aerobus::zpz<127>; using type =
POLYV<ZPZV<1>, ZPZV<126>, ZPZV<3»; };  // NOLINT
template<> struct ConwayPolynomial<127, 3> { using ZPZ = aerobus::zpz<127>; using type =
POLYV<ZPZV<1>, ZPZV<0>, ZPZV<3>, ZPZV<124»; };  // NOLINT
template<> struct ConwayPolynomial<127, 4> { using ZPZ = aerobus::zpz<127>; using type =
POLYV<ZPZV<1>, ZPZV<0>, ZPZV<2>, ZPZV<97>, ZPZV<3»; };  // NOLINT
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
template<> struct ConwayPolynomial<127, 6> { using ZPZ = aerobus::zpz<127>; using type =
POLYV<ZPZV<1>, ZPZV<0>, ZPZV<0>, ZPZV<84>, ZPZV<115>, ZPZV<82>, ZPZV<3»; };  // NOLINT
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<15>, ZPZV<124»; };  // NOLINT
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»; };  //
NOLINT
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<14>, ZPZV<119>, ZPZV<126>, ZPZV<124»;
};  // NOLINT
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<95>, ZPZV<60>, ZPZV<4>,
ZPZV<3»; };  // NOLINT
template<> struct ConwayPolynomial<127, 11> { 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<11>, ZPZV<124»; };  // NOLINT
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»; };  // NOLINT
template<> struct ConwayPolynomial<127, 13> { 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<8>, ZPZV<124»; };  // NOLINT
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<0>,
ZPZV<0>, ZPZV<0>, ZPZV<0>, ZPZV<0>, ZPZV<0>, ZPZV<0>, ZPZV<9>, ZPZV<124»; };  // NOLINT
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<0>, ZPZV<0>, ZPZV<30>, ZPZV<124»; };  //
NOLINT
template<> struct ConwayPolynomial<131, 1> { using ZPZ = aerobus::zpz<131>; using type =
POLYV<ZPZV<1>, ZPZV<129»; };  // NOLINT
template<> struct ConwayPolynomial<131, 2> { using ZPZ = aerobus::zpz<131>; using type =
POLYV<ZPZV<1>, ZPZV<127>, ZPZV<2»; };  // NOLINT
template<> struct ConwayPolynomial<131, 3> { using ZPZ = aerobus::zpz<131>; using type =
POLYV<ZPZV<1>, ZPZV<0>, ZPZV<3>, ZPZV<129»; };  // NOLINT
template<> struct ConwayPolynomial<131, 4> { using ZPZ = aerobus::zpz<131>; using type =
POLYV<ZPZV<1>, ZPZV<0>, ZPZV<9>, ZPZV<109>, ZPZV<2»; };  // NOLINT
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
template<> struct ConwayPolynomial<131, 6> { using ZPZ = aerobus::zpz<131>; using type =
POLYV<ZPZV<1>, ZPZV<0>, ZPZV<2>, ZPZV<66>, ZPZV<4>, ZPZV<22>, ZPZV<2»; };  // NOLINT
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<129»; };  // NOLINT
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
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<19>, ZPZV<129»; };
// NOLINT
template<> struct ConwayPolynomial<131, 10> { using ZPZ = aerobus::zpz<131>; using type =
POLYV<ZPZV<1>, ZPZV<0>, ZPZV<0>, ZPZV<0>, ZPZV<0>, ZPZV<124>, ZPZV<97>, ZPZV<9>, ZPZV<126>, ZPZV<44>,
ZPZV<2»; };  // NOLINT
template<> struct ConwayPolynomial<131, 11> { using ZPZ = aerobus::zpz<131>; 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<129»; };  // NOLINT
template<> struct ConwayPolynomial<131, 12> { using ZPZ = aerobus::zpz<131>; using type =
POLYV<ZPZV<1>, ZPZV<0>, ZPZV<0>, ZPZV<0>, ZPZV<0>, ZPZV<50>, ZPZV<122>, ZPZV<40>, ZPZV<83>, ZPZV<125>,
ZPZV<28>, ZPZV<103>, ZPZV<2»; };  // NOLINT
template<> struct ConwayPolynomial<131, 13> { using ZPZ = aerobus::zpz<131>; 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<9>, ZPZV<129»; };  // NOLINT
template<> struct ConwayPolynomial<131, 17> { using ZPZ = aerobus::zpz<131>; 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<6>, ZPZV<129»; };  // NOLINT
template<> struct ConwayPolynomial<131, 19> { using ZPZ = aerobus::zpz<131>; 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<9>, ZPZV<129»; };  //
```

```
NOLINT
template<> struct ConwayPolynomial<137, 1> { using ZPZ = aerobus::zpz<137>; using type = POLYV<ZPZV<1>, ZPZV<134»; };  // NOLINT
template<> struct ConwayPolynomial<137, 2> { using ZPZ = aerobus::zpz<137>; using type = POLYV<ZPZV<1>, ZPZV<131>, ZPZV<3»; };  // NOLINT
template<> struct ConwayPolynomial<137, 3> { using ZPZ = aerobus::zpz<137>; using type = POLYV<ZPZV<1>, ZPZV<0>, ZPZV<6>, ZPZV<134»; };  // NOLINT
template<> struct ConwayPolynomial<137, 4> { using ZPZ = aerobus::zpz<137>; using type = POLYV<ZPZV<1>, ZPZV<0>, ZPZV<1>, ZPZV<95>, ZPZV<3»; };  // NOLINT
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
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
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»; };  // NOLINT
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
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<80>, ZPZV<122>, ZPZV<134»; };  // NOLINT
template<> struct ConwayPolynomial<137, 10> { using ZPZ = aerobus::zpz<137>; using type = POLYV<ZPZV<1>, ZPZV<0>, ZPZV<0>, ZPZV<0>, ZPZV<2>, ZPZV<0>, ZPZV<20>, ZPZV<67>, ZPZV<93>, ZPZV<119>, ZPZV<3»; };  // NOLINT
template<> struct ConwayPolynomial<137, 11> { 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<1>, ZPZV<134»; };  // NOLINT
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<12>, ZPZV<36>, ZPZV<135>, ZPZV<61>, ZPZV<3»; };  // NOLINT
template<> struct ConwayPolynomial<137, 13> { 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<14>, ZPZV<134»; };  // NOLINT
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<136>, ZPZV<4>, ZPZV<134»; };  // NOLINT
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<0>, ZPZV<0>, ZPZV<0>, ZPZV<0>, ZPZV<0>, ZPZV<0>, ZPZV<0>, ZPZV<0>, ZPZV<0>, ZPZV<18>, ZPZV<134»; };  // NOLINT
template<> struct ConwayPolynomial<139, 1> { using ZPZ = aerobus::zpz<139>; using type = POLYV<ZPZV<1>, ZPZV<137»; };  // NOLINT
template<> struct ConwayPolynomial<139, 2> { using ZPZ = aerobus::zpz<139>; using type = POLYV<ZPZV<1>, ZPZV<138>, ZPZV<2»; };  // NOLINT
template<> struct ConwayPolynomial<139, 3> { using ZPZ = aerobus::zpz<139>; using type = POLYV<ZPZV<1>, ZPZV<0>, ZPZV<6>, ZPZV<137»; };  // NOLINT
template<> struct ConwayPolynomial<139, 4> { using ZPZ = aerobus::zpz<139>; using type = POLYV<ZPZV<1>, ZPZV<0>, ZPZV<7>, ZPZV<96>, ZPZV<2»; };  // NOLINT
template<> struct ConwayPolynomial<139, 5> { using ZPZ = aerobus::zpz<139>; using type = POLYV<ZPZV<1>, ZPZV<0>, ZPZV<0>, ZPZV<0>, ZPZV<10>, ZPZV<137»; };  // NOLINT
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
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<9>, ZPZV<137»; };  // NOLINT
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»; };  // NOLINT
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<87>, ZPZV<137»; };  // NOLINT
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»; };  // NOLINT
template<> struct ConwayPolynomial<139, 11> { 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<7>, ZPZV<137»; };  // NOLINT
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<10>, ZPZV<2»; };  // NOLINT
template<> struct ConwayPolynomial<139, 13> { 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<0>, ZPZV<22>, ZPZV<137»; };  // NOLINT
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<0>, ZPZV<0>, ZPZV<0>, ZPZV<0>, ZPZV<0>, ZPZV<2>, ZPZV<137»; };  // NOLINT
template<> struct ConwayPolynomial<139, 19> { 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<0>, ZPZV<0>, ZPZV<0>, ZPZV<0>, ZPZV<0>, ZPZV<0>, ZPZV<0>, ZPZV<23>, ZPZV<137»; };  // NOLINT
template<> struct ConwayPolynomial<149, 1> { using ZPZ = aerobus::zpz<149>; using type = POLYV<ZPZV<1>, ZPZV<147»; };  // NOLINT
template<> struct ConwayPolynomial<149, 2> { using ZPZ = aerobus::zpz<149>; using type = POLYV<ZPZV<1>, ZPZV<145>, ZPZV<2»; };  // NOLINT
template<> struct ConwayPolynomial<149, 3> { using ZPZ = aerobus::zpz<149>; using type = POLYV<ZPZV<1>, ZPZV<0>, ZPZV<3>, ZPZV<147»; };  // NOLINT
template<> struct ConwayPolynomial<149, 4> { using ZPZ = aerobus::zpz<149>; using type = POLYV<ZPZV<1>, ZPZV<0>, ZPZV<7>, ZPZV<107>, ZPZV<2»; };  // NOLINT
```

```
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
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
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
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<2»; };  // NOLINT
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<0>, ZPZV<146>, ZPZV<20>, ZPZV<147»; };  // NOLINT
template<> struct ConwayPolynomial<149, 10> { using ZPZ = aerobus::zpz<149>; using type = POLYV<ZPZV<1>, ZPZV<0>, ZPZV<0>, ZPZV<0>, ZPZV<0>, ZPZV<74>, ZPZV<42>, ZPZV<148>, ZPZV<143>, ZPZV<51>, ZPZV<2»; };  // NOLINT
template<> struct ConwayPolynomial<149, 11> { 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<0>, ZPZV<33>, ZPZV<147»; };  // NOLINT
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<121>, ZPZV<91>, ZPZV<52>, ZPZV<9>, ZPZV<104>, ZPZV<110>, ZPZV<2»; };  // NOLINT
template<> struct ConwayPolynomial<149, 13> { 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<0>, ZPZV<0>, ZPZV<0>, ZPZV<4>, ZPZV<147»; };  // NOLINT
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<0>, ZPZV<0>, ZPZV<0>, ZPZV<0>, ZPZV<0>, ZPZV<0>, ZPZV<0>, ZPZV<29>, ZPZV<147»; };  // NOLINT
template<> struct ConwayPolynomial<149, 19> { 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<0>, ZPZV<0>, ZPZV<0>, ZPZV<0>, ZPZV<0>, ZPZV<0>, ZPZV<0>, ZPZV<0>, ZPZV<0>, ZPZV<5>, ZPZV<147»; };  // NOLINT
template<> struct ConwayPolynomial<151, 1> { using ZPZ = aerobus::zpz<151>; using type = POLYV<ZPZV<1>, ZPZV<145»; };  // NOLINT
template<> struct ConwayPolynomial<151, 2> { using ZPZ = aerobus::zpz<151>; using type = POLYV<ZPZV<1>, ZPZV<149>, ZPZV<6»; };  // NOLINT
template<> struct ConwayPolynomial<151, 3> { using ZPZ = aerobus::zpz<151>; using type = POLYV<ZPZV<1>, ZPZV<0>, ZPZV<1>, ZPZV<145»; };  // NOLINT
template<> struct ConwayPolynomial<151, 4> { using ZPZ = aerobus::zpz<151>; using type = POLYV<ZPZV<1>, ZPZV<0>, ZPZV<13>, ZPZV<89>, ZPZV<6»; };  // NOLINT
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
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
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<9>, ZPZV<145»; };  // NOLINT
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
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<126>, ZPZV<96>, ZPZV<145»; };  // NOLINT
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
template<> struct ConwayPolynomial<151, 11> { 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<1>, ZPZV<145»; };  // NOLINT
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<6>, ZPZV<77>, ZPZV<107>, ZPZV<147>, ZPZV<6»; };  // NOLINT
template<> struct ConwayPolynomial<151, 13> { 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<12>, ZPZV<145»; };  // NOLINT
template<> struct ConwayPolynomial<151, 17> { 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<24>, ZPZV<145»; };  // NOLINT
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<9>, ZPZV<145»; };  // NOLINT
template<> struct ConwayPolynomial<157, 1> { using ZPZ = aerobus::zpz<157>; using type = POLYV<ZPZV<1>, ZPZV<152»; };  // NOLINT
template<> struct ConwayPolynomial<157, 2> { using ZPZ = aerobus::zpz<157>; using type = POLYV<ZPZV<1>, ZPZV<152>, ZPZV<5»; };  // NOLINT
template<> struct ConwayPolynomial<157, 3> { using ZPZ = aerobus::zpz<157>; using type = POLYV<ZPZV<1>, ZPZV<0>, ZPZV<1>, ZPZV<152»; };  // NOLINT
template<> struct ConwayPolynomial<157, 4> { using ZPZ = aerobus::zpz<157>; using type = POLYV<ZPZV<1>, ZPZV<0>, ZPZV<11>, ZPZV<136>, ZPZV<5»; };  // NOLINT
template<> struct ConwayPolynomial<157, 5> { using ZPZ = aerobus::zpz<157>; using type = POLYV<ZPZV<1>, ZPZV<0>, ZPZV<0>, ZPZV<0>, ZPZV<7>, ZPZV<152»; };  // NOLINT
template<> struct ConwayPolynomial<157, 6> { using ZPZ = aerobus::zpz<157>; using type = POLYV<ZPZV<1>, ZPZV<0>, ZPZV<3>, ZPZV<130>, ZPZV<43>, ZPZV<144>, ZPZV<5»; };  // NOLINT
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<152»; };  // NOLINT
template<> struct ConwayPolynomial<157, 8> { using ZPZ = aerobus::zpz<157>; using type = POLYV<ZPZV<1>, ZPZV<0>, ZPZV<0>, ZPZV<0>, ZPZV<3>, ZPZV<97>, ZPZV<40>, ZPZV<153>, ZPZV<5»; };  // NOLINT
```

```
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<2>, ZPZV<114>, ZPZV<52>, ZPZV<152»; };  // NOLINT
template<> struct ConwayPolynomial<157, 10> { using ZPZ = aerobus::zpz<157>; using type = POLYV<ZPZV<1>, ZPZV<0>, ZPZV<0>, ZPZV<0>, ZPZV<0>, ZPZV<61>, ZPZV<22>, ZPZV<124>, ZPZV<61>, ZPZV<93>, ZPZV<5»; };  // NOLINT
template<> struct ConwayPolynomial<157, 11> { 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>, ZPZV<0>, ZPZV<29>, ZPZV<152»; };  // NOLINT
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
template<> struct ConwayPolynomial<157, 13> { 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>, ZPZV<0>, ZPZV<0>, ZPZV<156>, ZPZV<9>, ZPZV<152»; };  // NOLINT
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>, ZPZV<0>, ZPZV<0>, ZPZV<0>, ZPZV<0>, ZPZV<0>, ZPZV<0>, ZPZV<0>, ZPZV<12>, ZPZV<152»; };  // NOLINT
template<> struct ConwayPolynomial<157, 19> { 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>, ZPZV<0>, ZPZV<0>, ZPZV<0>, ZPZV<0>, ZPZV<0>, ZPZV<0>, ZPZV<0>, ZPZV<0>, ZPZV<0>, ZPZV<14>, ZPZV<152»; };  // NOLINT
template<> struct ConwayPolynomial<163, 1> { using ZPZ = aerobus::zpz<163>; using type = POLYV<ZPZV<1>, ZPZV<161»; };  // NOLINT
template<> struct ConwayPolynomial<163, 2> { using ZPZ = aerobus::zpz<163>; using type = POLYV<ZPZV<1>, ZPZV<159>, ZPZV<2»; };  // NOLINT
template<> struct ConwayPolynomial<163, 3> { using ZPZ = aerobus::zpz<163>; using type = POLYV<ZPZV<1>, ZPZV<0>, ZPZV<7>, ZPZV<161»; };  // NOLINT
template<> struct ConwayPolynomial<163, 4> { using ZPZ = aerobus::zpz<163>; using type = POLYV<ZPZV<1>, ZPZV<0>, ZPZV<8>, ZPZV<91>, ZPZV<2»; };  // NOLINT
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
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
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<9>, ZPZV<161»; };  // NOLINT
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»; };  // NOLINT
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
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
template<> struct ConwayPolynomial<163, 11> { 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<11>, ZPZV<161»; };  // NOLINT
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<10>, ZPZV<69>, ZPZV<2»; };  // NOLINT
template<> struct ConwayPolynomial<163, 13> { 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>, ZPZV<0>, ZPZV<24>, ZPZV<161»; };  // NOLINT
template<> struct ConwayPolynomial<163, 17> { 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>, ZPZV<0>, ZPZV<0>, ZPZV<0>, ZPZV<0>, ZPZV<0>, ZPZV<71>, ZPZV<161»; };  // NOLINT
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>, ZPZV<0>, ZPZV<0>, ZPZV<0>, ZPZV<0>, ZPZV<0>, ZPZV<0>, ZPZV<0>, ZPZV<8>, ZPZV<161»; };  // NOLINT
template<> struct ConwayPolynomial<167, 1> { using ZPZ = aerobus::zpz<167>; using type = POLYV<ZPZV<1>, ZPZV<162»; };  // NOLINT
template<> struct ConwayPolynomial<167, 2> { using ZPZ = aerobus::zpz<167>; using type = POLYV<ZPZV<1>, ZPZV<166>, ZPZV<5»; };  // NOLINT
template<> struct ConwayPolynomial<167, 3> { using ZPZ = aerobus::zpz<167>; using type = POLYV<ZPZV<1>, ZPZV<0>, ZPZV<7>, ZPZV<162»; };  // NOLINT
template<> struct ConwayPolynomial<167, 4> { using ZPZ = aerobus::zpz<167>; using type = POLYV<ZPZV<1>, ZPZV<0>, ZPZV<3>, ZPZV<120>, ZPZV<5»; };  // NOLINT
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
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
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
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»; };  // NOLINT
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<165>, ZPZV<122>, ZPZV<162»; };  // NOLINT
template<> struct ConwayPolynomial<167, 10> { using ZPZ = aerobus::zpz<167>; using type = POLYV<ZPZV<1>, ZPZV<0>, ZPZV<0>, ZPZV<0>, ZPZV<85>, ZPZV<68>, ZPZV<109>, ZPZV<143>, ZPZV<148>, ZPZV<5»; };  // NOLINT
template<> struct ConwayPolynomial<167, 11> { 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<24>, ZPZV<162»; };  // NOLINT
```

```
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<131>, ZPZV<140>, ZPZV<41>, ZPZV<57>, ZPZV<5»; };  // NOLINT
template<> struct ConwayPolynomial<167, 13> { 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<0>, ZPZV<10>, ZPZV<162»; };  // NOLINT
template<> struct ConwayPolynomial<167, 17> { 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<0>, ZPZV<0>, ZPZV<0>, ZPZV<0>, ZPZV<0>, ZPZV<32>, ZPZV<162»; };  // NOLINT
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<0>, ZPZV<0>, ZPZV<0>, ZPZV<0>, ZPZV<0>, ZPZV<0>, ZPZV<0>, ZPZV<14>, ZPZV<162»; };  // NOLINT
template<> struct ConwayPolynomial<173, 1> { using ZPZ = aerobus::zpz<173>; using type = POLYV<ZPZV<1>, ZPZV<171»; };  // NOLINT
template<> struct ConwayPolynomial<173, 2> { using ZPZ = aerobus::zpz<173>; using type = POLYV<ZPZV<1>, ZPZV<169>, ZPZV<2»; };  // NOLINT
template<> struct ConwayPolynomial<173, 3> { using ZPZ = aerobus::zpz<173>; using type = POLYV<ZPZV<1>, ZPZV<0>, ZPZV<2>, ZPZV<171»; };  // NOLINT
template<> struct ConwayPolynomial<173, 4> { using ZPZ = aerobus::zpz<173>; using type = POLYV<ZPZV<1>, ZPZV<0>, ZPZV<1>, ZPZV<102>, ZPZV<2»; };  // NOLINT
template<> struct ConwayPolynomial<173, 5> { using ZPZ = aerobus::zpz<173>; using type = POLYV<ZPZV<1>, ZPZV<0>, ZPZV<0>, ZPZV<0>, ZPZV<6>, ZPZV<171»; };  // NOLINT
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
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<171»; };  // NOLINT
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
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<56>, ZPZV<104>, ZPZV<171»; };  // NOLINT
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<2»; };  // NOLINT
template<> struct ConwayPolynomial<173, 11> { 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<12>, ZPZV<171»; };  // NOLINT
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<0>, ZPZV<159>, ZPZV<22>, ZPZV<2»; };  // NOLINT
template<> struct ConwayPolynomial<173, 13> { 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<171»; };  // NOLINT
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<171»; };  // NOLINT
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<0>, ZPZV<0>, ZPZV<0>, ZPZV<0>, ZPZV<0>, ZPZV<0>, ZPZV<6>, ZPZV<171»; };  // NOLINT
template<> struct ConwayPolynomial<179, 1> { using ZPZ = aerobus::zpz<179>; using type = POLYV<ZPZV<1>, ZPZV<177»; };  // NOLINT
template<> struct ConwayPolynomial<179, 2> { using ZPZ = aerobus::zpz<179>; using type = POLYV<ZPZV<1>, ZPZV<172>, ZPZV<2»; };  // NOLINT
template<> struct ConwayPolynomial<179, 3> { using ZPZ = aerobus::zpz<179>; using type = POLYV<ZPZV<1>, ZPZV<0>, ZPZV<4>, ZPZV<177»; };  // NOLINT
template<> struct ConwayPolynomial<179, 4> { using ZPZ = aerobus::zpz<179>; using type = POLYV<ZPZV<1>, ZPZV<0>, ZPZV<1>, ZPZV<109>, ZPZV<2»; };  // NOLINT
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
template<> struct ConwayPolynomial<179, 6> { using ZPZ = aerobus::zpz<179>; using type = POLYV<ZPZV<1>, ZPZV<0>, ZPZV<7>, ZPZV<91>, ZPZV<55>, ZPZV<109>, ZPZV<2»; };  // NOLINT
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<177»; };  // NOLINT
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»; };  // NOLINT
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<40>, ZPZV<64>, ZPZV<177»; };  // NOLINT
template<> struct ConwayPolynomial<179, 10> { using ZPZ = aerobus::zpz<179>; using type = POLYV<ZPZV<1>, ZPZV<0>, ZPZV<0>, ZPZV<0>, ZPZV<0>, ZPZV<115>, ZPZV<71>, ZPZV<150>, ZPZV<49>, ZPZV<87>, ZPZV<2»; };  // NOLINT
template<> struct ConwayPolynomial<179, 11> { 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<28>, ZPZV<177»; };  // NOLINT
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<76>, ZPZV<8>, ZPZV<177>, ZPZV<1>, ZPZV<2»; };  // NOLINT
template<> struct ConwayPolynomial<179, 13> { 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<0>, ZPZV<0>, ZPZV<18>, ZPZV<177»; };  // NOLINT
template<> struct ConwayPolynomial<179, 17> { 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<0>, ZPZV<0>, ZPZV<0>, ZPZV<0>, ZPZV<0>, ZPZV<0>, ZPZV<4>, ZPZV<177»; };  // NOLINT
```

```
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<0>, ZPZV<0>, ZPZV<0>, ZPZV<0>, ZPZV<0>, ZPZV<0>, ZPZV<0>, ZPZV<0>, ZPZV<11>, ZPZV<177»; };  // NOLINT
template<> struct ConwayPolynomial<181, 1> { using ZPZ = aerobus::zpz<181>; using type = POLYV<ZPZV<1>, ZPZV<179»; };  // NOLINT
template<> struct ConwayPolynomial<181, 2> { using ZPZ = aerobus::zpz<181>; using type = POLYV<ZPZV<1>, ZPZV<177>, ZPZV<2»; };  // NOLINT
template<> struct ConwayPolynomial<181, 3> { using ZPZ = aerobus::zpz<181>; using type = POLYV<ZPZV<1>, ZPZV<0>, ZPZV<6>, ZPZV<179»; };  // NOLINT
template<> struct ConwayPolynomial<181, 4> { using ZPZ = aerobus::zpz<181>; using type = POLYV<ZPZV<1>, ZPZV<0>, ZPZV<6>, ZPZV<105>, ZPZV<2»; };  // NOLINT
template<> struct ConwayPolynomial<181, 5> { using ZPZ = aerobus::zpz<181>; using type = POLYV<ZPZV<1>, ZPZV<0>, ZPZV<0>, ZPZV<0>, ZPZV<21>, ZPZV<179»; };  // NOLINT
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
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<4>, ZPZV<179»; };  // NOLINT
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
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<11>, ZPZV<107>, ZPZV<168>, ZPZV<179»; };  // NOLINT
template<> struct ConwayPolynomial<181, 10> { using ZPZ = aerobus::zpz<181>; using type = POLYV<ZPZV<1>, ZPZV<0>, ZPZV<0>, ZPZV<0>, ZPZV<0>, ZPZV<154>, ZPZV<104>, ZPZV<94>, ZPZV<57>, ZPZV<88>, ZPZV<2»; };  // NOLINT
template<> struct ConwayPolynomial<181, 11> { 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<24>, ZPZV<179»; };  // NOLINT
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
template<> struct ConwayPolynomial<181, 13> { 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<0>, ZPZV<0>, ZPZV<8>, ZPZV<179»; };  // NOLINT
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<0>, ZPZV<0>, ZPZV<0>, ZPZV<0>, ZPZV<0>, ZPZV<0>, ZPZV<9>, ZPZV<179»; };  // NOLINT
template<> struct ConwayPolynomial<181, 19> { 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<0>, ZPZV<0>, ZPZV<0>, ZPZV<0>, ZPZV<0>, ZPZV<0>, ZPZV<0>, ZPZV<0>, ZPZV<36>, ZPZV<179»; };  // NOLINT
template<> struct ConwayPolynomial<191, 1> { using ZPZ = aerobus::zpz<191>; using type = POLYV<ZPZV<1>, ZPZV<172»; };  // NOLINT
template<> struct ConwayPolynomial<191, 2> { using ZPZ = aerobus::zpz<191>; using type = POLYV<ZPZV<1>, ZPZV<190>, ZPZV<19»; };  // NOLINT
template<> struct ConwayPolynomial<191, 3> { using ZPZ = aerobus::zpz<191>; using type = POLYV<ZPZV<1>, ZPZV<0>, ZPZV<4>, ZPZV<172»; };  // NOLINT
template<> struct ConwayPolynomial<191, 4> { using ZPZ = aerobus::zpz<191>; using type = POLYV<ZPZV<1>, ZPZV<0>, ZPZV<7>, ZPZV<100>, ZPZV<19»; };  // NOLINT
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
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<19»; };  // NOLINT
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»; };  // NOLINT
template<> struct ConwayPolynomial<191, 8> { using ZPZ = aerobus::zpz<191>; using type = POLYV<ZPZV<1>, ZPZV<0>, ZPZV<0>, ZPZV<0>, ZPZV<2>, ZPZV<164>, ZPZV<139>, ZPZV<171>, ZPZV<19»; };  // NOLINT
template<> struct ConwayPolynomial<191, 9> { using ZPZ = aerobus::zpz<191>; using type = POLYV<ZPZV<1>, ZPZV<0>, ZPZV<0>, ZPZV<0>, ZPZV<0>, ZPZV<0>, ZPZV<0>, ZPZV<62>, ZPZV<124>, ZPZV<172»; };  // NOLINT
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
template<> struct ConwayPolynomial<191, 11> { 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<6>, ZPZV<172»; };  // NOLINT
template<> struct ConwayPolynomial<191, 12> { using ZPZ = aerobus::zpz<191>; using type = POLYV<ZPZV<1>, ZPZV<0>, ZPZV<0>, ZPZV<0>, ZPZV<0>, ZPZV<79>, ZPZV<168>, ZPZV<25>, ZPZV<49>, ZPZV<90>, ZPZV<7>, ZPZV<151>, ZPZV<19»; };  // NOLINT
template<> struct ConwayPolynomial<191, 13> { 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<12>, ZPZV<172»; };  // NOLINT
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<0>, ZPZV<0>, ZPZV<0>, ZPZV<0>, ZPZV<2>, ZPZV<172»; };  // NOLINT
template<> struct ConwayPolynomial<191, 19> { 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<0>, ZPZV<0>, ZPZV<0>, ZPZV<0>, ZPZV<0>, ZPZV<190>, ZPZV<2>, ZPZV<172»; };  // NOLINT
template<> struct ConwayPolynomial<193, 1> { using ZPZ = aerobus::zpz<193>; using type = POLYV<ZPZV<1>, ZPZV<188»; };  // NOLINT
template<> struct ConwayPolynomial<193, 2> { using ZPZ = aerobus::zpz<193>; using type = POLYV<ZPZV<1>, ZPZV<192>, ZPZV<5»; };  // NOLINT
template<> struct ConwayPolynomial<193, 3> { using ZPZ = aerobus::zpz<193>; using type =
POLYV<ZPZV<1>, ZPZV<0>, ZPZV<1>, ZPZV<188»; };  // NOLINT
template<> struct ConwayPolynomial<193, 4> { using ZPZ = aerobus::zpz<193>; using type =
POLYV<ZPZV<1>, ZPZV<0>, ZPZV<6>, ZPZV<148>, ZPZV<5»; };  // NOLINT
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
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
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<8>, ZPZV<188»; };  // NOLINT
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<5»; };  //
NOLINT
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<27>, ZPZV<188»;
};  // NOLINT
template<> struct ConwayPolynomial<193, 10> { using ZPZ = aerobus::zpz<193>; using type =
POLYV<ZPZV<1>, ZPZV<0>, ZPZV<0>, ZPZV<0>, ZPZV<0>, ZPZV<20>, ZPZV<51>, ZPZV<77>, ZPZV<0>, ZPZV<89>,
ZPZV<5»; };  // NOLINT
template<> struct ConwayPolynomial<193, 11> { 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<0>, ZPZV<0>,
ZPZV<1>, ZPZV<188»; };  // NOLINT
template<> struct ConwayPolynomial<193, 12> { using ZPZ = aerobus::zpz<193>; using type =
POLYV<ZPZV<1>, ZPZV<0>, ZPZV<0>, ZPZV<0>, ZPZV<0>, ZPZV<155>, ZPZV<52>, ZPZV<135>, ZPZV<152>,
ZPZV<90>, ZPZV<46>, ZPZV<28>, ZPZV<5»; };  // NOLINT
template<> struct ConwayPolynomial<193, 13> { 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<0>, ZPZV<0>,
ZPZV<0>, ZPZV<0>, ZPZV<39>, ZPZV<188»; };  // NOLINT
template<> struct ConwayPolynomial<193, 17> { 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<0>, ZPZV<0>,
ZPZV<0>, ZPZV<0>, ZPZV<0>, ZPZV<0>, ZPZV<0>, ZPZV<9>, ZPZV<188»; };  // NOLINT
template<> struct ConwayPolynomial<193, 19> { 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<0>, ZPZV<0>,
ZPZV<0>, ZPZV<0>, ZPZV<0>, ZPZV<0>, ZPZV<0>, ZPZV<0>, ZPZV<0>, ZPZV<0>, ZPZV<5>, ZPZV<188»; };  //
NOLINT
template<> struct ConwayPolynomial<197, 1> { using ZPZ = aerobus::zpz<197>; using type =
POLYV<ZPZV<1>, ZPZV<195»; };  // NOLINT
template<> struct ConwayPolynomial<197, 2> { using ZPZ = aerobus::zpz<197>; using type =
POLYV<ZPZV<1>, ZPZV<192>, ZPZV<2»; };  // NOLINT
template<> struct ConwayPolynomial<197, 3> { using ZPZ = aerobus::zpz<197>; using type =
POLYV<ZPZV<1>, ZPZV<0>, ZPZV<3>, ZPZV<195»; };  // NOLINT
template<> struct ConwayPolynomial<197, 4> { using ZPZ = aerobus::zpz<197>; using type =
POLYV<ZPZV<1>, ZPZV<0>, ZPZV<16>, ZPZV<124>, ZPZV<2»; };  // NOLINT
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
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
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<6>, ZPZV<195»; };  // NOLINT
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<2»; };  //
NOLINT
template<> struct ConwayPolynomial<197, 9> { using ZPZ = aerobus::zpz<197>; using type =
POLYV<ZPZV<1>, ZPZV<0>, ZPZV<0>, ZPZV<0>, ZPZV<0>, ZPZV<13>, ZPZV<127>, ZPZV<8>, ZPZV<195»;
};  // NOLINT
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
template<> struct ConwayPolynomial<197, 11> { 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<0>, ZPZV<0>,
ZPZV<14>, ZPZV<195»; };  // NOLINT
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»; };  // NOLINT
template<> struct ConwayPolynomial<197, 13> { 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<0>, ZPZV<0>,
ZPZV<0>, ZPZV<0>, ZPZV<39>, ZPZV<195»; };  // NOLINT
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<0>, ZPZV<0>,
ZPZV<0>, ZPZV<0>, ZPZV<0>, ZPZV<0>, ZPZV<0>, ZPZV<0>, ZPZV<35>, ZPZV<195»; };  // NOLINT
template<> struct ConwayPolynomial<197, 19> { 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<0>, ZPZV<0>,
ZPZV<0>, ZPZV<0>, ZPZV<0>, ZPZV<0>, ZPZV<0>, ZPZV<0>, ZPZV<0>, ZPZV<0>, ZPZV<6>, ZPZV<195»; };  //
NOLINT
template<> struct ConwayPolynomial<199, 1> { using ZPZ = aerobus::zpz<199>; using type =
POLYV<ZPZV<1>, ZPZV<196»; };  // NOLINT
template<> struct ConwayPolynomial<199, 2> { using ZPZ = aerobus::zpz<199>; using type =
POLYV<ZPZV<1>, ZPZV<193>, ZPZV<3»; };  // NOLINT
template<> struct ConwayPolynomial<199, 3> { using ZPZ = aerobus::zpz<199>; using type =
POLYV<ZPZV<1>, ZPZV<0>, ZPZV<1>, ZPZV<196»; };  // NOLINT
template<> struct ConwayPolynomial<199, 4> { using ZPZ = aerobus::zpz<199>; using type =
POLYV<ZPZV<1>, ZPZV<0>, ZPZV<7>, ZPZV<162>, ZPZV<3»; };  // NOLINT
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
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
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
```

```
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>»; };  //
    NOLINT
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
template<> struct ConwayPolynomial<199, 10> { using ZPZ = aerobus::zpz<199>; using type =
    POLYV<ZPZV<1>, ZPZV<0>, ZPZV<0>, ZPZV<0>, ZPZV<0>, ZPZV<171>, ZPZV<158>, ZPZV<31>, ZPZV<54>, ZPZV<9>,
    ZPZV<3>»; };  // NOLINT
template<> struct ConwayPolynomial<199, 11> { using ZPZ = aerobus::zpz<199>; 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<196>»; };  // NOLINT
template<> struct ConwayPolynomial<199, 12> { using ZPZ = aerobus::zpz<199>; using type =
    POLYV<ZPZV<1>, ZPZV<0>, ZPZV<0>, ZPZV<0>, ZPZV<0>, ZPZV<33>, ZPZV<192>, ZPZV<197>, ZPZV<138>,
    ZPZV<69>, ZPZV<57>, ZPZV<151>, ZPZV<3>»; };  // NOLINT
template<> struct ConwayPolynomial<199, 13> { using ZPZ = aerobus::zpz<199>; 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<10>, ZPZV<196>»; };  // NOLINT
template<> struct ConwayPolynomial<199, 17> { using ZPZ = aerobus::zpz<199>; 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<13>, ZPZV<196>»; };  // NOLINT
template<> struct ConwayPolynomial<199, 19> { using ZPZ = aerobus::zpz<199>; 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<19>, ZPZV<196>»; };  //
    NOLINT
template<> struct ConwayPolynomial<211, 1> { using ZPZ = aerobus::zpz<211>; using type =
    POLYV<ZPZV<1>, ZPZV<209>»; };  // NOLINT
template<> struct ConwayPolynomial<211, 2> { using ZPZ = aerobus::zpz<211>; using type =
    POLYV<ZPZV<1>, ZPZV<207>, ZPZV<2>»; };  // NOLINT
template<> struct ConwayPolynomial<211, 3> { using ZPZ = aerobus::zpz<211>; using type =
    POLYV<ZPZV<1>, ZPZV<0>, ZPZV<2>, ZPZV<209>»; };  // NOLINT
template<> struct ConwayPolynomial<211, 4> { using ZPZ = aerobus::zpz<211>; using type =
    POLYV<ZPZV<1>, ZPZV<0>, ZPZV<8>, ZPZV<161>, ZPZV<2>»; };  // NOLINT
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
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
template<> struct ConwayPolynomial<211, 7> { using ZPZ = aerobus::zpz<211>; using type =
    POLYV<ZPZV<1>, ZPZV<0>, ZPZV<0>, ZPZV<0>, ZPZV<0>, ZPZV<0>, ZPZV<3>, ZPZV<209>»; };  // NOLINT
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<2>»; };  //
    NOLINT
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
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
template<> struct ConwayPolynomial<211, 11> { 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<7>, ZPZV<209>»; };  // NOLINT
template<> struct ConwayPolynomial<211, 12> { using ZPZ = aerobus::zpz<211>; using type =
    POLYV<ZPZV<1>, ZPZV<0>, ZPZV<0>, ZPZV<0>, ZPZV<0>, ZPZV<8>, ZPZV<50>, ZPZV<145>, ZPZV<126>, ZPZV<184>,
    ZPZV<84>, ZPZV<27>, ZPZV<2>»; };  // NOLINT
template<> struct ConwayPolynomial<211, 13> { 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<12>, ZPZV<209>»; };  // NOLINT
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<12>, ZPZV<209>»; };  // NOLINT
template<> struct ConwayPolynomial<211, 19> { 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<17>, ZPZV<209>»; };  //
    NOLINT
template<> struct ConwayPolynomial<223, 1> { using ZPZ = aerobus::zpz<223>; using type =
    POLYV<ZPZV<1>, ZPZV<220>»; };  // NOLINT
template<> struct ConwayPolynomial<223, 2> { using ZPZ = aerobus::zpz<223>; using type =
    POLYV<ZPZV<1>, ZPZV<221>, ZPZV<3>»; };  // NOLINT
template<> struct ConwayPolynomial<223, 3> { using ZPZ = aerobus::zpz<223>; using type =
    POLYV<ZPZV<1>, ZPZV<0>, ZPZV<6>, ZPZV<220>»; };  // NOLINT
template<> struct ConwayPolynomial<223, 4> { using ZPZ = aerobus::zpz<223>; using type =
    POLYV<ZPZV<1>, ZPZV<0>, ZPZV<6>, ZPZV<163>, ZPZV<3>»; };  // NOLINT
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
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
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>»; };  // NOLINT
template<> struct ConwayPolynomial<223, 8> { using ZPZ = aerobus::zpz<223>; using type =
    POLYV<ZPZV<1>, ZPZV<0>, ZPZV<0>, ZPZV<2>, ZPZV<139>, ZPZV<98>, ZPZV<138>, ZPZV<3>»; };  //
    NOLINT
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<1>, ZPZV<164>, ZPZV<64>, ZPZV<220>»;
    };  // NOLINT
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
```

```
template<> struct ConwayPolynomial<223, 11> { 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<8>, ZPZV<220»; };  // NOLINT
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
template<> struct ConwayPolynomial<223, 13> { 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<0>, ZPZV<23>, ZPZV<220»; };  // NOLINT
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<0>, ZPZV<0>, ZPZV<0>, ZPZV<0>, ZPZV<0>, ZPZV<1>, ZPZV<220»; };  // NOLINT
template<> struct ConwayPolynomial<223, 19> { 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<0>, ZPZV<0>, ZPZV<0>, ZPZV<0>, ZPZV<0>, ZPZV<0>, ZPZV<0>, ZPZV<7>, ZPZV<220»; };  // NOLINT
template<> struct ConwayPolynomial<227, 1> { using ZPZ = aerobus::zpz<227>; using type = POLYV<ZPZV<1>, ZPZV<225»; };  // NOLINT
template<> struct ConwayPolynomial<227, 2> { using ZPZ = aerobus::zpz<227>; using type = POLYV<ZPZV<1>, ZPZV<220>, ZPZV<2»; };  // NOLINT
template<> struct ConwayPolynomial<227, 3> { using ZPZ = aerobus::zpz<227>; using type = POLYV<ZPZV<1>, ZPZV<0>, ZPZV<2>, ZPZV<225»; };  // NOLINT
template<> struct ConwayPolynomial<227, 4> { using ZPZ = aerobus::zpz<227>; using type = POLYV<ZPZV<1>, ZPZV<0>, ZPZV<1>, ZPZV<143>, ZPZV<2»; };  // NOLINT
template<> struct ConwayPolynomial<227, 5> { using ZPZ = aerobus::zpz<227>; using type = POLYV<ZPZV<1>, ZPZV<0>, ZPZV<0>, ZPZV<0>, ZPZV<5>, ZPZV<225»; };  // NOLINT
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
template<> struct ConwayPolynomial<227, 7> { using ZPZ = aerobus::zpz<227>; using type = POLYV<ZPZV<1>, ZPZV<0>, ZPZV<0>, ZPZV<0>, ZPZV<0>, ZPZV<18>, ZPZV<225»; };  // NOLINT
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»; };  // NOLINT
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
template<> struct ConwayPolynomial<227, 10> { using ZPZ = aerobus::zpz<227>; using type = POLYV<ZPZV<1>, ZPZV<0>, ZPZV<0>, ZPZV<0>, ZPZV<0>, ZPZV<22>, ZPZV<199>, ZPZV<12>, ZPZV<93>, ZPZV<77>, ZPZV<2»; };  // NOLINT
template<> struct ConwayPolynomial<227, 11> { 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<0>, ZPZV<2>, ZPZV<225»; };  // NOLINT
template<> struct ConwayPolynomial<227, 12> { using ZPZ = aerobus::zpz<227>; using type = POLYV<ZPZV<1>, ZPZV<0>, ZPZV<0>, ZPZV<0>, ZPZV<123>, ZPZV<99>, ZPZV<160>, ZPZV<96>, ZPZV<127>, ZPZV<142>, ZPZV<94>, ZPZV<2»; };  // NOLINT
template<> struct ConwayPolynomial<227, 13> { 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<0>, ZPZV<0>, ZPZV<0>, ZPZV<2>, ZPZV<225»; };  // NOLINT
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<0>, ZPZV<0>, ZPZV<0>, ZPZV<0>, ZPZV<0>, ZPZV<0>, ZPZV<0>, ZPZV<8>, ZPZV<225»; };  // NOLINT
template<> struct ConwayPolynomial<227, 19> { 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<0>, ZPZV<0>, ZPZV<0>, ZPZV<0>, ZPZV<0>, ZPZV<0>, ZPZV<0>, ZPZV<0>, ZPZV<0>, ZPZV<34>, ZPZV<225»; };  // NOLINT
template<> struct ConwayPolynomial<229, 1> { using ZPZ = aerobus::zpz<229>; using type = POLYV<ZPZV<1>, ZPZV<223»; };  // NOLINT
template<> struct ConwayPolynomial<229, 2> { using ZPZ = aerobus::zpz<229>; using type = POLYV<ZPZV<1>, ZPZV<228>, ZPZV<6»; };  // NOLINT
template<> struct ConwayPolynomial<229, 3> { using ZPZ = aerobus::zpz<229>; using type = POLYV<ZPZV<1>, ZPZV<0>, ZPZV<1>, ZPZV<223»; };  // NOLINT
template<> struct ConwayPolynomial<229, 4> { using ZPZ = aerobus::zpz<229>; using type = POLYV<ZPZV<1>, ZPZV<0>, ZPZV<7>, ZPZV<162>, ZPZV<6»; };  // NOLINT
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
template<> struct ConwayPolynomial<229, 6> { using ZPZ = aerobus::zpz<229>; using type = POLYV<ZPZV<1>, ZPZV<0>, ZPZV<0>, ZPZV<24>, ZPZV<160>, ZPZV<186>, ZPZV<6»; };  // NOLINT
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<223»; };  // NOLINT
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
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<15>, ZPZV<117>, ZPZV<50>, ZPZV<223»; };  // NOLINT
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
template<> struct ConwayPolynomial<229, 11> { 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<0>, ZPZV<2>, ZPZV<223»; };  // NOLINT
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
template<> struct ConwayPolynomial<229, 13> { 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<0>, ZPZV<0>, ZPZV<0>, ZPZV<47>, ZPZV<223»; };  // NOLINT
```

```
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<0>, ZPZV<0>, ZPZV<0>, ZPZV<0>, ZPZV<0>, ZPZV<0>, ZPZV<2>, ZPZV<223»; };  // NOLINT
template<> struct ConwayPolynomial<229, 19> { 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<0>, ZPZV<0>, ZPZV<0>, ZPZV<0>, ZPZV<0>, ZPZV<0>, ZPZV<0>, ZPZV<0>, ZPZV<228>, ZPZV<15>, ZPZV<223»; };  // NOLINT
template<> struct ConwayPolynomial<233, 1> { using ZPZ = aerobus::zpz<233>; using type = POLYV<ZPZV<1>, ZPZV<230»; };  // NOLINT
template<> struct ConwayPolynomial<233, 2> { using ZPZ = aerobus::zpz<233>; using type = POLYV<ZPZV<1>, ZPZV<232>, ZPZV<3»; };  // NOLINT
template<> struct ConwayPolynomial<233, 3> { using ZPZ = aerobus::zpz<233>; using type = POLYV<ZPZV<1>, ZPZV<0>, ZPZV<1>, ZPZV<230»; };  // NOLINT
template<> struct ConwayPolynomial<233, 4> { using ZPZ = aerobus::zpz<233>; using type = POLYV<ZPZV<1>, ZPZV<0>, ZPZV<4>, ZPZV<158>, ZPZV<3»; };  // NOLINT
template<> struct ConwayPolynomial<233, 5> { using ZPZ = aerobus::zpz<233>; using type = POLYV<ZPZV<1>, ZPZV<0>, ZPZV<0>, ZPZV<0>, ZPZV<17>, ZPZV<230»; };  // NOLINT
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
template<> struct ConwayPolynomial<233, 7> { using ZPZ = aerobus::zpz<233>; using type = POLYV<ZPZV<1>, ZPZV<0>, ZPZV<0>, ZPZV<0>, ZPZV<0>, ZPZV<0>, ZPZV<4>, ZPZV<230»; };  // NOLINT
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
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<7>, ZPZV<56>, ZPZV<146>, ZPZV<230»; };  // NOLINT
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
template<> struct ConwayPolynomial<233, 11> { 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<5>, ZPZV<230»; };  // NOLINT
template<> struct ConwayPolynomial<233, 12> { using ZPZ = aerobus::zpz<233>; using type = POLYV<ZPZV<1>, ZPZV<0>, ZPZV<0>, ZPZV<0>, ZPZV<0>, ZPZV<96>, ZPZV<21>, ZPZV<114>, ZPZV<31>, ZPZV<19>, ZPZV<216>, ZPZV<20>, ZPZV<3»; };  // NOLINT
template<> struct ConwayPolynomial<233, 13> { 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<14>, ZPZV<230»; };  // NOLINT
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<0>, ZPZV<0>, ZPZV<0>, ZPZV<0>, ZPZV<0>, ZPZV<0>, ZPZV<0>, ZPZV<4>, ZPZV<230»; };  // NOLINT
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<0>, ZPZV<0>, ZPZV<0>, ZPZV<0>, ZPZV<0>, ZPZV<0>, ZPZV<25>, ZPZV<230»; };  // NOLINT
template<> struct ConwayPolynomial<239, 1> { using ZPZ = aerobus::zpz<239>; using type = POLYV<ZPZV<1>, ZPZV<232»; };  // NOLINT
template<> struct ConwayPolynomial<239, 2> { using ZPZ = aerobus::zpz<239>; using type = POLYV<ZPZV<1>, ZPZV<237>, ZPZV<7»; };  // NOLINT
template<> struct ConwayPolynomial<239, 3> { using ZPZ = aerobus::zpz<239>; using type = POLYV<ZPZV<1>, ZPZV<0>, ZPZV<11>, ZPZV<232»; };  // NOLINT
template<> struct ConwayPolynomial<239, 4> { using ZPZ = aerobus::zpz<239>; using type = POLYV<ZPZV<1>, ZPZV<0>, ZPZV<11>, ZPZV<132>, ZPZV<7»; };  // NOLINT
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
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
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<17>, ZPZV<232»; };  // NOLINT
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<54>, ZPZV<7»; };  // NOLINT
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<2>, ZPZV<88>, ZPZV<232»; };  // NOLINT
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
template<> struct ConwayPolynomial<239, 11> { 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<8>, ZPZV<232»; };  // NOLINT
template<> struct ConwayPolynomial<239, 12> { using ZPZ = aerobus::zpz<239>; using type = POLYV<ZPZV<1>, ZPZV<0>, ZPZV<0>, ZPZV<0>, ZPZV<1>, ZPZV<235>, ZPZV<14>, ZPZV<113>, ZPZV<182>, ZPZV<101>, ZPZV<81>, ZPZV<216>, ZPZV<7»; };  // NOLINT
template<> struct ConwayPolynomial<239, 13> { 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<2>, ZPZV<232»; };  // NOLINT
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<0>, ZPZV<0>, ZPZV<0>, ZPZV<0>, ZPZV<0>, ZPZV<0>, ZPZV<9>, ZPZV<232»; };  // NOLINT
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<0>, ZPZV<0>, ZPZV<0>, ZPZV<0>, ZPZV<0>, ZPZV<0>, ZPZV<24>, ZPZV<232»; };  // NOLINT
template<> struct ConwayPolynomial<241, 1> { using ZPZ = aerobus::zpz<241>; using type = POLYV<ZPZV<1>, ZPZV<234»; };  // NOLINT
```

```
template<> struct ConwayPolynomial<241, 2> { using ZPZ = aerobus::zpz<241>; using type = POLYV<ZPZV<1>, ZPZV<238>, ZPZV<7»; };  // NOLINT
template<> struct ConwayPolynomial<241, 3> { using ZPZ = aerobus::zpz<241>; using type = POLYV<ZPZV<1>, ZPZV<0>, ZPZV<1>, ZPZV<234»; };  // NOLINT
template<> struct ConwayPolynomial<241, 4> { using ZPZ = aerobus::zpz<241>; using type = POLYV<ZPZV<1>, ZPZV<0>, ZPZV<14>, ZPZV<152>, ZPZV<7»; };  // NOLINT
template<> struct ConwayPolynomial<241, 5> { using ZPZ = aerobus::zpz<241>; using type = POLYV<ZPZV<1>, ZPZV<0>, ZPZV<0>, ZPZV<0>, ZPZV<7>, ZPZV<234»; };  // NOLINT
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
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<2>, ZPZV<234»; };  // NOLINT
template<> struct ConwayPolynomial<241, 8> { using ZPZ = aerobus::zpz<241>; using type = POLYV<ZPZV<1>, ZPZV<0>, ZPZV<0>, ZPZV<0>, ZPZV<1>, ZPZV<173>, ZPZV<212>, ZPZV<153>, ZPZV<7»; };  // NOLINT
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
template<> struct ConwayPolynomial<241, 10> { using ZPZ = aerobus::zpz<241>; using type = POLYV<ZPZV<1>, ZPZV<0>, ZPZV<0>, ZPZV<0>, ZPZV<1>, ZPZV<29>, ZPZV<27>, ZPZV<145>, ZPZV<208>, ZPZV<55>, ZPZV<7»; };  // NOLINT
template<> struct ConwayPolynomial<241, 11> { 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<3>, ZPZV<234»; };  // NOLINT
template<> struct ConwayPolynomial<241, 12> { using ZPZ = aerobus::zpz<241>; using type = POLYV<ZPZV<1>, ZPZV<0>, ZPZV<0>, ZPZV<0>, ZPZV<0>, ZPZV<42>, ZPZV<10>, ZPZV<109>, ZPZV<168>, ZPZV<22>, ZPZV<197>, ZPZV<17>, ZPZV<7»; };  // NOLINT
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<0>, ZPZV<11>, ZPZV<234»; };  // NOLINT
template<> struct ConwayPolynomial<241, 17> { 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<0>, ZPZV<0>, ZPZV<0>, ZPZV<0>, ZPZV<0>, ZPZV<14>, ZPZV<234»; };  // NOLINT
template<> struct ConwayPolynomial<241, 19> { 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<0>, ZPZV<0>, ZPZV<0>, ZPZV<0>, ZPZV<0>, ZPZV<0>, ZPZV<0>, ZPZV<11>, ZPZV<234»; };  // NOLINT
template<> struct ConwayPolynomial<251, 1> { using ZPZ = aerobus::zpz<251>; using type = POLYV<ZPZV<1>, ZPZV<245»; };  // NOLINT
template<> struct ConwayPolynomial<251, 2> { using ZPZ = aerobus::zpz<251>; using type = POLYV<ZPZV<1>, ZPZV<242>, ZPZV<6»; };  // NOLINT
template<> struct ConwayPolynomial<251, 3> { using ZPZ = aerobus::zpz<251>; using type = POLYV<ZPZV<1>, ZPZV<0>, ZPZV<3>, ZPZV<245»; };  // NOLINT
template<> struct ConwayPolynomial<251, 4> { using ZPZ = aerobus::zpz<251>; using type = POLYV<ZPZV<1>, ZPZV<0>, ZPZV<3>, ZPZV<200>, ZPZV<6»; };  // NOLINT
template<> struct ConwayPolynomial<251, 5> { using ZPZ = aerobus::zpz<251>; using type = POLYV<ZPZV<1>, ZPZV<0>, ZPZV<0>, ZPZV<0>, ZPZV<3>, ZPZV<245»; };  // NOLINT
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
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
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
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<4>, ZPZV<187>, ZPZV<106>, ZPZV<245»; };  // NOLINT
template<> struct ConwayPolynomial<251, 10> { using ZPZ = aerobus::zpz<251>; using type = POLYV<ZPZV<1>, ZPZV<0>, ZPZV<0>, ZPZV<0>, ZPZV<1>, ZPZV<138>, ZPZV<110>, ZPZV<45>, ZPZV<34>, ZPZV<149>, ZPZV<6»; };  // NOLINT
template<> struct ConwayPolynomial<251, 11> { 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<26>, ZPZV<245»; };  // NOLINT
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
template<> struct ConwayPolynomial<251, 13> { 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<0>, ZPZV<15>, ZPZV<245»; };  // NOLINT
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<0>, ZPZV<0>, ZPZV<0>, ZPZV<0>, ZPZV<0>, ZPZV<9>, ZPZV<245»; };  // NOLINT
template<> struct ConwayPolynomial<251, 19> { 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<0>, ZPZV<0>, ZPZV<0>, ZPZV<0>, ZPZV<0>, ZPZV<0>, ZPZV<0>, ZPZV<8>, ZPZV<245»; };  // NOLINT
template<> struct ConwayPolynomial<257, 1> { using ZPZ = aerobus::zpz<257>; using type = POLYV<ZPZV<1>, ZPZV<254»; };  // NOLINT
template<> struct ConwayPolynomial<257, 2> { using ZPZ = aerobus::zpz<257>; using type = POLYV<ZPZV<1>, ZPZV<251>, ZPZV<3»; };  // NOLINT
template<> struct ConwayPolynomial<257, 3> { using ZPZ = aerobus::zpz<257>; using type = POLYV<ZPZV<1>, ZPZV<0>, ZPZV<6>, ZPZV<254»; };  // NOLINT
template<> struct ConwayPolynomial<257, 4> { using ZPZ = aerobus::zpz<257>; using type = POLYV<ZPZV<1>, ZPZV<0>, ZPZV<16>, ZPZV<187>, ZPZV<3»; };  // NOLINT
template<> struct ConwayPolynomial<257, 5> { using ZPZ = aerobus::zpz<257>; using type = POLYV<ZPZV<1>, ZPZV<0>, ZPZV<0>, ZPZV<0>, ZPZV<4>, ZPZV<254»; };  // NOLINT
template<> struct ConwayPolynomial<257, 6> { using ZPZ = aerobus::zpz<257>; using type =
POLYV<ZPZV<1>, ZPZV<0>, ZPZV<3>, ZPZV<62>, ZPZV<18>, ZPZV<138>, ZPZV<3»; };  // NOLINT
template<> struct ConwayPolynomial<257, 7> { using ZPZ = aerobus::zpz<257>; using type =
POLYV<ZPZV<1>, ZPZV<0>, ZPZV<0>, ZPZV<0>, ZPZV<0>, ZPZV<31>, ZPZV<254»; };  // NOLINT
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
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
template<> struct ConwayPolynomial<257, 10> { using ZPZ = aerobus::zpz<257>; using type =
POLYV<ZPZV<1>, ZPZV<0>, ZPZV<0>, ZPZV<0>, ZPZV<0>, ZPZV<97>, ZPZV<12>, ZPZV<225>, ZPZV<180>, ZPZV<20>,
ZPZV<3»; };  // NOLINT
template<> struct ConwayPolynomial<257, 11> { using ZPZ = aerobus::zpz<257>; using type =
POLYV<ZPZV<1>, ZPZV<0>, ZPZV<0>, ZPZV<0>, ZPZV<0>, ZPZV<0>, ZPZV<0>, ZPZV<0>, ZPZV<0>, ZPZV<0>,
ZPZV<40>, ZPZV<254»; };  // NOLINT
template<> struct ConwayPolynomial<257, 12> { using ZPZ = aerobus::zpz<257>; using type =
POLYV<ZPZV<1>, ZPZV<0>, ZPZV<0>, ZPZV<0>, ZPZV<2>, ZPZV<13>, ZPZV<225>, ZPZV<215>, ZPZV<173>,
ZPZV<249>, ZPZV<148>, ZPZV<20>, ZPZV<3»; };  // NOLINT
template<> struct ConwayPolynomial<257, 13> { using ZPZ = aerobus::zpz<257>; 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<5>, ZPZV<254»; };  // NOLINT
template<> struct ConwayPolynomial<257, 17> { using ZPZ = aerobus::zpz<257>; 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<8>, ZPZV<254»; };  // NOLINT
template<> struct ConwayPolynomial<257, 19> { using ZPZ = aerobus::zpz<257>; 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<10>, ZPZV<254»; };  //
NOLINT
template<> struct ConwayPolynomial<263, 1> { using ZPZ = aerobus::zpz<263>; using type =
POLYV<ZPZV<1>, ZPZV<258»; };  // NOLINT
template<> struct ConwayPolynomial<263, 2> { using ZPZ = aerobus::zpz<263>; using type =
POLYV<ZPZV<1>, ZPZV<261>, ZPZV<5»; };  // NOLINT
template<> struct ConwayPolynomial<263, 3> { using ZPZ = aerobus::zpz<263>; using type =
POLYV<ZPZV<1>, ZPZV<0>, ZPZV<14>, ZPZV<258»; };  // NOLINT
template<> struct ConwayPolynomial<263, 4> { using ZPZ = aerobus::zpz<263>; using type =
POLYV<ZPZV<1>, ZPZV<0>, ZPZV<5>, ZPZV<171>, ZPZV<5»; };  // NOLINT
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
template<> struct ConwayPolynomial<263, 6> { using ZPZ = aerobus::zpz<263>; using type =
POLYV<ZPZV<1>, ZPZV<0>, ZPZV<1>, ZPZV<222>, ZPZV<250>, ZPZV<225>, ZPZV<5»; };  // NOLINT
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<258»; };  // NOLINT
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
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<261>, ZPZV<29>, ZPZV<258»;
};  // NOLINT
template<> struct ConwayPolynomial<263, 10> { using ZPZ = aerobus::zpz<263>; using type =
POLYV<ZPZV<1>, ZPZV<0>, ZPZV<0>, ZPZV<0>, ZPZV<0>, ZPZV<245>, ZPZV<231>, ZPZV<198>, ZPZV<145>,
ZPZV<119>, ZPZV<5»; };  // NOLINT
template<> struct ConwayPolynomial<263, 11> { using ZPZ = aerobus::zpz<263>; 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<258»; };  // NOLINT
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
template<> struct ConwayPolynomial<269, 1> { using ZPZ = aerobus::zpz<269>; using type =
POLYV<ZPZV<1>, ZPZV<267»; };  // NOLINT
template<> struct ConwayPolynomial<269, 2> { using ZPZ = aerobus::zpz<269>; using type =
POLYV<ZPZV<1>, ZPZV<268>, ZPZV<2»; };  // NOLINT
template<> struct ConwayPolynomial<269, 3> { using ZPZ = aerobus::zpz<269>; using type =
POLYV<ZPZV<1>, ZPZV<0>, ZPZV<9>, ZPZV<267»; };  // NOLINT
template<> struct ConwayPolynomial<269, 4> { using ZPZ = aerobus::zpz<269>; using type =
POLYV<ZPZV<1>, ZPZV<0>, ZPZV<8>, ZPZV<262>, ZPZV<2»; };  // NOLINT
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
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
template<> struct ConwayPolynomial<269, 7> { using ZPZ = aerobus::zpz<269>; using type =
POLYV<ZPZV<1>, ZPZV<0>, ZPZV<0>, ZPZV<0>, ZPZV<0>, ZPZV<6>, ZPZV<267»; };  // NOLINT
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
template<> struct ConwayPolynomial<269, 9> { using ZPZ = aerobus::zpz<269>; using type =
POLYV<ZPZV<1>, ZPZV<0>, ZPZV<0>, ZPZV<0>, ZPZV<0>, ZPZV<2>, ZPZV<214>, ZPZV<267>, ZPZV<267»;
};  // NOLINT
template<> struct ConwayPolynomial<269, 10> { using ZPZ = aerobus::zpz<269>; using type =
POLYV<ZPZV<1>, ZPZV<0>, ZPZV<0>, ZPZV<2>, ZPZV<264>, ZPZV<243>, ZPZV<186>, ZPZV<61>,
ZPZV<10>, ZPZV<2»; };  // NOLINT
template<> struct ConwayPolynomial<269, 11> { using ZPZ = aerobus::zpz<269>; using type =
POLYV<ZPZV<1>, ZPZV<0>, ZPZV<0>, ZPZV<0>, ZPZV<0>, ZPZV<0>, ZPZV<0>, ZPZV<0>, ZPZV<0>, ZPZV<0>,
ZPZV<20>, ZPZV<267»; };  // NOLINT
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<63>, ZPZV<215>,
ZPZV<132>, ZPZV<180>, ZPZV<150>, ZPZV<2»; };  // NOLINT
template<> struct ConwayPolynomial<271, 1> { using ZPZ = aerobus::zpz<271>; using type =
POLYV<ZPZV<1>, ZPZV<265»; };  // NOLINT
template<> struct ConwayPolynomial<271, 2> { using ZPZ = aerobus::zpz<271>; using type =
POLYV<ZPZV<1>, ZPZV<269>, ZPZV<6»; };  // NOLINT
template<> struct ConwayPolynomial<271, 3> { using ZPZ = aerobus::zpz<271>; using type =
POLYV<ZPZV<1>, ZPZV<0>, ZPZV<2>, ZPZV<265»; };  // NOLINT
template<> struct ConwayPolynomial<271, 4> { using ZPZ = aerobus::zpz<271>; using type =
POLYV<ZPZV<1>, ZPZV<0>, ZPZV<3>, ZPZV<205>, ZPZV<6»; };  // NOLINT
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
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
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<22>, ZPZV<265»; };  // NOLINT
template<> struct ConwayPolynomial<271, 8> { using ZPZ = aerobus::zpz<271>; using type =
POLYV<ZPZV<1>, ZPZV<0>, ZPZV<0>, ZPZV<0>, ZPZV<1>, ZPZV<199>, ZPZV<114>, ZPZV<69>, ZPZV<6»; };  //
NOLINT
template<> struct ConwayPolynomial<271, 9> { using ZPZ = aerobus::zpz<271>; using type =
POLYV<ZPZV<1>, ZPZV<0>, ZPZV<0>, ZPZV<0>, ZPZV<0>, ZPZV<0>, ZPZV<10>, ZPZV<266>, ZPZV<186>, ZPZV<265»;
};  // NOLINT
template<> struct ConwayPolynomial<271, 10> { using ZPZ = aerobus::zpz<271>; using type =
POLYV<ZPZV<1>, ZPZV<0>, ZPZV<0>, ZPZV<1>, ZPZV<133>, ZPZV<10>, ZPZV<256>, ZPZV<74>,
ZPZV<126>, ZPZV<6»; };  // NOLINT
template<> struct ConwayPolynomial<271, 11> { using ZPZ = aerobus::zpz<271>; using type =
POLYV<ZPZV<1>, ZPZV<0>, ZPZV<0>, ZPZV<0>, ZPZV<0>, ZPZV<0>, ZPZV<0>, ZPZV<0>, ZPZV<0>, ZPZV<0>,
ZPZV<10>, ZPZV<265»; };  // NOLINT
template<> struct ConwayPolynomial<271, 12> { using ZPZ = aerobus::zpz<271>; using type =
POLYV<ZPZV<1>, 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
template<> struct ConwayPolynomial<277, 1> { using ZPZ = aerobus::zpz<277>; using type =
POLYV<ZPZV<1>, ZPZV<272»; };  // NOLINT
template<> struct ConwayPolynomial<277, 2> { using ZPZ = aerobus::zpz<277>; using type =
POLYV<ZPZV<1>, ZPZV<274>, ZPZV<5»; };  // NOLINT
template<> struct ConwayPolynomial<277, 3> { using ZPZ = aerobus::zpz<277>; using type =
POLYV<ZPZV<1>, ZPZV<0>, ZPZV<3>, ZPZV<272»; };  // NOLINT
template<> struct ConwayPolynomial<277, 4> { using ZPZ = aerobus::zpz<277>; using type =
POLYV<ZPZV<1>, ZPZV<0>, ZPZV<1>, ZPZV<222>, ZPZV<5»; };  // NOLINT
template<> struct ConwayPolynomial<277, 5> { using ZPZ = aerobus::zpz<277>; using type =
POLYV<ZPZV<1>, ZPZV<0>, ZPZV<0>, ZPZV<0>, ZPZV<1>, ZPZV<272»; };  // NOLINT
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
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<9>, ZPZV<272»; };  // NOLINT
template<> struct ConwayPolynomial<277, 8> { using ZPZ = aerobus::zpz<277>; using type =
POLYV<ZPZV<1>, ZPZV<0>, ZPZV<0>, ZPZV<0>, ZPZV<4>, ZPZV<187>, ZPZV<159>, ZPZV<176>, ZPZV<5»; };  //
NOLINT
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
template<> struct ConwayPolynomial<277, 10> { using ZPZ = aerobus::zpz<277>; using type =
POLYV<ZPZV<1>, ZPZV<0>, ZPZV<0>, ZPZV<0>, ZPZV<0>, ZPZV<206>, ZPZV<253>, ZPZV<237>, ZPZV<241>,
ZPZV<260>, ZPZV<5»; };  // NOLINT
template<> struct ConwayPolynomial<277, 11> { using ZPZ = aerobus::zpz<277>; using type =
POLYV<ZPZV<1>, ZPZV<0>, ZPZV<0>, ZPZV<0>, ZPZV<0>, ZPZV<0>, ZPZV<0>, ZPZV<0>, ZPZV<0>, ZPZV<0>,
ZPZV<5>, ZPZV<272»; };  // NOLINT
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<180>, ZPZV<115>, ZPZV<202>, ZPZV<5»; };  // NOLINT
template<> struct ConwayPolynomial<281, 1> { using ZPZ = aerobus::zpz<281>; using type =
POLYV<ZPZV<1>, ZPZV<278»; };  // NOLINT
template<> struct ConwayPolynomial<281, 2> { using ZPZ = aerobus::zpz<281>; using type =
POLYV<ZPZV<1>, ZPZV<280>, ZPZV<3»; };  // NOLINT
template<> struct ConwayPolynomial<281, 3> { using ZPZ = aerobus::zpz<281>; using type =
POLYV<ZPZV<1>, ZPZV<0>, ZPZV<1>, ZPZV<278»; };  // NOLINT
template<> struct ConwayPolynomial<281, 4> { using ZPZ = aerobus::zpz<281>; using type =
POLYV<ZPZV<1>, ZPZV<0>, ZPZV<7>, ZPZV<176>, ZPZV<3»; };  // NOLINT
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
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
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<19>, ZPZV<278»; };  // NOLINT
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
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<6>, ZPZV<148>, ZPZV<70>, ZPZV<278»;
};  // NOLINT
template<> struct ConwayPolynomial<281, 10> { using ZPZ = aerobus::zpz<281>; using type =
POLYV<ZPZV<1>, ZPZV<0>, ZPZV<0>, ZPZV<2>, ZPZV<258>, ZPZV<145>, ZPZV<13>, ZPZV<138>,
ZPZV<191>, ZPZV<3»; };  // NOLINT
template<> struct ConwayPolynomial<281, 11> { using ZPZ = aerobus::zpz<281>; using type =
POLYV<ZPZV<1>, ZPZV<0>, ZPZV<0>, ZPZV<0>, ZPZV<0>, ZPZV<0>, ZPZV<0>, ZPZV<0>, ZPZV<0>, ZPZV<0>,
ZPZV<36>, ZPZV<278»; };  // NOLINT
template<> struct ConwayPolynomial<281, 12> { using ZPZ = aerobus::zpz<281>; using type =
POLYV<ZPZV<1>, ZPZV<0>, ZPZV<0>, ZPZV<0>, ZPZV<0>, ZPZV<202>, ZPZV<68>, ZPZV<103>, ZPZV<116>,
ZPZV<58>, ZPZV<28>, ZPZV<191>, ZPZV<3»; };  // NOLINT
template<> struct ConwayPolynomial<283, 1> { using ZPZ = aerobus::zpz<283>; using type =
POLYV<ZPZV<1>, ZPZV<280»; };  // NOLINT
template<> struct ConwayPolynomial<283, 2> { using ZPZ = aerobus::zpz<283>; using type = POLYV<ZPZV<1>, ZPZV<282>, ZPZV<3»; };  // NOLINT
template<> struct ConwayPolynomial<283, 3> { using ZPZ = aerobus::zpz<283>; using type = POLYV<ZPZV<1>, ZPZV<0>, ZPZV<3>, ZPZV<280»; };  // NOLINT
template<> struct ConwayPolynomial<283, 4> { using ZPZ = aerobus::zpz<283>; using type = POLYV<ZPZV<1>, ZPZV<0>, ZPZV<5>, ZPZV<238>, ZPZV<3»; };  // NOLINT
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
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
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<8>, ZPZV<280»; };  // NOLINT
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<3»; };  // NOLINT
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
template<> struct ConwayPolynomial<283, 10> { using ZPZ = aerobus::zpz<283>; using type = POLYV<ZPZV<1>, ZPZV<0>, ZPZV<0>, ZPZV<3>, ZPZV<271>, ZPZV<185>, ZPZV<68>, ZPZV<100>, ZPZV<219>, ZPZV<3»; };  // NOLINT
template<> struct ConwayPolynomial<283, 11> { using ZPZ = aerobus::zpz<283>; 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<280»; };  // NOLINT
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<8>, ZPZV<96>, ZPZV<229>, ZPZV<49>, ZPZV<14>, ZPZV<56>, ZPZV<3»; };  // NOLINT
template<> struct ConwayPolynomial<293, 1> { using ZPZ = aerobus::zpz<293>; using type = POLYV<ZPZV<1>, ZPZV<291»; };  // NOLINT
template<> struct ConwayPolynomial<293, 2> { using ZPZ = aerobus::zpz<293>; using type = POLYV<ZPZV<1>, ZPZV<292>, ZPZV<2»; };  // NOLINT
template<> struct ConwayPolynomial<293, 3> { using ZPZ = aerobus::zpz<293>; using type = POLYV<ZPZV<1>, ZPZV<0>, ZPZV<2>, ZPZV<291»; };  // NOLINT
template<> struct ConwayPolynomial<293, 4> { using ZPZ = aerobus::zpz<293>; using type = POLYV<ZPZV<1>, ZPZV<0>, ZPZV<3>, ZPZV<166>, ZPZV<2»; };  // NOLINT
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
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
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<8>, ZPZV<291»; };  // NOLINT
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<2»; };  // NOLINT
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<208>, ZPZV<190>, ZPZV<291»; };  // NOLINT
template<> struct ConwayPolynomial<293, 10> { using ZPZ = aerobus::zpz<293>; using type = POLYV<ZPZV<1>, ZPZV<0>, ZPZV<0>, ZPZV<0>, ZPZV<0>, ZPZV<186>, ZPZV<28>, ZPZV<46>, ZPZV<184>, ZPZV<24>, ZPZV<2»; };  // NOLINT
template<> struct ConwayPolynomial<293, 11> { 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<0>, ZPZV<0>, ZPZV<3>, ZPZV<291»; };  // NOLINT
template<> struct ConwayPolynomial<293, 12> { using ZPZ = aerobus::zpz<293>; using type = POLYV<ZPZV<1>, ZPZV<0>, ZPZV<0>, ZPZV<0>, ZPZV<0>, ZPZV<159>, ZPZV<210>, ZPZV<125>, ZPZV<212>, ZPZV<167>, ZPZV<144>, ZPZV<157>, ZPZV<2»; };  // NOLINT
template<> struct ConwayPolynomial<307, 1> { using ZPZ = aerobus::zpz<307>; using type = POLYV<ZPZV<1>, ZPZV<302»; };  // NOLINT
template<> struct ConwayPolynomial<307, 2> { using ZPZ = aerobus::zpz<307>; using type = POLYV<ZPZV<1>, ZPZV<306>, ZPZV<5»; };  // NOLINT
template<> struct ConwayPolynomial<307, 3> { using ZPZ = aerobus::zpz<307>; using type = POLYV<ZPZV<1>, ZPZV<0>, ZPZV<7>, ZPZV<302»; };  // NOLINT
template<> struct ConwayPolynomial<307, 4> { using ZPZ = aerobus::zpz<307>; using type = POLYV<ZPZV<1>, ZPZV<0>, ZPZV<2>, ZPZV<239>, ZPZV<5»; };  // NOLINT
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
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
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<302»; };  // NOLINT
template<> struct ConwayPolynomial<307, 8> { using ZPZ = aerobus::zpz<307>; using type = POLYV<ZPZV<1>, ZPZV<0>, ZPZV<0>, ZPZV<0>, ZPZV<0>, ZPZV<283>, ZPZV<232>, ZPZV<131>, ZPZV<5»; };  // NOLINT
template<> struct ConwayPolynomial<307, 9> { using ZPZ = aerobus::zpz<307>; using type = POLYV<ZPZV<1>, ZPZV<0>, ZPZV<0>, ZPZV<0>, ZPZV<0>, ZPZV<1>, ZPZV<165>, ZPZV<70>, ZPZV<302»; };  // NOLINT
template<> struct ConwayPolynomial<311, 1> { using ZPZ = aerobus::zpz<311>; using type = POLYV<ZPZV<1>, ZPZV<294»; };  // NOLINT
template<> struct ConwayPolynomial<311, 2> { using ZPZ = aerobus::zpz<311>; using type = POLYV<ZPZV<1>, ZPZV<310>, ZPZV<17»; };  // NOLINT
template<> struct ConwayPolynomial<311, 3> { using ZPZ = aerobus::zpz<311>; using type = POLYV<ZPZV<1>, ZPZV<0>, ZPZV<3>, ZPZV<294»; };  // NOLINT
template<> struct ConwayPolynomial<311, 4> { using ZPZ = aerobus::zpz<311>; using type = POLYV<ZPZV<1>, ZPZV<0>, ZPZV<3>, ZPZV<163>, ZPZV<17»; };  // NOLINT
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
```

```
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
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<10>, ZPZV<294»; };  // NOLINT
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
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<1>, ZPZV<287>, ZPZV<74>, ZPZV<294»; };  // NOLINT
template<> struct ConwayPolynomial<313, 1> { using ZPZ = aerobus::zpz<313>; using type = POLYV<ZPZV<1>, ZPZV<303»; };  // NOLINT
template<> struct ConwayPolynomial<313, 2> { using ZPZ = aerobus::zpz<313>; using type = POLYV<ZPZV<1>, ZPZV<310>, ZPZV<10»; };  // NOLINT
template<> struct ConwayPolynomial<313, 3> { using ZPZ = aerobus::zpz<313>; using type = POLYV<ZPZV<1>, ZPZV<0>, ZPZV<2>, ZPZV<303»; };  // NOLINT
template<> struct ConwayPolynomial<313, 4> { using ZPZ = aerobus::zpz<313>; using type = POLYV<ZPZV<1>, ZPZV<0>, ZPZV<8>, ZPZV<239>, ZPZV<10»; };  // NOLINT
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
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
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
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»; };  // NOLINT
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
template<> struct ConwayPolynomial<317, 1> { using ZPZ = aerobus::zpz<317>; using type = POLYV<ZPZV<1>, ZPZV<315»; };  // NOLINT
template<> struct ConwayPolynomial<317, 2> { using ZPZ = aerobus::zpz<317>; using type = POLYV<ZPZV<1>, ZPZV<313>, ZPZV<2»; };  // NOLINT
template<> struct ConwayPolynomial<317, 3> { using ZPZ = aerobus::zpz<317>; using type = POLYV<ZPZV<1>, ZPZV<0>, ZPZV<7>, ZPZV<315»; };  // NOLINT
template<> struct ConwayPolynomial<317, 4> { using ZPZ = aerobus::zpz<317>; using type = POLYV<ZPZV<1>, ZPZV<0>, ZPZV<6>, ZPZV<178>, ZPZV<2»; };  // NOLINT
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
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
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
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
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<284>, ZPZV<296>, ZPZV<315»; };  // NOLINT
template<> struct ConwayPolynomial<331, 1> { using ZPZ = aerobus::zpz<331>; using type = POLYV<ZPZV<1>, ZPZV<328»; };  // NOLINT
template<> struct ConwayPolynomial<331, 2> { using ZPZ = aerobus::zpz<331>; using type = POLYV<ZPZV<1>, ZPZV<326>, ZPZV<3»; };  // NOLINT
template<> struct ConwayPolynomial<331, 3> { using ZPZ = aerobus::zpz<331>; using type = POLYV<ZPZV<1>, ZPZV<0>, ZPZV<1>, ZPZV<328»; };  // NOLINT
template<> struct ConwayPolynomial<331, 4> { using ZPZ = aerobus::zpz<331>; using type = POLYV<ZPZV<1>, ZPZV<0>, ZPZV<3>, ZPZV<290>, ZPZV<3»; };  // NOLINT
template<> struct ConwayPolynomial<331, 5> { using ZPZ = aerobus::zpz<331>; using type = POLYV<ZPZV<1>, ZPZV<0>, ZPZV<0>, ZPZV<0>, ZPZV<4>, ZPZV<328»; };  // NOLINT
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
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»; };  // NOLINT
template<> struct ConwayPolynomial<331, 8> { using ZPZ = aerobus::zpz<331>; using type = POLYV<ZPZV<1>, ZPZV<0>, ZPZV<0>, ZPZV<0>, ZPZV<1>, ZPZV<249>, ZPZV<308>, ZPZV<78>, ZPZV<3»; };  // NOLINT
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<194>, ZPZV<210>, ZPZV<328»; };  // NOLINT
template<> struct ConwayPolynomial<337, 1> { using ZPZ = aerobus::zpz<337>; using type = POLYV<ZPZV<1>, ZPZV<327»; };  // NOLINT
template<> struct ConwayPolynomial<337, 2> { using ZPZ = aerobus::zpz<337>; using type = POLYV<ZPZV<1>, ZPZV<332>, ZPZV<10»; };  // NOLINT
template<> struct ConwayPolynomial<337, 3> { using ZPZ = aerobus::zpz<337>; using type = POLYV<ZPZV<1>, ZPZV<0>, ZPZV<2>, ZPZV<327»; };  // NOLINT
template<> struct ConwayPolynomial<337, 4> { using ZPZ = aerobus::zpz<337>; using type = POLYV<ZPZV<1>, ZPZV<0>, ZPZV<25>, ZPZV<224>, ZPZV<10»; };  // NOLINT
template<> struct ConwayPolynomial<337, 5> { using ZPZ = aerobus::zpz<337>; using type = POLYV<ZPZV<1>, ZPZV<0>, ZPZV<0>, ZPZV<2>, ZPZV<327»; };  // NOLINT
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»; };  // NOLINT
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<5>, ZPZV<327»; };  // NOLINT
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
```

```
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<12>, ZPZV<148>, ZPZV<98>, ZPZV<327»; };  // NOLINT
template<> struct ConwayPolynomial<347, 1> { using ZPZ = aerobus::zpz<347>; using type = POLYV<ZPZV<1>, ZPZV<345»; };  // NOLINT
template<> struct ConwayPolynomial<347, 2> { using ZPZ = aerobus::zpz<347>; using type = POLYV<ZPZV<1>, ZPZV<343>, ZPZV<2»; };  // NOLINT
template<> struct ConwayPolynomial<347, 3> { using ZPZ = aerobus::zpz<347>; using type = POLYV<ZPZV<1>, ZPZV<0>, ZPZV<9>, ZPZV<345»; };  // NOLINT
template<> struct ConwayPolynomial<347, 4> { using ZPZ = aerobus::zpz<347>; using type = POLYV<ZPZV<1>, ZPZV<0>, ZPZV<13>, ZPZV<295>, ZPZV<2»; };  // NOLINT
template<> struct ConwayPolynomial<347, 5> { using ZPZ = aerobus::zpz<347>; using type = POLYV<ZPZV<1>, ZPZV<0>, ZPZV<0>, ZPZV<0>, ZPZV<3>, ZPZV<345»; };  // NOLINT
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
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»; };  // NOLINT
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<117>, ZPZV<2»; };  // NOLINT
template<> struct ConwayPolynomial<347, 9> { using ZPZ = aerobus::zpz<347>; using type = POLYV<ZPZV<1>, ZPZV<0>, ZPZV<0>, ZPZV<0>, ZPZV<0>, ZPZV<2>, ZPZV<235>, ZPZV<252>, ZPZV<345»; };  // NOLINT
template<> struct ConwayPolynomial<349, 1> { using ZPZ = aerobus::zpz<349>; using type = POLYV<ZPZV<1>, ZPZV<347»; };  // NOLINT
template<> struct ConwayPolynomial<349, 2> { using ZPZ = aerobus::zpz<349>; using type = POLYV<ZPZV<1>, ZPZV<348>, ZPZV<2»; };  // NOLINT
template<> struct ConwayPolynomial<349, 3> { using ZPZ = aerobus::zpz<349>; using type = POLYV<ZPZV<1>, ZPZV<0>, ZPZV<4>, ZPZV<347»; };  // NOLINT
template<> struct ConwayPolynomial<349, 4> { using ZPZ = aerobus::zpz<349>; using type = POLYV<ZPZV<1>, ZPZV<0>, ZPZV<3>, ZPZV<279>, ZPZV<2»; };  // NOLINT
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
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
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
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<2»; };  // NOLINT
template<> struct ConwayPolynomial<349, 9> { using ZPZ = aerobus::zpz<349>; using type = POLYV<ZPZV<1>, ZPZV<0>, ZPZV<0>, ZPZV<0>, ZPZV<0>, ZPZV<0>, ZPZV<36>, ZPZV<290>, ZPZV<130>, ZPZV<347»; };  // NOLINT
template<> struct ConwayPolynomial<353, 1> { using ZPZ = aerobus::zpz<353>; using type = POLYV<ZPZV<1>, ZPZV<350»; };  // NOLINT
template<> struct ConwayPolynomial<353, 2> { using ZPZ = aerobus::zpz<353>; using type = POLYV<ZPZV<1>, ZPZV<348>, ZPZV<3»; };  // NOLINT
template<> struct ConwayPolynomial<353, 3> { using ZPZ = aerobus::zpz<353>; using type = POLYV<ZPZV<1>, ZPZV<0>, ZPZV<3>, ZPZV<350»; };  // NOLINT
template<> struct ConwayPolynomial<353, 4> { using ZPZ = aerobus::zpz<353>; using type = POLYV<ZPZV<1>, ZPZV<0>, ZPZV<0>, ZPZV<199>, ZPZV<3»; };  // NOLINT
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
template<> struct ConwayPolynomial<353, 6> { using ZPZ = aerobus::zpz<353>; using type = POLYV<ZPZV<1>, ZPZV<0>, ZPZV<1>, ZPZV<215>, ZPZV<226>, ZPZV<295>, ZPZV<3»; };  // NOLINT
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<16>, ZPZV<350»; };  // NOLINT
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<3»; };  // NOLINT
template<> struct ConwayPolynomial<353, 9> { using ZPZ = aerobus::zpz<353>; using type = POLYV<ZPZV<1>, ZPZV<0>, ZPZV<0>, ZPZV<0>, ZPZV<0>, ZPZV<2>, ZPZV<319>, ZPZV<49>, ZPZV<350»; };  // NOLINT
template<> struct ConwayPolynomial<359, 1> { using ZPZ = aerobus::zpz<359>; using type = POLYV<ZPZV<1>, ZPZV<352»; };  // NOLINT
template<> struct ConwayPolynomial<359, 2> { using ZPZ = aerobus::zpz<359>; using type = POLYV<ZPZV<1>, ZPZV<358>, ZPZV<7»; };  // NOLINT
template<> struct ConwayPolynomial<359, 3> { using ZPZ = aerobus::zpz<359>; using type = POLYV<ZPZV<1>, ZPZV<0>, ZPZV<3>, ZPZV<352»; };  // NOLINT
template<> struct ConwayPolynomial<359, 4> { using ZPZ = aerobus::zpz<359>; using type = POLYV<ZPZV<1>, ZPZV<0>, ZPZV<2>, ZPZV<229>, ZPZV<7»; };  // NOLINT
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
template<> struct ConwayPolynomial<359, 6> { using ZPZ = aerobus::zpz<359>; using type = POLYV<ZPZV<1>, ZPZV<0>, ZPZV<4>, ZPZV<309>, ZPZV<327>, ZPZV<327>, ZPZV<7»; };  // NOLINT
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<1>, ZPZV<352»; };  // NOLINT
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
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<356>, ZPZV<165>, ZPZV<352»; };  // NOLINT
template<> struct ConwayPolynomial<367, 1> { using ZPZ = aerobus::zpz<367>; using type = POLYV<ZPZV<1>, ZPZV<361»; };  // NOLINT
template<> struct ConwayPolynomial<367, 2> { using ZPZ = aerobus::zpz<367>; using type = POLYV<ZPZV<1>, ZPZV<366>, ZPZV<6»; };  // NOLINT
```

```
template<> struct ConwayPolynomial<367, 3> { using ZPZ = aerobus::zpz<367>; using type = POLYV<ZPZV<1>, ZPZV<0>, ZPZV<10>, ZPZV<361»; };  // NOLINT
template<> struct ConwayPolynomial<367, 4> { using ZPZ = aerobus::zpz<367>; using type = POLYV<ZPZV<1>, ZPZV<0>, ZPZV<3>, ZPZV<295>, ZPZV<6»; };  // NOLINT
template<> struct ConwayPolynomial<367, 5> { using ZPZ = aerobus::zpz<367>; using type = POLYV<ZPZV<1>, ZPZV<0>, ZPZV<0>, ZPZV<0>, ZPZV<3>, ZPZV<361»; };  // NOLINT
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
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<13>, ZPZV<361»; };  // NOLINT
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»; };  // NOLINT
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<15>, ZPZV<213>, ZPZV<268>, ZPZV<361»; };  // NOLINT
template<> struct ConwayPolynomial<373, 1> { using ZPZ = aerobus::zpz<373>; using type = POLYV<ZPZV<1>, ZPZV<371»; };  // NOLINT
template<> struct ConwayPolynomial<373, 2> { using ZPZ = aerobus::zpz<373>; using type = POLYV<ZPZV<1>, ZPZV<369>, ZPZV<2»; };  // NOLINT
template<> struct ConwayPolynomial<373, 3> { using ZPZ = aerobus::zpz<373>; using type = POLYV<ZPZV<1>, ZPZV<0>, ZPZV<5>, ZPZV<371»; };  // NOLINT
template<> struct ConwayPolynomial<373, 4> { using ZPZ = aerobus::zpz<373>; using type = POLYV<ZPZV<1>, ZPZV<0>, ZPZV<15>, ZPZV<304>, ZPZV<2»; };  // NOLINT
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
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
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<7>, ZPZV<371»; };  // NOLINT
template<> struct ConwayPolynomial<373, 8> { using ZPZ = aerobus::zpz<373>; using type = POLYV<ZPZV<1>, ZPZV<0>, ZPZV<0>, ZPZV<0>, ZPZV<0>, ZPZV<203>, ZPZV<219>, ZPZV<66>, ZPZV<2»; };  // NOLINT
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
template<> struct ConwayPolynomial<379, 1> { using ZPZ = aerobus::zpz<379>; using type = POLYV<ZPZV<1>, ZPZV<377»; };  // NOLINT
template<> struct ConwayPolynomial<379, 2> { using ZPZ = aerobus::zpz<379>; using type = POLYV<ZPZV<1>, ZPZV<374>, ZPZV<2»; };  // NOLINT
template<> struct ConwayPolynomial<379, 3> { using ZPZ = aerobus::zpz<379>; using type = POLYV<ZPZV<1>, ZPZV<0>, ZPZV<5>, ZPZV<377»; };  // NOLINT
template<> struct ConwayPolynomial<379, 4> { using ZPZ = aerobus::zpz<379>; using type = POLYV<ZPZV<1>, ZPZV<0>, ZPZV<2>, ZPZV<327>, ZPZV<2»; };  // NOLINT
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
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
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<14>, ZPZV<377»; };  // NOLINT
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»; };  // NOLINT
template<> struct ConwayPolynomial<379, 9> { using ZPZ = aerobus::zpz<379>; using type = POLYV<ZPZV<1>, ZPZV<0>, ZPZV<0>, ZPZV<0>, ZPZV<0>, ZPZV<0>, ZPZV<11>, ZPZV<362>, ZPZV<369>, ZPZV<377»; };  // NOLINT
template<> struct ConwayPolynomial<383, 1> { using ZPZ = aerobus::zpz<383>; using type = POLYV<ZPZV<1>, ZPZV<378»; };  // NOLINT
template<> struct ConwayPolynomial<383, 2> { using ZPZ = aerobus::zpz<383>; using type = POLYV<ZPZV<1>, ZPZV<382>, ZPZV<5»; };  // NOLINT
template<> struct ConwayPolynomial<383, 3> { using ZPZ = aerobus::zpz<383>; using type = POLYV<ZPZV<1>, ZPZV<0>, ZPZV<1>, ZPZV<378»; };  // NOLINT
template<> struct ConwayPolynomial<383, 4> { using ZPZ = aerobus::zpz<383>; using type = POLYV<ZPZV<1>, ZPZV<0>, ZPZV<7>, ZPZV<309>, ZPZV<5»; };  // NOLINT
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
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
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<6>, ZPZV<378»; };  // NOLINT
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
template<> struct ConwayPolynomial<383, 9> { using ZPZ = aerobus::zpz<383>; using type = POLYV<ZPZV<1>, ZPZV<0>, ZPZV<0>, ZPZV<0>, ZPZV<0>, ZPZV<0>, ZPZV<2>, ZPZV<137>, ZPZV<76>, ZPZV<378»; };  // NOLINT
template<> struct ConwayPolynomial<389, 1> { using ZPZ = aerobus::zpz<389>; using type = POLYV<ZPZV<1>, ZPZV<387»; };  // NOLINT
template<> struct ConwayPolynomial<389, 2> { using ZPZ = aerobus::zpz<389>; using type = POLYV<ZPZV<1>, ZPZV<379>, ZPZV<2»; };  // NOLINT
template<> struct ConwayPolynomial<389, 3> { using ZPZ = aerobus::zpz<389>; using type = POLYV<ZPZV<1>, ZPZV<0>, ZPZV<2>, ZPZV<387»; };  // NOLINT
template<> struct ConwayPolynomial<389, 4> { using ZPZ = aerobus::zpz<389>; using type = POLYV<ZPZV<1>, ZPZV<0>, ZPZV<2>, ZPZV<266>, ZPZV<2»; };  // NOLINT
template<> struct ConwayPolynomial<389, 5> { using ZPZ = aerobus::zpz<389>; using type = POLYV<ZPZV<1>, ZPZV<0>, ZPZV<0>, ZPZV<0>, ZPZV<4>, ZPZV<387»; };  // NOLINT
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
template<> struct ConwayPolynomial<389, 7> { using ZPZ = aerobus::zpz<389>; using type =
POLYV<ZPZV<1>, ZPZV<0>, ZPZV<0>, ZPZV<0>, ZPZV<0>, ZPZV<24>, ZPZV<387»; };  // NOLINT
template<> struct ConwayPolynomial<389, 8> { using ZPZ = aerobus::zpz<389>; using type =
POLYV<ZPZV<1>, ZPZV<0>, ZPZV<0>, ZPZV<0>, ZPZV<0>, ZPZV<351>, ZPZV<19>, ZPZV<290>, ZPZV<2»; };  //
NOLINT
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
template<> struct ConwayPolynomial<397, 1> { using ZPZ = aerobus::zpz<397>; using type =
POLYV<ZPZV<1>, ZPZV<392»; };  // NOLINT
template<> struct ConwayPolynomial<397, 2> { using ZPZ = aerobus::zpz<397>; using type =
POLYV<ZPZV<1>, ZPZV<392>, ZPZV<5»; };  // NOLINT
template<> struct ConwayPolynomial<397, 3> { using ZPZ = aerobus::zpz<397>; using type =
POLYV<ZPZV<1>, ZPZV<0>, ZPZV<2>, ZPZV<392»; };  // NOLINT
template<> struct ConwayPolynomial<397, 4> { using ZPZ = aerobus::zpz<397>; using type =
POLYV<ZPZV<1>, ZPZV<0>, ZPZV<12>, ZPZV<363>, ZPZV<5»; };  // NOLINT
template<> struct ConwayPolynomial<397, 5> { using ZPZ = aerobus::zpz<397>; using type =
POLYV<ZPZV<1>, ZPZV<0>, ZPZV<0>, ZPZV<0>, ZPZV<7>, ZPZV<392»; };  // NOLINT
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»; };  // NOLINT
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<12>, ZPZV<392»; };  // NOLINT
template<> struct ConwayPolynomial<397, 8> { using ZPZ = aerobus::zpz<397>; using type =
POLYV<ZPZV<1>, ZPZV<0>, ZPZV<0>, ZPZV<0>, ZPZV<1>, ZPZV<375>, ZPZV<255>, ZPZV<203>, ZPZV<5»; };  //
NOLINT
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<252>, ZPZV<392»;
};  // NOLINT
template<> struct ConwayPolynomial<401, 1> { using ZPZ = aerobus::zpz<401>; using type =
POLYV<ZPZV<1>, ZPZV<398»; };  // NOLINT
template<> struct ConwayPolynomial<401, 2> { using ZPZ = aerobus::zpz<401>; using type =
POLYV<ZPZV<1>, ZPZV<396>, ZPZV<3»; };  // NOLINT
template<> struct ConwayPolynomial<401, 3> { using ZPZ = aerobus::zpz<401>; using type =
POLYV<ZPZV<1>, ZPZV<0>, ZPZV<3>, ZPZV<398»; };  // NOLINT
template<> struct ConwayPolynomial<401, 4> { using ZPZ = aerobus::zpz<401>; using type =
POLYV<ZPZV<1>, ZPZV<0>, ZPZV<2>, ZPZV<372>, ZPZV<3»; };  // NOLINT
template<> struct ConwayPolynomial<401, 5> { using ZPZ = aerobus::zpz<401>; using type =
POLYV<ZPZV<1>, ZPZV<0>, ZPZV<0>, ZPZV<0>, ZPZV<4>, ZPZV<398»; };  // NOLINT
template<> struct ConwayPolynomial<401, 6> { using ZPZ = aerobus::zpz<401>; using type =
POLYV<ZPZV<1>, ZPZV<0>, ZPZV<4>, ZPZV<115>, ZPZV<81>, ZPZV<51>, ZPZV<3»; };  // NOLINT
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<398»; };  // NOLINT
template<> struct ConwayPolynomial<401, 8> { using ZPZ = aerobus::zpz<401>; using type =
POLYV<ZPZV<1>, ZPZV<0>, ZPZV<0>, ZPZV<0>, ZPZV<380>, ZPZV<113>, ZPZV<164>, ZPZV<3»; };  //
NOLINT
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<199>, ZPZV<158>, ZPZV<398»;
};  // NOLINT
template<> struct ConwayPolynomial<409, 1> { using ZPZ = aerobus::zpz<409>; using type =
POLYV<ZPZV<1>, ZPZV<388»; };  // NOLINT
template<> struct ConwayPolynomial<409, 2> { using ZPZ = aerobus::zpz<409>; using type =
POLYV<ZPZV<1>, ZPZV<404>, ZPZV<21»; };  // NOLINT
template<> struct ConwayPolynomial<409, 3> { using ZPZ = aerobus::zpz<409>; using type =
POLYV<ZPZV<1>, ZPZV<0>, ZPZV<3>, ZPZV<388»; };  // NOLINT
template<> struct ConwayPolynomial<409, 4> { using ZPZ = aerobus::zpz<409>; using type =
POLYV<ZPZV<1>, ZPZV<0>, ZPZV<12>, ZPZV<407>, ZPZV<21»; };  // NOLINT
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
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
template<> struct ConwayPolynomial<409, 7> { using ZPZ = aerobus::zpz<409>; using type =
POLYV<ZPZV<1>, ZPZV<0>, ZPZV<0>, ZPZV<0>, ZPZV<0>, ZPZV<0>, ZPZV<7>, ZPZV<388»; };  // NOLINT
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<21»; };  //
NOLINT
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<8>, ZPZV<318>, ZPZV<211>, ZPZV<388»;
};  // NOLINT
template<> struct ConwayPolynomial<419, 1> { using ZPZ = aerobus::zpz<419>; using type =
POLYV<ZPZV<1>, ZPZV<417»; };  // NOLINT
template<> struct ConwayPolynomial<419, 2> { using ZPZ = aerobus::zpz<419>; using type =
POLYV<ZPZV<1>, ZPZV<418>, ZPZV<2»; };  // NOLINT
template<> struct ConwayPolynomial<419, 3> { using ZPZ = aerobus::zpz<419>; using type =
POLYV<ZPZV<1>, ZPZV<0>, ZPZV<11>, ZPZV<417»; };  // NOLINT
template<> struct ConwayPolynomial<419, 4> { using ZPZ = aerobus::zpz<419>; using type =
POLYV<ZPZV<1>, ZPZV<0>, ZPZV<4>, ZPZV<373>, ZPZV<2»; };  // NOLINT
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
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
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<4>, ZPZV<417»; };  // NOLINT
template<> struct ConwayPolynomial<419, 8> { using ZPZ = aerobus::zpz<419>; using type =
POLYV<ZPZV<1>, ZPZV<0>, ZPZV<0>, ZPZV<8>, ZPZV<234>, ZPZV<388>, ZPZV<151>, ZPZV<2»; };  //
NOLINT
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<93>, ZPZV<386>, ZPZV<417»;
}; // NOLINT
template<> struct ConwayPolynomial<421, 1> { using ZPZ = aerobus::zpz<421>; using type =
POLYV<ZPZV<1>, ZPZV<419»; }; // NOLINT
template<> struct ConwayPolynomial<421, 2> { using ZPZ = aerobus::zpz<421>; using type =
POLYV<ZPZV<1>, ZPZV<417>, ZPZV<2»; }; // NOLINT
template<> struct ConwayPolynomial<421, 3> { using ZPZ = aerobus::zpz<421>; using type =
POLYV<ZPZV<1>, ZPZV<0>, ZPZV<2>, ZPZV<419»; }; // NOLINT
template<> struct ConwayPolynomial<421, 4> { using ZPZ = aerobus::zpz<421>; using type =
POLYV<ZPZV<1>, ZPZV<0>, ZPZV<10>, ZPZV<257>, ZPZV<2»; }; // NOLINT
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
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
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»; }; // NOLINT
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<32>, ZPZV<77>, ZPZV<2»; }; //
NOLINT
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
template<> struct ConwayPolynomial<431, 1> { using ZPZ = aerobus::zpz<431>; using type =
POLYV<ZPZV<1>, ZPZV<424»; }; // NOLINT
template<> struct ConwayPolynomial<431, 2> { using ZPZ = aerobus::zpz<431>; using type =
POLYV<ZPZV<1>, ZPZV<430>, ZPZV<7»; }; // NOLINT
template<> struct ConwayPolynomial<431, 3> { using ZPZ = aerobus::zpz<431>; using type =
POLYV<ZPZV<1>, ZPZV<0>, ZPZV<2>, ZPZV<424»; }; // NOLINT
template<> struct ConwayPolynomial<431, 4> { using ZPZ = aerobus::zpz<431>; using type =
POLYV<ZPZV<1>, ZPZV<0>, ZPZV<2>, ZPZV<323>, ZPZV<7»; }; // NOLINT
template<> struct ConwayPolynomial<431, 5> { using ZPZ = aerobus::zpz<431>; using type =
POLYV<ZPZV<1>, ZPZV<0>, ZPZV<0>, ZPZV<0>, ZPZV<7>, ZPZV<424»; }; // NOLINT
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
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<1>, ZPZV<424»; }; // NOLINT
template<> struct ConwayPolynomial<431, 8> { using ZPZ = aerobus::zpz<431>; using type =
POLYV<ZPZV<1>, ZPZV<0>, ZPZV<0>, ZPZV<0>, ZPZV<5>, ZPZV<243>, ZPZV<286>, ZPZV<115>, ZPZV<7»; }; //
NOLINT
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
template<> struct ConwayPolynomial<433, 1> { using ZPZ = aerobus::zpz<433>; using type =
POLYV<ZPZV<1>, ZPZV<428»; }; // NOLINT
template<> struct ConwayPolynomial<433, 2> { using ZPZ = aerobus::zpz<433>; using type =
POLYV<ZPZV<1>, ZPZV<432>, ZPZV<5»; }; // NOLINT
template<> struct ConwayPolynomial<433, 3> { using ZPZ = aerobus::zpz<433>; using type =
POLYV<ZPZV<1>, ZPZV<0>, ZPZV<1>, ZPZV<428»; }; // NOLINT
template<> struct ConwayPolynomial<433, 4> { using ZPZ = aerobus::zpz<433>; using type =
POLYV<ZPZV<1>, ZPZV<0>, ZPZV<6>, ZPZV<402>, ZPZV<5»; }; // NOLINT
template<> struct ConwayPolynomial<433, 5> { using ZPZ = aerobus::zpz<433>; using type =
POLYV<ZPZV<1>, ZPZV<0>, ZPZV<0>, ZPZV<0>, ZPZV<5>, ZPZV<428»; }; // NOLINT
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
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<428»; }; // NOLINT
template<> struct ConwayPolynomial<433, 8> { using ZPZ = aerobus::zpz<433>; using type =
POLYV<ZPZV<1>, ZPZV<0>, ZPZV<0>, ZPZV<10>, ZPZV<347>, ZPZV<32>, ZPZV<39>, ZPZV<5»; }; //
NOLINT
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
template<> struct ConwayPolynomial<439, 1> { using ZPZ = aerobus::zpz<439>; using type =
POLYV<ZPZV<1>, ZPZV<424»; }; // NOLINT
template<> struct ConwayPolynomial<439, 2> { using ZPZ = aerobus::zpz<439>; using type =
POLYV<ZPZV<1>, ZPZV<436>, ZPZV<15»; }; // NOLINT
template<> struct ConwayPolynomial<439, 3> { using ZPZ = aerobus::zpz<439>; using type =
POLYV<ZPZV<1>, ZPZV<0>, ZPZV<3>, ZPZV<424»; }; // NOLINT
template<> struct ConwayPolynomial<439, 4> { using ZPZ = aerobus::zpz<439>; using type =
POLYV<ZPZV<1>, ZPZV<0>, ZPZV<0>, ZPZV<323>, ZPZV<15»; }; // NOLINT
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
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
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<1>, ZPZV<424»; }; // NOLINT
template<> struct ConwayPolynomial<439, 8> { using ZPZ = aerobus::zpz<439>; using type =
POLYV<ZPZV<1>, ZPZV<0>, ZPZV<0>, ZPZV<0>, ZPZV<0>, ZPZV<359>, ZPZV<296>, ZPZV<266>, ZPZV<15»; }; //
NOLINT
template<> struct ConwayPolynomial<439, 9> { using ZPZ = aerobus::zpz<439>; using type =
POLYV<ZPZV<1>, ZPZV<0>, ZPZV<0>, ZPZV<0>, ZPZV<0>, ZPZV<0>, ZPZV<16>, ZPZV<342>, ZPZV<254>, ZPZV<424»;
}; // NOLINT
template<> struct ConwayPolynomial<443, 1> { using ZPZ = aerobus::zpz<443>; using type =
POLYV<ZPZV<1>, ZPZV<441»; }; // NOLINT
template<> struct ConwayPolynomial<443, 2> { using ZPZ = aerobus::zpz<443>; using type =
POLYV<ZPZV<1>, ZPZV<437>, ZPZV<2»; }; // NOLINT
template<> struct ConwayPolynomial<443, 3> { using ZPZ = aerobus::zpz<443>; using type =
POLYV<ZPZV<1>, ZPZV<0>, ZPZV<4>, ZPZV<441»; };  // NOLINT
template<> struct ConwayPolynomial<443, 4> { using ZPZ = aerobus::zpz<443>; using type = POLYV<ZPZV<1>, ZPZV<0>, ZPZV<4>, ZPZV<383>, ZPZV<2»; };  // NOLINT
template<> struct ConwayPolynomial<443, 5> { using ZPZ = aerobus::zpz<443>; using type = POLYV<ZPZV<1>, ZPZV<0>, ZPZV<0>, ZPZV<0>, ZPZV<4>, ZPZV<441»; };  // NOLINT
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
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<441»; };  // NOLINT
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
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<3>, ZPZV<125>, ZPZV<109>, ZPZV<441»; };  // NOLINT
template<> struct ConwayPolynomial<449, 1> { using ZPZ = aerobus::zpz<449>; using type = POLYV<ZPZV<1>, ZPZV<446»; };  // NOLINT
template<> struct ConwayPolynomial<449, 2> { using ZPZ = aerobus::zpz<449>; using type = POLYV<ZPZV<1>, ZPZV<444>, ZPZV<3»; };  // NOLINT
template<> struct ConwayPolynomial<449, 3> { using ZPZ = aerobus::zpz<449>; using type = POLYV<ZPZV<1>, ZPZV<0>, ZPZV<1>, ZPZV<446»; };  // NOLINT
template<> struct ConwayPolynomial<449, 4> { using ZPZ = aerobus::zpz<449>; using type = POLYV<ZPZV<1>, ZPZV<0>, ZPZV<2>, ZPZV<249>, ZPZV<3»; };  // NOLINT
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
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
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<28>, ZPZV<446»; };  // NOLINT
template<> struct ConwayPolynomial<449, 8> { using ZPZ = aerobus::zpz<449>; using type = POLYV<ZPZV<1>, ZPZV<0>, ZPZV<0>, ZPZV<0>, ZPZV<0>, ZPZV<361>, ZPZV<348>, ZPZV<124>, ZPZV<3»; };  // NOLINT
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<446»; };  // NOLINT
template<> struct ConwayPolynomial<457, 1> { using ZPZ = aerobus::zpz<457>; using type = POLYV<ZPZV<1>, ZPZV<444»; };  // NOLINT
template<> struct ConwayPolynomial<457, 2> { using ZPZ = aerobus::zpz<457>; using type = POLYV<ZPZV<1>, ZPZV<454>, ZPZV<13»; };  // NOLINT
template<> struct ConwayPolynomial<457, 3> { using ZPZ = aerobus::zpz<457>; using type = POLYV<ZPZV<1>, ZPZV<0>, ZPZV<1>, ZPZV<444»; };  // NOLINT
template<> struct ConwayPolynomial<457, 4> { using ZPZ = aerobus::zpz<457>; using type = POLYV<ZPZV<1>, ZPZV<0>, ZPZV<8>, ZPZV<407>, ZPZV<13»; };  // NOLINT
template<> struct ConwayPolynomial<457, 5> { using ZPZ = aerobus::zpz<457>; using type = POLYV<ZPZV<1>, ZPZV<0>, ZPZV<0>, ZPZV<0>, ZPZV<4>, ZPZV<444»; };  // NOLINT
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
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
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
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
template<> struct ConwayPolynomial<461, 1> { using ZPZ = aerobus::zpz<461>; using type = POLYV<ZPZV<1>, ZPZV<459»; };  // NOLINT
template<> struct ConwayPolynomial<461, 2> { using ZPZ = aerobus::zpz<461>; using type = POLYV<ZPZV<1>, ZPZV<460>, ZPZV<2»; };  // NOLINT
template<> struct ConwayPolynomial<461, 3> { using ZPZ = aerobus::zpz<461>; using type = POLYV<ZPZV<1>, ZPZV<0>, ZPZV<3>, ZPZV<459»; };  // NOLINT
template<> struct ConwayPolynomial<461, 4> { using ZPZ = aerobus::zpz<461>; using type = POLYV<ZPZV<1>, ZPZV<0>, ZPZV<3>, ZPZV<393>, ZPZV<2»; };  // NOLINT
template<> struct ConwayPolynomial<461, 5> { using ZPZ = aerobus::zpz<461>; using type = POLYV<ZPZV<1>, ZPZV<0>, ZPZV<0>, ZPZV<0>, ZPZV<14>, ZPZV<459»; };  // NOLINT
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
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
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<2»; };  // NOLINT
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<210>, ZPZV<276>, ZPZV<459»; };  // NOLINT
template<> struct ConwayPolynomial<463, 1> { using ZPZ = aerobus::zpz<463>; using type = POLYV<ZPZV<1>, ZPZV<460»; };  // NOLINT
template<> struct ConwayPolynomial<463, 2> { using ZPZ = aerobus::zpz<463>; using type = POLYV<ZPZV<1>, ZPZV<461>, ZPZV<3»; };  // NOLINT
template<> struct ConwayPolynomial<463, 3> { using ZPZ = aerobus::zpz<463>; using type = POLYV<ZPZV<1>, ZPZV<0>, ZPZV<10>, ZPZV<460»; };  // NOLINT
template<> struct ConwayPolynomial<463, 4> { using ZPZ = aerobus::zpz<463>; using type = POLYV<ZPZV<1>, ZPZV<0>, ZPZV<17>, ZPZV<262>, ZPZV<3»; };  // NOLINT
template<> struct ConwayPolynomial<463, 5> { using ZPZ = aerobus::zpz<463>; using type = POLYV<ZPZV<1>, ZPZV<0>, ZPZV<0>, ZPZV<0>, ZPZV<3>, ZPZV<460»; };  // NOLINT
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
```

```
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<13>, ZPZV<460»; };  // NOLINT
template<> struct ConwayPolynomial<463, 8> { using ZPZ = aerobus::zpz<463>; using type = POLYV<ZPZV<1>, ZPZV<0>, ZPZV<0>, ZPZV<0>, ZPZV<234>, ZPZV<414>, ZPZV<396>, ZPZV<3»; };  // NOLINT
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<227>, ZPZV<460»; };  // NOLINT
template<> struct ConwayPolynomial<467, 1> { using ZPZ = aerobus::zpz<467>; using type = POLYV<ZPZV<1>, ZPZV<465»; };  // NOLINT
template<> struct ConwayPolynomial<467, 2> { using ZPZ = aerobus::zpz<467>; using type = POLYV<ZPZV<1>, ZPZV<463>, ZPZV<2»; };  // NOLINT
template<> struct ConwayPolynomial<467, 3> { using ZPZ = aerobus::zpz<467>; using type = POLYV<ZPZV<1>, ZPZV<0>, ZPZV<2>, ZPZV<465»; };  // NOLINT
template<> struct ConwayPolynomial<467, 4> { using ZPZ = aerobus::zpz<467>; using type = POLYV<ZPZV<1>, ZPZV<0>, ZPZV<14>, ZPZV<353>, ZPZV<2»; };  // NOLINT
template<> struct ConwayPolynomial<467, 5> { using ZPZ = aerobus::zpz<467>; using type = POLYV<ZPZV<1>, ZPZV<0>, ZPZV<0>, ZPZV<0>, ZPZV<2>, ZPZV<465»; };  // NOLINT
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
template<> struct ConwayPolynomial<467, 7> { using ZPZ = aerobus::zpz<467>; using type = POLYV<ZPZV<1>, ZPZV<0>, ZPZV<0>, ZPZV<0>, ZPZV<0>, ZPZV<0>, ZPZV<7>, ZPZV<465»; };  // NOLINT
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
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<447>, ZPZV<465»; };  // NOLINT
template<> struct ConwayPolynomial<479, 1> { using ZPZ = aerobus::zpz<479>; using type = POLYV<ZPZV<1>, ZPZV<466»; };  // NOLINT
template<> struct ConwayPolynomial<479, 2> { using ZPZ = aerobus::zpz<479>; using type = POLYV<ZPZV<1>, ZPZV<474>, ZPZV<13»; };  // NOLINT
template<> struct ConwayPolynomial<479, 3> { using ZPZ = aerobus::zpz<479>; using type = POLYV<ZPZV<1>, ZPZV<0>, ZPZV<4>, ZPZV<466»; };  // NOLINT
template<> struct ConwayPolynomial<479, 4> { using ZPZ = aerobus::zpz<479>; using type = POLYV<ZPZV<1>, ZPZV<0>, ZPZV<6>, ZPZV<386>, ZPZV<13»; };  // NOLINT
template<> struct ConwayPolynomial<479, 5> { using ZPZ = aerobus::zpz<479>; using type = POLYV<ZPZV<1>, ZPZV<0>, ZPZV<0>, ZPZV<0>, ZPZV<2>, ZPZV<466»; };  // NOLINT
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
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<466»; };  // NOLINT
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
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
template<> struct ConwayPolynomial<487, 1> { using ZPZ = aerobus::zpz<487>; using type = POLYV<ZPZV<1>, ZPZV<484»; };  // NOLINT
template<> struct ConwayPolynomial<487, 2> { using ZPZ = aerobus::zpz<487>; using type = POLYV<ZPZV<1>, ZPZV<485>, ZPZV<3»; };  // NOLINT
template<> struct ConwayPolynomial<487, 3> { using ZPZ = aerobus::zpz<487>; using type = POLYV<ZPZV<1>, ZPZV<0>, ZPZV<4>, ZPZV<484»; };  // NOLINT
template<> struct ConwayPolynomial<487, 4> { using ZPZ = aerobus::zpz<487>; using type = POLYV<ZPZV<1>, ZPZV<0>, ZPZV<4>, ZPZV<483>, ZPZV<3»; };  // NOLINT
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
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
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<484»; };  // NOLINT
template<> struct ConwayPolynomial<487, 8> { using ZPZ = aerobus::zpz<487>; using type = POLYV<ZPZV<1>, ZPZV<0>, ZPZV<0>, ZPZV<0>, ZPZV<1>, ZPZV<283>, ZPZV<249>, ZPZV<137>, ZPZV<3»; };  // NOLINT
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<271>, ZPZV<447>, ZPZV<484»; };  // NOLINT
template<> struct ConwayPolynomial<491, 1> { using ZPZ = aerobus::zpz<491>; using type = POLYV<ZPZV<1>, ZPZV<489»; };  // NOLINT
template<> struct ConwayPolynomial<491, 2> { using ZPZ = aerobus::zpz<491>; using type = POLYV<ZPZV<1>, ZPZV<487>, ZPZV<2»; };  // NOLINT
template<> struct ConwayPolynomial<491, 3> { using ZPZ = aerobus::zpz<491>; using type = POLYV<ZPZV<1>, ZPZV<0>, ZPZV<2>, ZPZV<489»; };  // NOLINT
template<> struct ConwayPolynomial<491, 4> { using ZPZ = aerobus::zpz<491>; using type = POLYV<ZPZV<1>, ZPZV<0>, ZPZV<7>, ZPZV<360>, ZPZV<2»; };  // NOLINT
template<> struct ConwayPolynomial<491, 5> { using ZPZ = aerobus::zpz<491>; using type = POLYV<ZPZV<1>, ZPZV<0>, ZPZV<0>, ZPZV<0>, ZPZV<5>, ZPZV<489»; };  // NOLINT
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<2»; };  // NOLINT
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»; };  // NOLINT
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
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<0>, ZPZV<149>, ZPZV<453>, ZPZV<489»;
};  // NOLINT
template<> struct ConwayPolynomial<499, 1> { using ZPZ = aerobus::zpz<499>; using type = POLYV<ZPZV<1>, ZPZV<492»; };  // NOLINT
template<> struct ConwayPolynomial<499, 2> { using ZPZ = aerobus::zpz<499>; using type = POLYV<ZPZV<1>, ZPZV<493>, ZPZV<7»; };  // NOLINT
template<> struct ConwayPolynomial<499, 3> { using ZPZ = aerobus::zpz<499>; using type = POLYV<ZPZV<1>, ZPZV<0>, ZPZV<2>, ZPZV<492»; };  // NOLINT
template<> struct ConwayPolynomial<499, 4> { using ZPZ = aerobus::zpz<499>; using type = POLYV<ZPZV<1>, ZPZV<0>, ZPZV<4>, ZPZV<495>, ZPZV<7»; };  // NOLINT
template<> struct ConwayPolynomial<499, 5> { using ZPZ = aerobus::zpz<499>; using type = POLYV<ZPZV<1>, ZPZV<0>, ZPZV<0>, ZPZV<0>, ZPZV<17>, ZPZV<492»; };  // NOLINT
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
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<8>, ZPZV<492»; };  // NOLINT
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
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
template<> struct ConwayPolynomial<503, 1> { using ZPZ = aerobus::zpz<503>; using type = POLYV<ZPZV<1>, ZPZV<498»; };  // NOLINT
template<> struct ConwayPolynomial<503, 2> { using ZPZ = aerobus::zpz<503>; using type = POLYV<ZPZV<1>, ZPZV<498>, ZPZV<5»; };  // NOLINT
template<> struct ConwayPolynomial<503, 3> { using ZPZ = aerobus::zpz<503>; using type = POLYV<ZPZV<1>, ZPZV<0>, ZPZV<2>, ZPZV<498»; };  // NOLINT
template<> struct ConwayPolynomial<503, 4> { using ZPZ = aerobus::zpz<503>; using type = POLYV<ZPZV<1>, ZPZV<0>, ZPZV<6>, ZPZV<325>, ZPZV<5»; };  // NOLINT
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
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
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<11>, ZPZV<498»; };  // NOLINT
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
template<> struct ConwayPolynomial<503, 9> { using ZPZ = aerobus::zpz<503>; using type = POLYV<ZPZV<1>, ZPZV<0>, ZPZV<0>, ZPZV<0>, ZPZV<0>, ZPZV<3>, ZPZV<158>, ZPZV<337>, ZPZV<498»; };  // NOLINT
template<> struct ConwayPolynomial<509, 1> { using ZPZ = aerobus::zpz<509>; using type = POLYV<ZPZV<1>, ZPZV<507»; };  // NOLINT
template<> struct ConwayPolynomial<509, 2> { using ZPZ = aerobus::zpz<509>; using type = POLYV<ZPZV<1>, ZPZV<508>, ZPZV<2»; };  // NOLINT
template<> struct ConwayPolynomial<509, 3> { using ZPZ = aerobus::zpz<509>; using type = POLYV<ZPZV<1>, ZPZV<0>, ZPZV<3>, ZPZV<507»; };  // NOLINT
template<> struct ConwayPolynomial<509, 4> { using ZPZ = aerobus::zpz<509>; using type = POLYV<ZPZV<1>, ZPZV<0>, ZPZV<4>, ZPZV<408>, ZPZV<2»; };  // NOLINT
template<> struct ConwayPolynomial<509, 5> { using ZPZ = aerobus::zpz<509>; using type = POLYV<ZPZV<1>, ZPZV<0>, ZPZV<0>, ZPZV<0>, ZPZV<3>, ZPZV<507»; };  // NOLINT
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
template<> struct ConwayPolynomial<509, 7> { using ZPZ = aerobus::zpz<509>; using type = POLYV<ZPZV<1>, ZPZV<0>, ZPZV<0>, ZPZV<0>, ZPZV<0>, ZPZV<0>, ZPZV<6>, ZPZV<507»; };  // NOLINT
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»; };  // NOLINT
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<314>, ZPZV<28>, ZPZV<507»; };  // NOLINT
template<> struct ConwayPolynomial<521, 1> { using ZPZ = aerobus::zpz<521>; using type = POLYV<ZPZV<1>, ZPZV<518»; };  // NOLINT
template<> struct ConwayPolynomial<521, 2> { using ZPZ = aerobus::zpz<521>; using type = POLYV<ZPZV<1>, ZPZV<515>, ZPZV<3»; };  // NOLINT
template<> struct ConwayPolynomial<521, 3> { using ZPZ = aerobus::zpz<521>; using type = POLYV<ZPZV<1>, ZPZV<0>, ZPZV<4>, ZPZV<518»; };  // NOLINT
template<> struct ConwayPolynomial<521, 4> { using ZPZ = aerobus::zpz<521>; using type = POLYV<ZPZV<1>, ZPZV<0>, ZPZV<0>, ZPZV<509>, ZPZV<3»; };  // NOLINT
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
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
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»; };  // NOLINT
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»; };  // NOLINT
template<> struct ConwayPolynomial<521, 9> { using ZPZ = aerobus::zpz<521>; using type = POLYV<ZPZV<1>, ZPZV<0>, ZPZV<0>, ZPZV<0>, ZPZV<0>, ZPZV<5>, ZPZV<181>, ZPZV<483>, ZPZV<518»; };  // NOLINT
template<> struct ConwayPolynomial<523, 1> { using ZPZ = aerobus::zpz<523>; using type = POLYV<ZPZV<1>, ZPZV<521»; };  // NOLINT
template<> struct ConwayPolynomial<523, 2> { using ZPZ = aerobus::zpz<523>; using type = POLYV<ZPZV<1>, ZPZV<522>, ZPZV<2»; };  // NOLINT
template<> struct ConwayPolynomial<523, 3> { using ZPZ = aerobus::zpz<523>; using type = POLYV<ZPZV<1>, ZPZV<0>, ZPZV<5>, ZPZV<521»; };  // NOLINT
```

```
template<> struct ConwayPolynomial<523, 4> { using ZPZ = aerobus::zpz<523>; using type = POLYV<ZPZV<1>, ZPZV<0>, ZPZV<2>, ZPZV<382>, ZPZV<2»; };  // NOLINT
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
template<> struct ConwayPolynomial<523, 6> { using ZPZ = aerobus::zpz<523>; using type = POLYV<ZPZV<1>, ZPZV<0>, ZPZV<0>, ZPZV<475>, ZPZV<475>, ZPZV<371>, ZPZV<2»; };  // NOLINT
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»; };  // NOLINT
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
template<> struct ConwayPolynomial<523, 9> { using ZPZ = aerobus::zpz<523>; using type = POLYV<ZPZV<1>, ZPZV<0>, ZPZV<0>, ZPZV<0>, ZPZV<0>, ZPZV<0>, ZPZV<19>, ZPZV<342>, ZPZV<145>, ZPZV<521»; };  // NOLINT
template<> struct ConwayPolynomial<541, 1> { using ZPZ = aerobus::zpz<541>; using type = POLYV<ZPZV<1>, ZPZV<539»; };  // NOLINT
template<> struct ConwayPolynomial<541, 2> { using ZPZ = aerobus::zpz<541>; using type = POLYV<ZPZV<1>, ZPZV<537>, ZPZV<2»; };  // NOLINT
template<> struct ConwayPolynomial<541, 3> { using ZPZ = aerobus::zpz<541>; using type = POLYV<ZPZV<1>, ZPZV<0>, ZPZV<2>, ZPZV<539»; };  // NOLINT
template<> struct ConwayPolynomial<541, 4> { using ZPZ = aerobus::zpz<541>; using type = POLYV<ZPZV<1>, ZPZV<0>, ZPZV<6>, ZPZV<333>, ZPZV<2»; };  // NOLINT
template<> struct ConwayPolynomial<541, 5> { using ZPZ = aerobus::zpz<541>; using type = POLYV<ZPZV<1>, ZPZV<0>, ZPZV<0>, ZPZV<0>, ZPZV<3>, ZPZV<539»; };  // NOLINT
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
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<539»; };  // NOLINT
template<> struct ConwayPolynomial<541, 8> { using ZPZ = aerobus::zpz<541>; using type = POLYV<ZPZV<1>, ZPZV<0>, ZPZV<0>, ZPZV<0>, ZPZV<3>, ZPZV<376>, ZPZV<108>, ZPZV<113>, ZPZV<2»; };  // NOLINT
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
template<> struct ConwayPolynomial<547, 1> { using ZPZ = aerobus::zpz<547>; using type = POLYV<ZPZV<1>, ZPZV<545»; };  // NOLINT
template<> struct ConwayPolynomial<547, 2> { using ZPZ = aerobus::zpz<547>; using type = POLYV<ZPZV<1>, ZPZV<543>, ZPZV<2»; };  // NOLINT
template<> struct ConwayPolynomial<547, 3> { using ZPZ = aerobus::zpz<547>; using type = POLYV<ZPZV<1>, ZPZV<0>, ZPZV<4>, ZPZV<545»; };  // NOLINT
template<> struct ConwayPolynomial<547, 4> { using ZPZ = aerobus::zpz<547>; using type = POLYV<ZPZV<1>, ZPZV<0>, ZPZV<8>, ZPZV<334>, ZPZV<2»; };  // NOLINT
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
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
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<11>, ZPZV<545»; };  // NOLINT
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»; };  // NOLINT
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<11>, ZPZV<238>, ZPZV<263>, ZPZV<545»; };  // NOLINT
template<> struct ConwayPolynomial<557, 1> { using ZPZ = aerobus::zpz<557>; using type = POLYV<ZPZV<1>, ZPZV<555»; };  // NOLINT
template<> struct ConwayPolynomial<557, 2> { using ZPZ = aerobus::zpz<557>; using type = POLYV<ZPZV<1>, ZPZV<553>, ZPZV<2»; };  // NOLINT
template<> struct ConwayPolynomial<557, 3> { using ZPZ = aerobus::zpz<557>; using type = POLYV<ZPZV<1>, ZPZV<0>, ZPZV<3>, ZPZV<555»; };  // NOLINT
template<> struct ConwayPolynomial<557, 4> { using ZPZ = aerobus::zpz<557>; using type = POLYV<ZPZV<1>, ZPZV<0>, ZPZV<7>, ZPZV<430>, ZPZV<2»; };  // NOLINT
template<> struct ConwayPolynomial<557, 5> { using ZPZ = aerobus::zpz<557>; using type = POLYV<ZPZV<1>, ZPZV<0>, ZPZV<0>, ZPZV<0>, ZPZV<9>, ZPZV<555»; };  // NOLINT
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
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»; };  // NOLINT
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»; };  // NOLINT
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<1>, ZPZV<456>, ZPZV<434>, ZPZV<555»; };  // NOLINT
template<> struct ConwayPolynomial<563, 1> { using ZPZ = aerobus::zpz<563>; using type = POLYV<ZPZV<1>, ZPZV<561»; };  // NOLINT
template<> struct ConwayPolynomial<563, 2> { using ZPZ = aerobus::zpz<563>; using type = POLYV<ZPZV<1>, ZPZV<559>, ZPZV<2»; };  // NOLINT
template<> struct ConwayPolynomial<563, 3> { using ZPZ = aerobus::zpz<563>; using type = POLYV<ZPZV<1>, ZPZV<0>, ZPZV<3>, ZPZV<561»; };  // NOLINT
template<> struct ConwayPolynomial<563, 4> { using ZPZ = aerobus::zpz<563>; using type = POLYV<ZPZV<1>, ZPZV<0>, ZPZV<20>, ZPZV<399>, ZPZV<2»; };  // NOLINT
template<> struct ConwayPolynomial<563, 5> { using ZPZ = aerobus::zpz<563>; using type = POLYV<ZPZV<1>, ZPZV<0>, ZPZV<0>, ZPZV<0>, ZPZV<6>, ZPZV<561»; };  // NOLINT
template<> struct ConwayPolynomial<563, 6> { using ZPZ = aerobus::zpz<563>; using type = POLYV<ZPZV<1>, ZPZV<0>, ZPZV<1>, ZPZV<122>, ZPZV<303>, ZPZV<246>, ZPZV<2»; };  // NOLINT
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
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
template<> struct ConwayPolynomial<563, 9> { using ZPZ = aerobus::zpz<563>; using type =
POLYV<ZPZV<1>, ZPZV<0>, ZPZV<0>, ZPZV<0>, ZPZV<0>, ZPZV<0>, ZPZV<3>, ZPZV<15>, ZPZV<19>, ZPZV<561»; };
// NOLINT
template<> struct ConwayPolynomial<569, 1> { using ZPZ = aerobus::zpz<569>; using type =
POLYV<ZPZV<1>, ZPZV<566»; };  // NOLINT
template<> struct ConwayPolynomial<569, 2> { using ZPZ = aerobus::zpz<569>; using type =
POLYV<ZPZV<1>, ZPZV<568>, ZPZV<3»; };  // NOLINT
template<> struct ConwayPolynomial<569, 3> { using ZPZ = aerobus::zpz<569>; using type =
POLYV<ZPZV<1>, ZPZV<0>, ZPZV<4>, ZPZV<566»; };  // NOLINT
template<> struct ConwayPolynomial<569, 4> { using ZPZ = aerobus::zpz<569>; using type =
POLYV<ZPZV<1>, ZPZV<0>, ZPZV<7>, ZPZV<381>, ZPZV<3»; };  // NOLINT
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
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
template<> struct ConwayPolynomial<569, 7> { using ZPZ = aerobus::zpz<569>; using type =
POLYV<ZPZV<1>, ZPZV<0>, ZPZV<0>, ZPZV<0>, ZPZV<0>, ZPZV<0>, ZPZV<5>, ZPZV<566»; };  // NOLINT
template<> struct ConwayPolynomial<569, 8> { using ZPZ = aerobus::zpz<569>; using type =
POLYV<ZPZV<1>, ZPZV<0>, ZPZV<0>, ZPZV<0>, ZPZV<1>, ZPZV<527>, ZPZV<173>, ZPZV<241>, ZPZV<3»; };  //
NOLINT
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<1>, ZPZV<478>, ZPZV<566>, ZPZV<566»;
};  // NOLINT
template<> struct ConwayPolynomial<571, 1> { using ZPZ = aerobus::zpz<571>; using type =
POLYV<ZPZV<1>, ZPZV<568»; };  // NOLINT
template<> struct ConwayPolynomial<571, 2> { using ZPZ = aerobus::zpz<571>; using type =
POLYV<ZPZV<1>, ZPZV<570>, ZPZV<3»; };  // NOLINT
template<> struct ConwayPolynomial<571, 3> { using ZPZ = aerobus::zpz<571>; using type =
POLYV<ZPZV<1>, ZPZV<0>, ZPZV<8>, ZPZV<568»; };  // NOLINT
template<> struct ConwayPolynomial<571, 4> { using ZPZ = aerobus::zpz<571>; using type =
POLYV<ZPZV<1>, ZPZV<0>, ZPZV<2>, ZPZV<402>, ZPZV<3»; };  // NOLINT
template<> struct ConwayPolynomial<571, 5> { using ZPZ = aerobus::zpz<571>; using type =
POLYV<ZPZV<1>, ZPZV<0>, ZPZV<0>, ZPZV<0>, ZPZV<4>, ZPZV<568»; };  // NOLINT
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
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<7>, ZPZV<568»; };  // NOLINT
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<3»; };  //
NOLINT
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<34>, ZPZV<545>, ZPZV<179>, ZPZV<568»;
};  // NOLINT
template<> struct ConwayPolynomial<577, 1> { using ZPZ = aerobus::zpz<577>; using type =
POLYV<ZPZV<1>, ZPZV<572»; };  // NOLINT
template<> struct ConwayPolynomial<577, 2> { using ZPZ = aerobus::zpz<577>; using type =
POLYV<ZPZV<1>, ZPZV<572>, ZPZV<5»; };  // NOLINT
template<> struct ConwayPolynomial<577, 3> { using ZPZ = aerobus::zpz<577>; using type =
POLYV<ZPZV<1>, ZPZV<0>, ZPZV<2>, ZPZV<572»; };  // NOLINT
template<> struct ConwayPolynomial<577, 4> { using ZPZ = aerobus::zpz<577>; using type =
POLYV<ZPZV<1>, ZPZV<0>, ZPZV<12>, ZPZV<494>, ZPZV<5»; };  // NOLINT
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
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
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<8>, ZPZV<572»; };  // NOLINT
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»; };  //
NOLINT
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>, ZPZV<449>, ZPZV<572»;
};  // NOLINT
template<> struct ConwayPolynomial<587, 1> { using ZPZ = aerobus::zpz<587>; using type =
POLYV<ZPZV<1>, ZPZV<585»; };  // NOLINT
template<> struct ConwayPolynomial<587, 2> { using ZPZ = aerobus::zpz<587>; using type =
POLYV<ZPZV<1>, ZPZV<583>, ZPZV<2»; };  // NOLINT
template<> struct ConwayPolynomial<587, 3> { using ZPZ = aerobus::zpz<587>; using type =
POLYV<ZPZV<1>, ZPZV<0>, ZPZV<3>, ZPZV<585»; };  // NOLINT
template<> struct ConwayPolynomial<587, 4> { using ZPZ = aerobus::zpz<587>; using type =
POLYV<ZPZV<1>, ZPZV<0>, ZPZV<16>, ZPZV<444>, ZPZV<2»; };  // NOLINT
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
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
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
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<44>, ZPZV<91>, ZPZV<2»; };  //
NOLINT
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<0>, ZPZV<333>, ZPZV<55>, ZPZV<585»;
};  // NOLINT
```

```
template<> struct ConwayPolynomial<593, 1> { using ZPZ = aerobus::zpz<593>; using type = POLYV<ZPZV<1>, ZPZV<590»; };  // NOLINT
template<> struct ConwayPolynomial<593, 2> { using ZPZ = aerobus::zpz<593>; using type = POLYV<ZPZV<1>, ZPZV<592>, ZPZV<3»; };  // NOLINT
template<> struct ConwayPolynomial<593, 3> { using ZPZ = aerobus::zpz<593>; using type = POLYV<ZPZV<1>, ZPZV<0>, ZPZV<6>, ZPZV<590»; };  // NOLINT
template<> struct ConwayPolynomial<593, 4> { using ZPZ = aerobus::zpz<593>; using type = POLYV<ZPZV<1>, ZPZV<0>, ZPZV<4>, ZPZV<419>, ZPZV<3»; };  // NOLINT
template<> struct ConwayPolynomial<593, 5> { using ZPZ = aerobus::zpz<593>; using type = POLYV<ZPZV<1>, ZPZV<0>, ZPZV<0>, ZPZV<0>, ZPZV<3>, ZPZV<590»; };  // NOLINT
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<3»; };  // NOLINT
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»; };  // NOLINT
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<3»; };  // NOLINT
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<8>, ZPZV<223>, ZPZV<523>, ZPZV<590»; };  // NOLINT
template<> struct ConwayPolynomial<599, 1> { using ZPZ = aerobus::zpz<599>; using type = POLYV<ZPZV<1>, ZPZV<592»; };  // NOLINT
template<> struct ConwayPolynomial<599, 2> { using ZPZ = aerobus::zpz<599>; using type = POLYV<ZPZV<1>, ZPZV<598>, ZPZV<7»; };  // NOLINT
template<> struct ConwayPolynomial<599, 3> { using ZPZ = aerobus::zpz<599>; using type = POLYV<ZPZV<1>, ZPZV<0>, ZPZV<2>, ZPZV<592»; };  // NOLINT
template<> struct ConwayPolynomial<599, 4> { using ZPZ = aerobus::zpz<599>; using type = POLYV<ZPZV<1>, ZPZV<0>, ZPZV<3>, ZPZV<419>, ZPZV<7»; };  // NOLINT
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
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
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
template<> struct ConwayPolynomial<599, 8> { using ZPZ = aerobus::zpz<599>; using type = POLYV<ZPZV<1>, ZPZV<0>, ZPZV<0>, ZPZV<3>, ZPZV<440>, ZPZV<37>, ZPZV<124>, ZPZV<7»; };  // NOLINT
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<3>, ZPZV<114>, ZPZV<98>, ZPZV<592»; };  // NOLINT
template<> struct ConwayPolynomial<601, 1> { using ZPZ = aerobus::zpz<601>; using type = POLYV<ZPZV<1>, ZPZV<594»; };  // NOLINT
template<> struct ConwayPolynomial<601, 2> { using ZPZ = aerobus::zpz<601>; using type = POLYV<ZPZV<1>, ZPZV<598>, ZPZV<7»; };  // NOLINT
template<> struct ConwayPolynomial<601, 3> { using ZPZ = aerobus::zpz<601>; using type = POLYV<ZPZV<1>, ZPZV<0>, ZPZV<1>, ZPZV<594»; };  // NOLINT
template<> struct ConwayPolynomial<601, 4> { using ZPZ = aerobus::zpz<601>; using type = POLYV<ZPZV<1>, ZPZV<0>, ZPZV<14>, ZPZV<347>, ZPZV<7»; };  // NOLINT
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
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
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<594»; };  // NOLINT
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»; };  // NOLINT
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
template<> struct ConwayPolynomial<607, 1> { using ZPZ = aerobus::zpz<607>; using type = POLYV<ZPZV<1>, ZPZV<604»; };  // NOLINT
template<> struct ConwayPolynomial<607, 2> { using ZPZ = aerobus::zpz<607>; using type = POLYV<ZPZV<1>, ZPZV<606>, ZPZV<3»; };  // NOLINT
template<> struct ConwayPolynomial<607, 3> { using ZPZ = aerobus::zpz<607>; using type = POLYV<ZPZV<1>, ZPZV<0>, ZPZV<5>, ZPZV<604»; };  // NOLINT
template<> struct ConwayPolynomial<607, 4> { using ZPZ = aerobus::zpz<607>; using type = POLYV<ZPZV<1>, ZPZV<0>, ZPZV<8>, ZPZV<449>, ZPZV<3»; };  // NOLINT
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
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
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<9>, ZPZV<604»; };  // NOLINT
template<> struct ConwayPolynomial<607, 8> { using ZPZ = aerobus::zpz<607>; using type = POLYV<ZPZV<1>, ZPZV<0>, ZPZV<0>, ZPZV<4>, ZPZV<468>, ZPZV<35>, ZPZV<449>, ZPZV<3»; };  // NOLINT
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
template<> struct ConwayPolynomial<613, 1> { using ZPZ = aerobus::zpz<613>; using type = POLYV<ZPZV<1>, ZPZV<611»; };  // NOLINT
template<> struct ConwayPolynomial<613, 2> { using ZPZ = aerobus::zpz<613>; using type = POLYV<ZPZV<1>, ZPZV<609>, ZPZV<2»; };  // NOLINT
template<> struct ConwayPolynomial<613, 3> { using ZPZ = aerobus::zpz<613>; using type = POLYV<ZPZV<1>, ZPZV<0>, ZPZV<6>, ZPZV<611»; };  // NOLINT
template<> struct ConwayPolynomial<613, 4> { using ZPZ = aerobus::zpz<613>; using type =
POLYV<ZPZV<1>, ZPZV<0>, ZPZV<12>, ZPZV<333>, ZPZV<2»; };  // NOLINT
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
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
template<> struct ConwayPolynomial<613, 7> { using ZPZ = aerobus::zpz<613>; using type = POLYV<ZPZV<1>, ZPZV<0>, ZPZV<0>, ZPZV<0>, ZPZV<0>, ZPZV<0>, ZPZV<6>, ZPZV<611»; };  // NOLINT
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
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<536>, ZPZV<611»; };  // NOLINT
template<> struct ConwayPolynomial<617, 1> { using ZPZ = aerobus::zpz<617>; using type = POLYV<ZPZV<1>, ZPZV<614»; };  // NOLINT
template<> struct ConwayPolynomial<617, 2> { using ZPZ = aerobus::zpz<617>; using type = POLYV<ZPZV<1>, ZPZV<612>, ZPZV<3»; };  // NOLINT
template<> struct ConwayPolynomial<617, 3> { using ZPZ = aerobus::zpz<617>; using type = POLYV<ZPZV<1>, ZPZV<0>, ZPZV<3>, ZPZV<614»; };  // NOLINT
template<> struct ConwayPolynomial<617, 4> { using ZPZ = aerobus::zpz<617>; using type = POLYV<ZPZV<1>, ZPZV<0>, ZPZV<2>, ZPZV<503>, ZPZV<3»; };  // NOLINT
template<> struct ConwayPolynomial<617, 5> { using ZPZ = aerobus::zpz<617>; using type = POLYV<ZPZV<1>, ZPZV<0>, ZPZV<0>, ZPZV<0>, ZPZV<4>, ZPZV<614»; };  // NOLINT
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
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<7>, ZPZV<614»; };  // NOLINT
template<> struct ConwayPolynomial<617, 8> { using ZPZ = aerobus::zpz<617>; using type = POLYV<ZPZV<1>, ZPZV<0>, ZPZV<0>, ZPZV<0>, ZPZV<0>, ZPZV<519>, ZPZV<501>, ZPZV<155>, ZPZV<3»; };  // NOLINT
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<15>, ZPZV<388>, ZPZV<543>, ZPZV<614»; };  // NOLINT
template<> struct ConwayPolynomial<619, 1> { using ZPZ = aerobus::zpz<619>; using type = POLYV<ZPZV<1>, ZPZV<617»; };  // NOLINT
template<> struct ConwayPolynomial<619, 2> { using ZPZ = aerobus::zpz<619>; using type = POLYV<ZPZV<1>, ZPZV<618>, ZPZV<2»; };  // NOLINT
template<> struct ConwayPolynomial<619, 3> { using ZPZ = aerobus::zpz<619>; using type = POLYV<ZPZV<1>, ZPZV<0>, ZPZV<6>, ZPZV<617»; };  // NOLINT
template<> struct ConwayPolynomial<619, 4> { using ZPZ = aerobus::zpz<619>; using type = POLYV<ZPZV<1>, ZPZV<0>, ZPZV<6>, ZPZV<492>, ZPZV<2»; };  // NOLINT
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
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
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<617»; };  // NOLINT
template<> struct ConwayPolynomial<619, 8> { using ZPZ = aerobus::zpz<619>; using type = POLYV<ZPZV<1>, ZPZV<0>, ZPZV<0>, ZPZV<0>, ZPZV<10>, ZPZV<416>, ZPZV<383>, ZPZV<225>, ZPZV<2»; };  // NOLINT
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<579>, ZPZV<310>, ZPZV<617»; };  // NOLINT
template<> struct ConwayPolynomial<631, 1> { using ZPZ = aerobus::zpz<631>; using type = POLYV<ZPZV<1>, ZPZV<628»; };  // NOLINT
template<> struct ConwayPolynomial<631, 2> { using ZPZ = aerobus::zpz<631>; using type = POLYV<ZPZV<1>, ZPZV<629>, ZPZV<3»; };  // NOLINT
template<> struct ConwayPolynomial<631, 3> { using ZPZ = aerobus::zpz<631>; using type = POLYV<ZPZV<1>, ZPZV<0>, ZPZV<5>, ZPZV<628»; };  // NOLINT
template<> struct ConwayPolynomial<631, 4> { using ZPZ = aerobus::zpz<631>; using type = POLYV<ZPZV<1>, ZPZV<0>, ZPZV<6>, ZPZV<376>, ZPZV<3»; };  // NOLINT
template<> struct ConwayPolynomial<631, 5> { using ZPZ = aerobus::zpz<631>; using type = POLYV<ZPZV<1>, ZPZV<0>, ZPZV<0>, ZPZV<0>, ZPZV<5>, ZPZV<628»; };  // NOLINT
template<> struct ConwayPolynomial<631, 6> { using ZPZ = aerobus::zpz<631>; using type = POLYV<ZPZV<1>, ZPZV<0>, ZPZV<0>, ZPZV<516>, ZPZV<541>, ZPZV<106>, ZPZV<3»; };  // NOLINT
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
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»; };  // NOLINT
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
template<> struct ConwayPolynomial<641, 1> { using ZPZ = aerobus::zpz<641>; using type = POLYV<ZPZV<1>, ZPZV<638»; };  // NOLINT
template<> struct ConwayPolynomial<641, 2> { using ZPZ = aerobus::zpz<641>; using type = POLYV<ZPZV<1>, ZPZV<635>, ZPZV<3»; };  // NOLINT
template<> struct ConwayPolynomial<641, 3> { using ZPZ = aerobus::zpz<641>; using type = POLYV<ZPZV<1>, ZPZV<0>, ZPZV<4>, ZPZV<638»; };  // NOLINT
template<> struct ConwayPolynomial<641, 4> { using ZPZ = aerobus::zpz<641>; using type = POLYV<ZPZV<1>, ZPZV<0>, ZPZV<0>, ZPZV<629>, ZPZV<3»; };  // NOLINT
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
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
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<3>, ZPZV<638»; };  // NOLINT
```

```
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<3»; };  // NOLINT
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
template<> struct ConwayPolynomial<643, 1> { using ZPZ = aerobus::zpz<643>; using type = POLYV<ZPZV<1>, ZPZV<632»; };  // NOLINT
template<> struct ConwayPolynomial<643, 2> { using ZPZ = aerobus::zpz<643>; using type = POLYV<ZPZV<1>, ZPZV<641>, ZPZV<11»; };  // NOLINT
template<> struct ConwayPolynomial<643, 3> { using ZPZ = aerobus::zpz<643>; using type = POLYV<ZPZV<1>, ZPZV<0>, ZPZV<1>, ZPZV<632»; };  // NOLINT
template<> struct ConwayPolynomial<643, 4> { using ZPZ = aerobus::zpz<643>; using type = POLYV<ZPZV<1>, ZPZV<0>, ZPZV<0>, ZPZV<600>, ZPZV<11»; };  // NOLINT
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
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
template<> struct ConwayPolynomial<643, 7> { using ZPZ = aerobus::zpz<643>; using type = POLYV<ZPZV<1>, ZPZV<0>, ZPZV<0>, ZPZV<0>, ZPZV<0>, ZPZV<0>, ZPZV<5>, ZPZV<632»; };  // NOLINT
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
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
template<> struct ConwayPolynomial<647, 1> { using ZPZ = aerobus::zpz<647>; using type = POLYV<ZPZV<1>, ZPZV<642»; };  // NOLINT
template<> struct ConwayPolynomial<647, 2> { using ZPZ = aerobus::zpz<647>; using type = POLYV<ZPZV<1>, ZPZV<645>, ZPZV<5»; };  // NOLINT
template<> struct ConwayPolynomial<647, 3> { using ZPZ = aerobus::zpz<647>; using type = POLYV<ZPZV<1>, ZPZV<0>, ZPZV<6>, ZPZV<642»; };  // NOLINT
template<> struct ConwayPolynomial<647, 4> { using ZPZ = aerobus::zpz<647>; using type = POLYV<ZPZV<1>, ZPZV<0>, ZPZV<3>, ZPZV<643>, ZPZV<5»; };  // NOLINT
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
template<> struct ConwayPolynomial<647, 6> { using ZPZ = aerobus::zpz<647>; using type = POLYV<ZPZV<1>, ZPZV<0>, ZPZV<5>, ZPZV<308>, ZPZV<385>, ZPZV<642>, ZPZV<5»; };  // NOLINT
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»; };  // NOLINT
template<> struct ConwayPolynomial<647, 8> { using ZPZ = aerobus::zpz<647>; using type = POLYV<ZPZV<1>, ZPZV<0>, ZPZV<0>, ZPZV<0>, ZPZV<9>, ZPZV<603>, ZPZV<259>, ZPZV<271>, ZPZV<5»; };  // NOLINT
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
template<> struct ConwayPolynomial<653, 1> { using ZPZ = aerobus::zpz<653>; using type = POLYV<ZPZV<1>, ZPZV<651»; };  // NOLINT
template<> struct ConwayPolynomial<653, 2> { using ZPZ = aerobus::zpz<653>; using type = POLYV<ZPZV<1>, ZPZV<649>, ZPZV<2»; };  // NOLINT
template<> struct ConwayPolynomial<653, 3> { using ZPZ = aerobus::zpz<653>; using type = POLYV<ZPZV<1>, ZPZV<0>, ZPZV<3>, ZPZV<651»; };  // NOLINT
template<> struct ConwayPolynomial<653, 4> { using ZPZ = aerobus::zpz<653>; using type = POLYV<ZPZV<1>, ZPZV<0>, ZPZV<6>, ZPZV<596>, ZPZV<2»; };  // NOLINT
template<> struct ConwayPolynomial<653, 5> { using ZPZ = aerobus::zpz<653>; using type = POLYV<ZPZV<1>, ZPZV<0>, ZPZV<0>, ZPZV<0>, ZPZV<5>, ZPZV<651»; };  // NOLINT
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<2»; };  // NOLINT
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<15>, ZPZV<651»; };  // NOLINT
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
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<365>, ZPZV<60>, ZPZV<651»; };  // NOLINT
template<> struct ConwayPolynomial<659, 1> { using ZPZ = aerobus::zpz<659>; using type = POLYV<ZPZV<1>, ZPZV<657»; };  // NOLINT
template<> struct ConwayPolynomial<659, 2> { using ZPZ = aerobus::zpz<659>; using type = POLYV<ZPZV<1>, ZPZV<655>, ZPZV<2»; };  // NOLINT
template<> struct ConwayPolynomial<659, 3> { using ZPZ = aerobus::zpz<659>; using type = POLYV<ZPZV<1>, ZPZV<0>, ZPZV<2>, ZPZV<657»; };  // NOLINT
template<> struct ConwayPolynomial<659, 4> { using ZPZ = aerobus::zpz<659>; using type = POLYV<ZPZV<1>, ZPZV<0>, ZPZV<8>, ZPZV<351>, ZPZV<2»; };  // NOLINT
template<> struct ConwayPolynomial<659, 5> { using ZPZ = aerobus::zpz<659>; using type = POLYV<ZPZV<1>, ZPZV<0>, ZPZV<0>, ZPZV<0>, ZPZV<4>, ZPZV<657»; };  // NOLINT
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
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<5>, ZPZV<657»; };  // NOLINT
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
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
template<> struct ConwayPolynomial<661, 1> { using ZPZ = aerobus::zpz<661>; using type =
POLYV<ZPZV<1>, ZPZV<659»; };  // NOLINT
template<> struct ConwayPolynomial<661, 2> { using ZPZ = aerobus::zpz<661>; using type =
POLYV<ZPZV<1>, ZPZV<660>, ZPZV<2»; };  // NOLINT
template<> struct ConwayPolynomial<661, 3> { using ZPZ = aerobus::zpz<661>; using type =
POLYV<ZPZV<1>, ZPZV<0>, ZPZV<4>, ZPZV<659»; };  // NOLINT
template<> struct ConwayPolynomial<661, 4> { using ZPZ = aerobus::zpz<661>; using type =
POLYV<ZPZV<1>, ZPZV<0>, ZPZV<7>, ZPZV<616>, ZPZV<2»; };  // NOLINT
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
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
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<2>, ZPZV<659»; };  // NOLINT
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<2»; };  //
NOLINT
template<> struct ConwayPolynomial<661, 9> { using ZPZ = aerobus::zpz<661>; using type =
POLYV<ZPZV<1>, ZPZV<0>, ZPZV<0>, ZPZV<0>, ZPZV<0>, ZPZV<0>, ZPZV<18>, ZPZV<389>, ZPZV<220>, ZPZV<659»;
};  // NOLINT
template<> struct ConwayPolynomial<673, 1> { using ZPZ = aerobus::zpz<673>; using type =
POLYV<ZPZV<1>, ZPZV<668»; };  // NOLINT
template<> struct ConwayPolynomial<673, 2> { using ZPZ = aerobus::zpz<673>; using type =
POLYV<ZPZV<1>, ZPZV<672>, ZPZV<5»; };  // NOLINT
template<> struct ConwayPolynomial<673, 3> { using ZPZ = aerobus::zpz<673>; using type =
POLYV<ZPZV<1>, ZPZV<0>, ZPZV<1>, ZPZV<668»; };  // NOLINT
template<> struct ConwayPolynomial<673, 4> { using ZPZ = aerobus::zpz<673>; using type =
POLYV<ZPZV<1>, ZPZV<0>, ZPZV<6>, ZPZV<416>, ZPZV<5»; };  // NOLINT
template<> struct ConwayPolynomial<673, 5> { using ZPZ = aerobus::zpz<673>; using type =
POLYV<ZPZV<1>, ZPZV<0>, ZPZV<0>, ZPZV<0>, ZPZV<15>, ZPZV<668»; };  // NOLINT
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
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<668»; };  // NOLINT
template<> struct ConwayPolynomial<673, 8> { using ZPZ = aerobus::zpz<673>; using type =
POLYV<ZPZV<1>, ZPZV<0>, ZPZV<0>, ZPZV<0>, ZPZV<6>, ZPZV<669>, ZPZV<587>, ZPZV<302>, ZPZV<5»; };  //
NOLINT
template<> struct ConwayPolynomial<673, 9> { using ZPZ = aerobus::zpz<673>; using type =
POLYV<ZPZV<1>, ZPZV<0>, ZPZV<0>, ZPZV<0>, ZPZV<0>, ZPZV<0>, ZPZV<11>, ZPZV<347>, ZPZV<553>, ZPZV<668»;
};  // NOLINT
template<> struct ConwayPolynomial<677, 1> { using ZPZ = aerobus::zpz<677>; using type =
POLYV<ZPZV<1>, ZPZV<675»; };  // NOLINT
template<> struct ConwayPolynomial<677, 2> { using ZPZ = aerobus::zpz<677>; using type =
POLYV<ZPZV<1>, ZPZV<672>, ZPZV<2»; };  // NOLINT
template<> struct ConwayPolynomial<677, 3> { using ZPZ = aerobus::zpz<677>; using type =
POLYV<ZPZV<1>, ZPZV<0>, ZPZV<2>, ZPZV<675»; };  // NOLINT
template<> struct ConwayPolynomial<677, 4> { using ZPZ = aerobus::zpz<677>; using type =
POLYV<ZPZV<1>, ZPZV<0>, ZPZV<0>, ZPZV<631>, ZPZV<2»; };  // NOLINT
template<> struct ConwayPolynomial<677, 5> { using ZPZ = aerobus::zpz<677>; using type =
POLYV<ZPZV<1>, ZPZV<0>, ZPZV<0>, ZPZV<0>, ZPZV<5>, ZPZV<675»; };  // NOLINT
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
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
template<> struct ConwayPolynomial<677, 8> { using ZPZ = aerobus::zpz<677>; using type =
POLYV<ZPZV<1>, ZPZV<0>, ZPZV<0>, ZPZV<0>, ZPZV<0>, ZPZV<363>, ZPZV<619>, ZPZV<152>, ZPZV<2»; };  //
NOLINT
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<504>, ZPZV<404>, ZPZV<675»;
};  // NOLINT
template<> struct ConwayPolynomial<683, 1> { using ZPZ = aerobus::zpz<683>; using type =
POLYV<ZPZV<1>, ZPZV<678»; };  // NOLINT
template<> struct ConwayPolynomial<683, 2> { using ZPZ = aerobus::zpz<683>; using type =
POLYV<ZPZV<1>, ZPZV<682>, ZPZV<5»; };  // NOLINT
template<> struct ConwayPolynomial<683, 3> { using ZPZ = aerobus::zpz<683>; using type =
POLYV<ZPZV<1>, ZPZV<0>, ZPZV<5>, ZPZV<678»; };  // NOLINT
template<> struct ConwayPolynomial<683, 4> { using ZPZ = aerobus::zpz<683>; using type =
POLYV<ZPZV<1>, ZPZV<0>, ZPZV<5>, ZPZV<455>, ZPZV<5»; };  // NOLINT
template<> struct ConwayPolynomial<683, 5> { using ZPZ = aerobus::zpz<683>; using type =
POLYV<ZPZV<1>, ZPZV<0>, ZPZV<0>, ZPZV<0>, ZPZV<7>, ZPZV<678»; };  // NOLINT
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
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
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<5»; };  //
NOLINT
template<> struct ConwayPolynomial<683, 9> { using ZPZ = aerobus::zpz<683>; using type =
POLYV<ZPZV<1>, ZPZV<0>, ZPZV<0>, ZPZV<0>, ZPZV<0>, ZPZV<0>, ZPZV<0>, ZPZV<85>, ZPZV<444>, ZPZV<678»;
};  // NOLINT
template<> struct ConwayPolynomial<691, 1> { using ZPZ = aerobus::zpz<691>; using type =
POLYV<ZPZV<1>, ZPZV<688»; };  // NOLINT
template<> struct ConwayPolynomial<691, 2> { using ZPZ = aerobus::zpz<691>; using type =
POLYV<ZPZV<1>, ZPZV<686>, ZPZV<3»; };  // NOLINT
template<> struct ConwayPolynomial<691, 3> { using ZPZ = aerobus::zpz<691>; using type =
POLYV<ZPZV<1>, ZPZV<0>, ZPZV<14>, ZPZV<688»; };  // NOLINT
template<> struct ConwayPolynomial<691, 4> { using ZPZ = aerobus::zpz<691>; using type =
POLYV<ZPZV<1>, ZPZV<0>, ZPZV<3>, ZPZV<632>, ZPZV<3»; };  // NOLINT
```

```
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
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
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
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
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<443>, ZPZV<688»; };  // NOLINT
template<> struct ConwayPolynomial<701, 1> { using ZPZ = aerobus::zpz<701>; using type = POLYV<ZPZV<1>, ZPZV<699»; };  // NOLINT
template<> struct ConwayPolynomial<701, 2> { using ZPZ = aerobus::zpz<701>; using type = POLYV<ZPZV<1>, ZPZV<697>, ZPZV<2»; };  // NOLINT
template<> struct ConwayPolynomial<701, 3> { using ZPZ = aerobus::zpz<701>; using type = POLYV<ZPZV<1>, ZPZV<0>, ZPZV<2>, ZPZV<699»; };  // NOLINT
template<> struct ConwayPolynomial<701, 4> { using ZPZ = aerobus::zpz<701>; using type = POLYV<ZPZV<1>, ZPZV<0>, ZPZV<12>, ZPZV<379>, ZPZV<2»; };  // NOLINT
template<> struct ConwayPolynomial<701, 5> { using ZPZ = aerobus::zpz<701>; using type = POLYV<ZPZV<1>, ZPZV<0>, ZPZV<0>, ZPZV<0>, ZPZV<5>, ZPZV<699»; };  // NOLINT
template<> struct ConwayPolynomial<701, 6> { using ZPZ = aerobus::zpz<701>; using type = POLYV<ZPZV<1>, ZPZV<0>, ZPZV<1>, ZPZV<571>, ZPZV<327>, ZPZV<285>, ZPZV<2»; };  // NOLINT
template<> struct ConwayPolynomial<701, 7> { using ZPZ = aerobus::zpz<701>; using type = POLYV<ZPZV<1>, ZPZV<0>, ZPZV<0>, ZPZV<0>, ZPZV<0>, ZPZV<0>, ZPZV<10>, ZPZV<699»; };  // NOLINT
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
template<> struct ConwayPolynomial<701, 9> { using ZPZ = aerobus::zpz<701>; using type = POLYV<ZPZV<1>, ZPZV<0>, ZPZV<0>, ZPZV<0>, ZPZV<0>, ZPZV<1>, ZPZV<459>, ZPZV<373>, ZPZV<699»; };  // NOLINT
template<> struct ConwayPolynomial<709, 1> { using ZPZ = aerobus::zpz<709>; using type = POLYV<ZPZV<1>, ZPZV<707»; };  // NOLINT
template<> struct ConwayPolynomial<709, 2> { using ZPZ = aerobus::zpz<709>; using type = POLYV<ZPZV<1>, ZPZV<705>, ZPZV<2»; };  // NOLINT
template<> struct ConwayPolynomial<709, 3> { using ZPZ = aerobus::zpz<709>; using type = POLYV<ZPZV<1>, ZPZV<0>, ZPZV<2>, ZPZV<707»; };  // NOLINT
template<> struct ConwayPolynomial<709, 4> { using ZPZ = aerobus::zpz<709>; using type = POLYV<ZPZV<1>, ZPZV<0>, ZPZV<6>, ZPZV<384>, ZPZV<2»; };  // NOLINT
template<> struct ConwayPolynomial<709, 5> { using ZPZ = aerobus::zpz<709>; using type = POLYV<ZPZV<1>, ZPZV<0>, ZPZV<0>, ZPZV<0>, ZPZV<10>, ZPZV<707»; };  // NOLINT
template<> struct ConwayPolynomial<709, 6> { using ZPZ = aerobus::zpz<709>; using type = POLYV<ZPZV<1>, ZPZV<0>, ZPZV<0>, ZPZV<669>, ZPZV<514>, ZPZV<295>, ZPZV<2»; };  // NOLINT
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<4>, ZPZV<707»; };  // NOLINT
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
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<3>, ZPZV<257>, ZPZV<171>, ZPZV<707»; };  // NOLINT
template<> struct ConwayPolynomial<719, 1> { using ZPZ = aerobus::zpz<719>; using type = POLYV<ZPZV<1>, ZPZV<708»; };  // NOLINT
template<> struct ConwayPolynomial<719, 2> { using ZPZ = aerobus::zpz<719>; using type = POLYV<ZPZV<1>, ZPZV<715>, ZPZV<11»; };  // NOLINT
template<> struct ConwayPolynomial<719, 3> { using ZPZ = aerobus::zpz<719>; using type = POLYV<ZPZV<1>, ZPZV<0>, ZPZV<1>, ZPZV<708»; };  // NOLINT
template<> struct ConwayPolynomial<719, 4> { using ZPZ = aerobus::zpz<719>; using type = POLYV<ZPZV<1>, ZPZV<0>, ZPZV<5>, ZPZV<602>, ZPZV<11»; };  // NOLINT
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
template<> struct ConwayPolynomial<719, 6> { using ZPZ = aerobus::zpz<719>; using type = POLYV<ZPZV<1>, ZPZV<0>, ZPZV<1>, ZPZV<533>, ZPZV<591>, ZPZV<182>, ZPZV<11»; };  // NOLINT
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<11>, ZPZV<708»; };  // NOLINT
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»; };  // NOLINT
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<288>, ZPZV<560>, ZPZV<708»; };  // NOLINT
template<> struct ConwayPolynomial<727, 1> { using ZPZ = aerobus::zpz<727>; using type = POLYV<ZPZV<1>, ZPZV<722»; };  // NOLINT
template<> struct ConwayPolynomial<727, 2> { using ZPZ = aerobus::zpz<727>; using type = POLYV<ZPZV<1>, ZPZV<725>, ZPZV<5»; };  // NOLINT
template<> struct ConwayPolynomial<727, 3> { using ZPZ = aerobus::zpz<727>; using type = POLYV<ZPZV<1>, ZPZV<0>, ZPZV<7>, ZPZV<722»; };  // NOLINT
template<> struct ConwayPolynomial<727, 4> { using ZPZ = aerobus::zpz<727>; using type = POLYV<ZPZV<1>, ZPZV<0>, ZPZV<3>, ZPZV<723>, ZPZV<5»; };  // NOLINT
template<> struct ConwayPolynomial<727, 5> { using ZPZ = aerobus::zpz<727>; using type = POLYV<ZPZV<1>, ZPZV<0>, ZPZV<0>, ZPZV<0>, ZPZV<5>, ZPZV<722»; };  // NOLINT
template<> struct ConwayPolynomial<727, 6> { using ZPZ = aerobus::zpz<727>; using type = POLYV<ZPZV<1>, ZPZV<0>, ZPZV<86>, ZPZV<397>, ZPZV<672>, ZPZV<5»; };  // NOLINT
template<> struct ConwayPolynomial<727, 7> { using ZPZ = aerobus::zpz<727>; using type = POLYV<ZPZV<1>, ZPZV<0>, ZPZV<0>, ZPZV<0>, ZPZV<0>, ZPZV<0>, ZPZV<17>, ZPZV<722»; };  // NOLINT
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»; };  //
NOLINT
template<> struct ConwayPolynomial<727, 9> { using ZPZ = aerobus::zpz<727>; using type =
POLYV<ZPZV<1>, ZPZV<0>, ZPZV<0>, ZPZV<0>, ZPZV<0>, ZPZV<6>, ZPZV<573>, ZPZV<502>, ZPZV<722»;
};  // NOLINT
template<> struct ConwayPolynomial<733, 1> { using ZPZ = aerobus::zpz<733>; using type =
POLYV<ZPZV<1>, ZPZV<727»; };  // NOLINT
template<> struct ConwayPolynomial<733, 2> { using ZPZ = aerobus::zpz<733>; using type =
POLYV<ZPZV<1>, ZPZV<732>, ZPZV<6»; };  // NOLINT
template<> struct ConwayPolynomial<733, 3> { using ZPZ = aerobus::zpz<733>; using type =
POLYV<ZPZV<1>, ZPZV<0>, ZPZV<4>, ZPZV<727»; };  // NOLINT
template<> struct ConwayPolynomial<733, 4> { using ZPZ = aerobus::zpz<733>; using type =
POLYV<ZPZV<1>, ZPZV<0>, ZPZV<12>, ZPZV<539>, ZPZV<6»; };  // NOLINT
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
template<> struct ConwayPolynomial<733, 6> { using ZPZ = aerobus::zpz<733>; using type =
POLYV<ZPZV<1>, ZPZV<0>, ZPZV<1>, ZPZV<174>, ZPZV<549>, ZPZV<151>, ZPZV<6»; };  // NOLINT
template<> struct ConwayPolynomial<733, 7> { using ZPZ = aerobus::zpz<733>; using type =
POLYV<ZPZV<1>, ZPZV<0>, ZPZV<0>, ZPZV<0>, ZPZV<0>, ZPZV<0>, ZPZV<3>, ZPZV<727»; };  // NOLINT
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»; };  //
NOLINT
template<> struct ConwayPolynomial<733, 9> { using ZPZ = aerobus::zpz<733>; using type =
POLYV<ZPZV<1>, ZPZV<0>, ZPZV<0>, ZPZV<0>, ZPZV<0>, ZPZV<0>, ZPZV<2>, ZPZV<337>, ZPZV<6>, ZPZV<727»; };
// NOLINT
template<> struct ConwayPolynomial<739, 1> { using ZPZ = aerobus::zpz<739>; using type =
POLYV<ZPZV<1>, ZPZV<736»; };  // NOLINT
template<> struct ConwayPolynomial<739, 2> { using ZPZ = aerobus::zpz<739>; using type =
POLYV<ZPZV<1>, ZPZV<734>, ZPZV<3»; };  // NOLINT
template<> struct ConwayPolynomial<739, 3> { using ZPZ = aerobus::zpz<739>; using type =
POLYV<ZPZV<1>, ZPZV<0>, ZPZV<11>, ZPZV<736»; };  // NOLINT
template<> struct ConwayPolynomial<739, 4> { using ZPZ = aerobus::zpz<739>; using type =
POLYV<ZPZV<1>, ZPZV<0>, ZPZV<3>, ZPZV<678>, ZPZV<3»; };  // NOLINT
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
template<> struct ConwayPolynomial<739, 6> { using ZPZ = aerobus::zpz<739>; using type =
POLYV<ZPZV<1>, ZPZV<0>, ZPZV<0>, ZPZV<422>, ZPZV<447>, ZPZV<625>, ZPZV<3»; };  // NOLINT
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
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
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<736»;
};  // NOLINT
template<> struct ConwayPolynomial<743, 1> { using ZPZ = aerobus::zpz<743>; using type =
POLYV<ZPZV<1>, ZPZV<738»; };  // NOLINT
template<> struct ConwayPolynomial<743, 2> { using ZPZ = aerobus::zpz<743>; using type =
POLYV<ZPZV<1>, ZPZV<742>, ZPZV<5»; };  // NOLINT
template<> struct ConwayPolynomial<743, 3> { using ZPZ = aerobus::zpz<743>; using type =
POLYV<ZPZV<1>, ZPZV<0>, ZPZV<3>, ZPZV<738»; };  // NOLINT
template<> struct ConwayPolynomial<743, 4> { using ZPZ = aerobus::zpz<743>; using type =
POLYV<ZPZV<1>, ZPZV<0>, ZPZV<5>, ZPZV<425>, ZPZV<5»; };  // NOLINT
template<> struct ConwayPolynomial<743, 5> { using ZPZ = aerobus::zpz<743>; using type =
POLYV<ZPZV<1>, ZPZV<0>, ZPZV<0>, ZPZV<0>, ZPZV<6>, ZPZV<738»; };  // NOLINT
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
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<738»; };  // NOLINT
template<> struct ConwayPolynomial<743, 8> { using ZPZ = aerobus::zpz<743>; using type =
POLYV<ZPZV<1>, ZPZV<0>, ZPZV<0>, ZPZV<0>, ZPZV<0>, ZPZV<551>, ZPZV<279>, ZPZV<588>, ZPZV<5»; };  //
NOLINT
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<327>, ZPZV<676>, ZPZV<738»;
};  // NOLINT
template<> struct ConwayPolynomial<751, 1> { using ZPZ = aerobus::zpz<751>; using type =
POLYV<ZPZV<1>, ZPZV<748»; };  // NOLINT
template<> struct ConwayPolynomial<751, 2> { using ZPZ = aerobus::zpz<751>; using type =
POLYV<ZPZV<1>, ZPZV<749>, ZPZV<3»; };  // NOLINT
template<> struct ConwayPolynomial<751, 3> { using ZPZ = aerobus::zpz<751>; using type =
POLYV<ZPZV<1>, ZPZV<0>, ZPZV<5>, ZPZV<748»; };  // NOLINT
template<> struct ConwayPolynomial<751, 4> { using ZPZ = aerobus::zpz<751>; using type =
POLYV<ZPZV<1>, ZPZV<0>, ZPZV<3>, ZPZV<525>, ZPZV<3»; };  // NOLINT
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
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
template<> struct ConwayPolynomial<751, 7> { using ZPZ = aerobus::zpz<751>; using type =
POLYV<ZPZV<1>, ZPZV<0>, ZPZV<0>, ZPZV<0>, ZPZV<0>, ZPZV<0>, ZPZV<7>, ZPZV<748»; };  // NOLINT
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»; };  //
NOLINT
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<10>, ZPZV<703>, ZPZV<489>, ZPZV<748»;
};  // NOLINT
template<> struct ConwayPolynomial<757, 1> { using ZPZ = aerobus::zpz<757>; using type =
POLYV<ZPZV<1>, ZPZV<755»; };  // NOLINT
```

```
template<> struct ConwayPolynomial<757, 2> { using ZPZ = aerobus::zpz<757>; using type = POLYV<ZPZV<1>, ZPZV<753>, ZPZV<2»; };  // NOLINT
template<> struct ConwayPolynomial<757, 3> { using ZPZ = aerobus::zpz<757>; using type = POLYV<ZPZV<1>, ZPZV<0>, ZPZV<6>, ZPZV<755»; };  // NOLINT
template<> struct ConwayPolynomial<757, 4> { using ZPZ = aerobus::zpz<757>; using type = POLYV<ZPZV<1>, ZPZV<0>, ZPZV<10>, ZPZV<537>, ZPZV<2»; };  // NOLINT
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
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
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<4>, ZPZV<755»; };  // NOLINT
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»; };  // NOLINT
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<702>, ZPZV<755»; };  // NOLINT
template<> struct ConwayPolynomial<761, 1> { using ZPZ = aerobus::zpz<761>; using type = POLYV<ZPZV<1>, ZPZV<755»; };  // NOLINT
template<> struct ConwayPolynomial<761, 2> { using ZPZ = aerobus::zpz<761>; using type = POLYV<ZPZV<1>, ZPZV<758>, ZPZV<6»; };  // NOLINT
template<> struct ConwayPolynomial<761, 3> { using ZPZ = aerobus::zpz<761>; using type = POLYV<ZPZV<1>, ZPZV<0>, ZPZV<12>, ZPZV<755»; };  // NOLINT
template<> struct ConwayPolynomial<761, 4> { using ZPZ = aerobus::zpz<761>; using type = POLYV<ZPZV<1>, ZPZV<0>, ZPZV<0>, ZPZV<658>, ZPZV<6»; };  // NOLINT
template<> struct ConwayPolynomial<761, 5> { using ZPZ = aerobus::zpz<761>; using type = POLYV<ZPZV<1>, ZPZV<0>, ZPZV<0>, ZPZV<0>, ZPZV<6>, ZPZV<755»; };  // NOLINT
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
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»; };  // NOLINT
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<6»; };  // NOLINT
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
template<> struct ConwayPolynomial<769, 1> { using ZPZ = aerobus::zpz<769>; using type = POLYV<ZPZV<1>, ZPZV<758»; };  // NOLINT
template<> struct ConwayPolynomial<769, 2> { using ZPZ = aerobus::zpz<769>; using type = POLYV<ZPZV<1>, ZPZV<765>, ZPZV<11»; };  // NOLINT
template<> struct ConwayPolynomial<769, 3> { using ZPZ = aerobus::zpz<769>; using type = POLYV<ZPZV<1>, ZPZV<0>, ZPZV<2>, ZPZV<758»; };  // NOLINT
template<> struct ConwayPolynomial<769, 4> { using ZPZ = aerobus::zpz<769>; using type = POLYV<ZPZV<1>, ZPZV<0>, ZPZV<32>, ZPZV<741>, ZPZV<11»; };  // NOLINT
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
template<> struct ConwayPolynomial<769, 6> { using ZPZ = aerobus::zpz<769>; using type = POLYV<ZPZV<1>, ZPZV<0>, ZPZV<0>, ZPZV<43>, ZPZV<326>, ZPZV<650>, ZPZV<11»; };  // NOLINT
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<8>, ZPZV<758»; };  // NOLINT
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
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<623>, ZPZV<751>, ZPZV<758»; };  // NOLINT
template<> struct ConwayPolynomial<773, 1> { using ZPZ = aerobus::zpz<773>; using type = POLYV<ZPZV<1>, ZPZV<771»; };  // NOLINT
template<> struct ConwayPolynomial<773, 2> { using ZPZ = aerobus::zpz<773>; using type = POLYV<ZPZV<1>, ZPZV<772>, ZPZV<2»; };  // NOLINT
template<> struct ConwayPolynomial<773, 3> { using ZPZ = aerobus::zpz<773>; using type = POLYV<ZPZV<1>, ZPZV<0>, ZPZV<2>, ZPZV<771»; };  // NOLINT
template<> struct ConwayPolynomial<773, 4> { using ZPZ = aerobus::zpz<773>; using type = POLYV<ZPZV<1>, ZPZV<0>, ZPZV<4>, ZPZV<444>, ZPZV<2»; };  // NOLINT
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
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
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<9>, ZPZV<771»; };  // NOLINT
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
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
template<> struct ConwayPolynomial<787, 1> { using ZPZ = aerobus::zpz<787>; using type = POLYV<ZPZV<1>, ZPZV<785»; };  // NOLINT
template<> struct ConwayPolynomial<787, 2> { using ZPZ = aerobus::zpz<787>; using type = POLYV<ZPZV<1>, ZPZV<786>, ZPZV<2»; };  // NOLINT
template<> struct ConwayPolynomial<787, 3> { using ZPZ = aerobus::zpz<787>; using type = POLYV<ZPZV<1>, ZPZV<0>, ZPZV<2>, ZPZV<785»; };  // NOLINT
template<> struct ConwayPolynomial<787, 4> { using ZPZ = aerobus::zpz<787>; using type = POLYV<ZPZV<1>, ZPZV<0>, ZPZV<11>, ZPZV<605>, ZPZV<2»; };  // NOLINT
template<> struct ConwayPolynomial<787, 5> { using ZPZ = aerobus::zpz<787>; using type =
POLYV<ZPZV<1>, ZPZV<0>, ZPZV<0>, ZPZV<0>, ZPZV<9>, ZPZV<785»; };  // NOLINT
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
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<3>, ZPZV<785»; };  // NOLINT
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<715>, ZPZV<2»; };  //
NOLINT
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
template<> struct ConwayPolynomial<797, 1> { using ZPZ = aerobus::zpz<797>; using type =
POLYV<ZPZV<1>, ZPZV<795»; };  // NOLINT
template<> struct ConwayPolynomial<797, 2> { using ZPZ = aerobus::zpz<797>; using type =
POLYV<ZPZV<1>, ZPZV<793>, ZPZV<2»; };  // NOLINT
template<> struct ConwayPolynomial<797, 3> { using ZPZ = aerobus::zpz<797>; using type =
POLYV<ZPZV<1>, ZPZV<0>, ZPZV<2>, ZPZV<795»; };  // NOLINT
template<> struct ConwayPolynomial<797, 4> { using ZPZ = aerobus::zpz<797>; using type =
POLYV<ZPZV<1>, ZPZV<0>, ZPZV<1>, ZPZV<717>, ZPZV<2»; };  // NOLINT
template<> struct ConwayPolynomial<797, 5> { using ZPZ = aerobus::zpz<797>; using type =
POLYV<ZPZV<1>, ZPZV<0>, ZPZV<0>, ZPZV<0>, ZPZV<3>, ZPZV<795»; };  // NOLINT
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
template<> struct ConwayPolynomial<797, 7> { using ZPZ = aerobus::zpz<797>; using type =
POLYV<ZPZV<1>, ZPZV<0>, ZPZV<0>, ZPZV<0>, ZPZV<0>, ZPZV<0>, ZPZV<11>, ZPZV<795»; };  // NOLINT
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
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
template<> struct ConwayPolynomial<809, 1> { using ZPZ = aerobus::zpz<809>; using type =
POLYV<ZPZV<1>, ZPZV<806»; };  // NOLINT
template<> struct ConwayPolynomial<809, 2> { using ZPZ = aerobus::zpz<809>; using type =
POLYV<ZPZV<1>, ZPZV<799>, ZPZV<3»; };  // NOLINT
template<> struct ConwayPolynomial<809, 3> { using ZPZ = aerobus::zpz<809>; using type =
POLYV<ZPZV<1>, ZPZV<0>, ZPZV<1>, ZPZV<806»; };  // NOLINT
template<> struct ConwayPolynomial<809, 4> { using ZPZ = aerobus::zpz<809>; using type =
POLYV<ZPZV<1>, ZPZV<0>, ZPZV<4>, ZPZV<644>, ZPZV<3»; };  // NOLINT
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
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
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<3>, ZPZV<806»; };  // NOLINT
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
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
template<> struct ConwayPolynomial<811, 1> { using ZPZ = aerobus::zpz<811>; using type =
POLYV<ZPZV<1>, ZPZV<808»; };  // NOLINT
template<> struct ConwayPolynomial<811, 2> { using ZPZ = aerobus::zpz<811>; using type =
POLYV<ZPZV<1>, ZPZV<806>, ZPZV<3»; };  // NOLINT
template<> struct ConwayPolynomial<811, 3> { using ZPZ = aerobus::zpz<811>; using type =
POLYV<ZPZV<1>, ZPZV<0>, ZPZV<1>, ZPZV<808»; };  // NOLINT
template<> struct ConwayPolynomial<811, 4> { using ZPZ = aerobus::zpz<811>; using type =
POLYV<ZPZV<1>, ZPZV<0>, ZPZV<3>, ZPZV<453>, ZPZV<3»; };  // NOLINT
template<> struct ConwayPolynomial<811, 5> { using ZPZ = aerobus::zpz<811>; using type =
POLYV<ZPZV<1>, ZPZV<0>, ZPZV<0>, ZPZV<0>, ZPZV<3>, ZPZV<808»; };  // NOLINT
template<> struct ConwayPolynomial<811, 6> { using ZPZ = aerobus::zpz<811>; using type =
POLYV<ZPZV<1>, ZPZV<0>, ZPZV<0>, ZPZV<780>, ZPZV<755>, ZPZV<307>, ZPZV<3»; };  // NOLINT
template<> struct ConwayPolynomial<811, 7> { using ZPZ = aerobus::zpz<811>; using type =
POLYV<ZPZV<1>, ZPZV<0>, ZPZV<0>, ZPZV<0>, ZPZV<0>, ZPZV<0>, ZPZV<1>, ZPZV<808»; };  // NOLINT
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<525>, ZPZV<3»; };  //
NOLINT
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
template<> struct ConwayPolynomial<821, 1> { using ZPZ = aerobus::zpz<821>; using type =
POLYV<ZPZV<1>, ZPZV<819»; };  // NOLINT
template<> struct ConwayPolynomial<821, 2> { using ZPZ = aerobus::zpz<821>; using type =
POLYV<ZPZV<1>, ZPZV<816>, ZPZV<2»; };  // NOLINT
template<> struct ConwayPolynomial<821, 3> { using ZPZ = aerobus::zpz<821>; using type =
POLYV<ZPZV<1>, ZPZV<0>, ZPZV<2>, ZPZV<819»; };  // NOLINT
template<> struct ConwayPolynomial<821, 4> { using ZPZ = aerobus::zpz<821>; using type =
POLYV<ZPZV<1>, ZPZV<0>, ZPZV<15>, ZPZV<662>, ZPZV<2»; };  // NOLINT
template<> struct ConwayPolynomial<821, 5> { using ZPZ = aerobus::zpz<821>; using type =
POLYV<ZPZV<1>, ZPZV<0>, ZPZV<0>, ZPZV<0>, ZPZV<9>, ZPZV<819»; };  // NOLINT
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
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<10>, ZPZV<819»; };  // NOLINT
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<2»; };  //
```

```
NOLINT
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
template<> struct ConwayPolynomial<823, 1> { using ZPZ = aerobus::zpz<823>; using type = POLYV<ZPZV<1>, ZPZV<820»; };  // NOLINT
template<> struct ConwayPolynomial<823, 2> { using ZPZ = aerobus::zpz<823>; using type = POLYV<ZPZV<1>, ZPZV<821>, ZPZV<3»; };  // NOLINT
template<> struct ConwayPolynomial<823, 3> { using ZPZ = aerobus::zpz<823>; using type = POLYV<ZPZV<1>, ZPZV<0>, ZPZV<3>, ZPZV<820»; };  // NOLINT
template<> struct ConwayPolynomial<823, 4> { using ZPZ = aerobus::zpz<823>; using type = POLYV<ZPZV<1>, ZPZV<0>, ZPZV<4>, ZPZV<819>, ZPZV<3»; };  // NOLINT
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
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
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
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
template<> struct ConwayPolynomial<823, 9> { using ZPZ = aerobus::zpz<823>; using type = POLYV<ZPZV<1>, ZPZV<0>, ZPZV<0>, ZPZV<0>, ZPZV<0>, ZPZV<0>, ZPZV<6>, ZPZV<740>, ZPZV<609>, ZPZV<820»; };  // NOLINT
template<> struct ConwayPolynomial<827, 1> { using ZPZ = aerobus::zpz<827>; using type = POLYV<ZPZV<1>, ZPZV<825»; };  // NOLINT
template<> struct ConwayPolynomial<827, 2> { using ZPZ = aerobus::zpz<827>; using type = POLYV<ZPZV<1>, ZPZV<821>, ZPZV<2»; };  // NOLINT
template<> struct ConwayPolynomial<827, 3> { using ZPZ = aerobus::zpz<827>; using type = POLYV<ZPZV<1>, ZPZV<0>, ZPZV<5>, ZPZV<825»; };  // NOLINT
template<> struct ConwayPolynomial<827, 4> { using ZPZ = aerobus::zpz<827>; using type = POLYV<ZPZV<1>, ZPZV<0>, ZPZV<18>, ZPZV<605>, ZPZV<2»; };  // NOLINT
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
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
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<825»; };  // NOLINT
template<> struct ConwayPolynomial<827, 8> { using ZPZ = aerobus::zpz<827>; using type = POLYV<ZPZV<1>, ZPZV<0>, ZPZV<0>, ZPZV<2>, ZPZV<812>, ZPZV<79>, ZPZV<32>, ZPZV<2»; };  // NOLINT
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<177>, ZPZV<372>, ZPZV<825»; };  // NOLINT
template<> struct ConwayPolynomial<829, 1> { using ZPZ = aerobus::zpz<829>; using type = POLYV<ZPZV<1>, ZPZV<827»; };  // NOLINT
template<> struct ConwayPolynomial<829, 2> { using ZPZ = aerobus::zpz<829>; using type = POLYV<ZPZV<1>, ZPZV<828>, ZPZV<2»; };  // NOLINT
template<> struct ConwayPolynomial<829, 3> { using ZPZ = aerobus::zpz<829>; using type = POLYV<ZPZV<1>, ZPZV<0>, ZPZV<5>, ZPZV<827»; };  // NOLINT
template<> struct ConwayPolynomial<829, 4> { using ZPZ = aerobus::zpz<829>; using type = POLYV<ZPZV<1>, ZPZV<0>, ZPZV<9>, ZPZV<604>, ZPZV<2»; };  // NOLINT
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
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
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<827»; };  // NOLINT
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
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<3>, ZPZV<621>, ZPZV<552>, ZPZV<827»; };  // NOLINT
template<> struct ConwayPolynomial<839, 1> { using ZPZ = aerobus::zpz<839>; using type = POLYV<ZPZV<1>, ZPZV<828»; };  // NOLINT
template<> struct ConwayPolynomial<839, 2> { using ZPZ = aerobus::zpz<839>; using type = POLYV<ZPZV<1>, ZPZV<838>, ZPZV<11»; };  // NOLINT
template<> struct ConwayPolynomial<839, 3> { using ZPZ = aerobus::zpz<839>; using type = POLYV<ZPZV<1>, ZPZV<0>, ZPZV<4>, ZPZV<828»; };  // NOLINT
template<> struct ConwayPolynomial<839, 4> { using ZPZ = aerobus::zpz<839>; using type = POLYV<ZPZV<1>, ZPZV<0>, ZPZV<2>, ZPZV<609>, ZPZV<11»; };  // NOLINT
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
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
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<7>, ZPZV<828»; };  // NOLINT
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»; };  // NOLINT
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<349>, ZPZV<206>, ZPZV<828»; };  // NOLINT
template<> struct ConwayPolynomial<853, 1> { using ZPZ = aerobus::zpz<853>; using type = POLYV<ZPZV<1>, ZPZV<851»; };  // NOLINT
template<> struct ConwayPolynomial<853, 2> { using ZPZ = aerobus::zpz<853>; using type =
POLYV<ZPZV<1>, ZPZV<852>, ZPZV<2»; };  // NOLINT
template<> struct ConwayPolynomial<853, 3> { using ZPZ = aerobus::zpz<853>; using type = POLYV<ZPZV<1>, ZPZV<0>, ZPZV<4>, ZPZV<851»; };  // NOLINT
template<> struct ConwayPolynomial<853, 4> { using ZPZ = aerobus::zpz<853>; using type = POLYV<ZPZV<1>, ZPZV<0>, ZPZV<3>, ZPZV<623>, ZPZV<2»; };  // NOLINT
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
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
template<> struct ConwayPolynomial<853, 7> { using ZPZ = aerobus::zpz<853>; using type = POLYV<ZPZV<1>, ZPZV<0>, ZPZV<0>, ZPZV<0>, ZPZV<0>, ZPZV<0>, ZPZV<4>, ZPZV<851»; };  // NOLINT
template<> struct ConwayPolynomial<853, 8> { using ZPZ = aerobus::zpz<853>; using type = POLYV<ZPZV<1>, ZPZV<0>, ZPZV<0>, ZPZV<0>, ZPZV<8>, ZPZV<544>, ZPZV<846>, ZPZV<118>, ZPZV<2»; };  // NOLINT
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<11>, ZPZV<677>, ZPZV<821>, ZPZV<851»; };  // NOLINT
template<> struct ConwayPolynomial<857, 1> { using ZPZ = aerobus::zpz<857>; using type = POLYV<ZPZV<1>, ZPZV<854»; };  // NOLINT
template<> struct ConwayPolynomial<857, 2> { using ZPZ = aerobus::zpz<857>; using type = POLYV<ZPZV<1>, ZPZV<850>, ZPZV<3»; };  // NOLINT
template<> struct ConwayPolynomial<857, 3> { using ZPZ = aerobus::zpz<857>; using type = POLYV<ZPZV<1>, ZPZV<0>, ZPZV<4>, ZPZV<854»; };  // NOLINT
template<> struct ConwayPolynomial<857, 4> { using ZPZ = aerobus::zpz<857>; using type = POLYV<ZPZV<1>, ZPZV<0>, ZPZV<0>, ZPZV<528>, ZPZV<3»; };  // NOLINT
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
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<3»; };  // NOLINT
template<> struct ConwayPolynomial<857, 7> { using ZPZ = aerobus::zpz<857>; using type = POLYV<ZPZV<1>, ZPZV<0>, ZPZV<0>, ZPZV<0>, ZPZV<0>, ZPZV<0>, ZPZV<7>, ZPZV<854»; };  // NOLINT
template<> struct ConwayPolynomial<857, 8> { using ZPZ = aerobus::zpz<857>; using type = POLYV<ZPZV<1>, ZPZV<0>, ZPZV<0>, ZPZV<0>, ZPZV<3>, ZPZV<611>, ZPZV<552>, ZPZV<494>, ZPZV<3»; };  // NOLINT
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<308>, ZPZV<719>, ZPZV<854»; };  // NOLINT
template<> struct ConwayPolynomial<859, 1> { using ZPZ = aerobus::zpz<859>; using type = POLYV<ZPZV<1>, ZPZV<857»; };  // NOLINT
template<> struct ConwayPolynomial<859, 2> { using ZPZ = aerobus::zpz<859>; using type = POLYV<ZPZV<1>, ZPZV<858>, ZPZV<2»; };  // NOLINT
template<> struct ConwayPolynomial<859, 3> { using ZPZ = aerobus::zpz<859>; using type = POLYV<ZPZV<1>, ZPZV<0>, ZPZV<6>, ZPZV<857»; };  // NOLINT
template<> struct ConwayPolynomial<859, 4> { using ZPZ = aerobus::zpz<859>; using type = POLYV<ZPZV<1>, ZPZV<0>, ZPZV<2>, ZPZV<530>, ZPZV<2»; };  // NOLINT
template<> struct ConwayPolynomial<859, 5> { using ZPZ = aerobus::zpz<859>; using type = POLYV<ZPZV<1>, ZPZV<0>, ZPZV<0>, ZPZV<0>, ZPZV<12>, ZPZV<857»; };  // NOLINT
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
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<2>, ZPZV<857»; };  // NOLINT
template<> struct ConwayPolynomial<859, 8> { using ZPZ = aerobus::zpz<859>; using type = POLYV<ZPZV<1>, ZPZV<0>, ZPZV<0>, ZPZV<0>, ZPZV<0>, ZPZV<522>, ZPZV<446>, ZPZV<672>, ZPZV<2»; };  // NOLINT
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<7>, ZPZV<648>, ZPZV<845>, ZPZV<857»; };  // NOLINT
template<> struct ConwayPolynomial<863, 1> { using ZPZ = aerobus::zpz<863>; using type = POLYV<ZPZV<1>, ZPZV<858»; };  // NOLINT
template<> struct ConwayPolynomial<863, 2> { using ZPZ = aerobus::zpz<863>; using type = POLYV<ZPZV<1>, ZPZV<862>, ZPZV<5»; };  // NOLINT
template<> struct ConwayPolynomial<863, 3> { using ZPZ = aerobus::zpz<863>; using type = POLYV<ZPZV<1>, ZPZV<0>, ZPZV<5>, ZPZV<858»; };  // NOLINT
template<> struct ConwayPolynomial<863, 4> { using ZPZ = aerobus::zpz<863>; using type = POLYV<ZPZV<1>, ZPZV<0>, ZPZV<2>, ZPZV<770>, ZPZV<5»; };  // NOLINT
template<> struct ConwayPolynomial<863, 5> { using ZPZ = aerobus::zpz<863>; using type = POLYV<ZPZV<1>, ZPZV<0>, ZPZV<0>, ZPZV<0>, ZPZV<10>, ZPZV<858»; };  // NOLINT
template<> struct ConwayPolynomial<863, 6> { using ZPZ = aerobus::zpz<863>; using type = POLYV<ZPZV<1>, ZPZV<0>, ZPZV<1>, ZPZV<330>, ZPZV<62>, ZPZV<300>, ZPZV<5»; };  // NOLINT
template<> struct ConwayPolynomial<863, 7> { using ZPZ = aerobus::zpz<863>; using type = POLYV<ZPZV<1>, ZPZV<0>, ZPZV<0>, ZPZV<0>, ZPZV<0>, ZPZV<0>, ZPZV<1>, ZPZV<858»; };  // NOLINT
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<5»; };  // NOLINT
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<1>, ZPZV<858»; }; // NOLINT
template<> struct ConwayPolynomial<877, 1> { using ZPZ = aerobus::zpz<877>; using type = POLYV<ZPZV<1>, ZPZV<875»; };  // NOLINT
template<> struct ConwayPolynomial<877, 2> { using ZPZ = aerobus::zpz<877>; using type = POLYV<ZPZV<1>, ZPZV<873>, ZPZV<2»; };  // NOLINT
template<> struct ConwayPolynomial<877, 3> { using ZPZ = aerobus::zpz<877>; using type = POLYV<ZPZV<1>, ZPZV<0>, ZPZV<5>, ZPZV<875»; };  // NOLINT
template<> struct ConwayPolynomial<877, 4> { using ZPZ = aerobus::zpz<877>; using type = POLYV<ZPZV<1>, ZPZV<0>, ZPZV<6>, ZPZV<604>, ZPZV<2»; };  // NOLINT
template<> struct ConwayPolynomial<877, 5> { using ZPZ = aerobus::zpz<877>; using type = POLYV<ZPZV<1>, ZPZV<0>, ZPZV<0>, ZPZV<0>, ZPZV<6>, ZPZV<875»; };  // NOLINT
```

```
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
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<875>»; };  // NOLINT
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
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<9>, ZPZV<770>, ZPZV<278>, ZPZV<875>»; };  // NOLINT
template<> struct ConwayPolynomial<881, 1> { using ZPZ = aerobus::zpz<881>; using type = POLYV<ZPZV<1>, ZPZV<878>»; };  // NOLINT
template<> struct ConwayPolynomial<881, 2> { using ZPZ = aerobus::zpz<881>; using type = POLYV<ZPZV<1>, ZPZV<869>, ZPZV<3>»; };  // NOLINT
template<> struct ConwayPolynomial<881, 3> { using ZPZ = aerobus::zpz<881>; using type = POLYV<ZPZV<1>, ZPZV<0>, ZPZV<1>, ZPZV<878>»; };  // NOLINT
template<> struct ConwayPolynomial<881, 4> { using ZPZ = aerobus::zpz<881>; using type = POLYV<ZPZV<1>, ZPZV<0>, ZPZV<0>, ZPZV<447>, ZPZV<3>»; };  // NOLINT
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
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
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<6>, ZPZV<878>»; };  // NOLINT
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>»; };  // NOLINT
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
template<> struct ConwayPolynomial<883, 1> { using ZPZ = aerobus::zpz<883>; using type = POLYV<ZPZV<1>, ZPZV<881>»; };  // NOLINT
template<> struct ConwayPolynomial<883, 2> { using ZPZ = aerobus::zpz<883>; using type = POLYV<ZPZV<1>, ZPZV<879>, ZPZV<2>»; };  // NOLINT
template<> struct ConwayPolynomial<883, 3> { using ZPZ = aerobus::zpz<883>; using type = POLYV<ZPZV<1>, ZPZV<0>, ZPZV<6>, ZPZV<881>»; };  // NOLINT
template<> struct ConwayPolynomial<883, 4> { using ZPZ = aerobus::zpz<883>; using type = POLYV<ZPZV<1>, ZPZV<0>, ZPZV<8>, ZPZV<715>, ZPZV<2>»; };  // NOLINT
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
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
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<881>»; };  // NOLINT
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<2>»; };  // NOLINT
template<> struct ConwayPolynomial<883, 9> { using ZPZ = aerobus::zpz<883>; using type = POLYV<ZPZV<1>, ZPZV<0>, ZPZV<0>, ZPZV<0>, ZPZV<0>, ZPZV<5>, ZPZV<360>, ZPZV<557>, ZPZV<881>»; };  // NOLINT
template<> struct ConwayPolynomial<887, 1> { using ZPZ = aerobus::zpz<887>; using type = POLYV<ZPZV<1>, ZPZV<882>»; };  // NOLINT
template<> struct ConwayPolynomial<887, 2> { using ZPZ = aerobus::zpz<887>; using type = POLYV<ZPZV<1>, ZPZV<885>, ZPZV<5>»; };  // NOLINT
template<> struct ConwayPolynomial<887, 3> { using ZPZ = aerobus::zpz<887>; using type = POLYV<ZPZV<1>, ZPZV<0>, ZPZV<1>, ZPZV<882>»; };  // NOLINT
template<> struct ConwayPolynomial<887, 4> { using ZPZ = aerobus::zpz<887>; using type = POLYV<ZPZV<1>, ZPZV<0>, ZPZV<3>, ZPZV<883>, ZPZV<5>»; };  // NOLINT
template<> struct ConwayPolynomial<887, 5> { using ZPZ = aerobus::zpz<887>; using type = POLYV<ZPZV<1>, ZPZV<0>, ZPZV<0>, ZPZV<0>, ZPZV<5>, ZPZV<882>»; };  // NOLINT
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
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<882>»; };  // NOLINT
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
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
template<> struct ConwayPolynomial<907, 1> { using ZPZ = aerobus::zpz<907>; using type = POLYV<ZPZV<1>, ZPZV<905>»; };  // NOLINT
template<> struct ConwayPolynomial<907, 2> { using ZPZ = aerobus::zpz<907>; using type = POLYV<ZPZV<1>, ZPZV<903>, ZPZV<2>»; };  // NOLINT
template<> struct ConwayPolynomial<907, 3> { using ZPZ = aerobus::zpz<907>; using type = POLYV<ZPZV<1>, ZPZV<0>, ZPZV<4>, ZPZV<905>»; };  // NOLINT
template<> struct ConwayPolynomial<907, 4> { using ZPZ = aerobus::zpz<907>; using type = POLYV<ZPZV<1>, ZPZV<0>, ZPZV<14>, ZPZV<478>, ZPZV<2>»; };  // NOLINT
template<> struct ConwayPolynomial<907, 5> { using ZPZ = aerobus::zpz<907>; using type = POLYV<ZPZV<1>, ZPZV<0>, ZPZV<0>, ZPZV<2>, ZPZV<905>»; };  // NOLINT
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
template<> struct ConwayPolynomial<907, 7> { using ZPZ = aerobus::zpz<907>; using type = POLYV<ZPZV<1>, ZPZV<0>, ZPZV<0>, ZPZV<0>, ZPZV<0>, ZPZV<0>, ZPZV<2>, ZPZV<905>»; };  // NOLINT
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<2>»; };  // NOLINT
```

```
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<1>, ZPZV<783>, ZPZV<57>, ZPZV<905»; };  // NOLINT
template<> struct ConwayPolynomial<911, 1> { using ZPZ = aerobus::zpz<911>; using type = POLYV<ZPZV<1>, ZPZV<894»; };  // NOLINT
template<> struct ConwayPolynomial<911, 2> { using ZPZ = aerobus::zpz<911>; using type = POLYV<ZPZV<1>, ZPZV<909>, ZPZV<17»; };  // NOLINT
template<> struct ConwayPolynomial<911, 3> { using ZPZ = aerobus::zpz<911>; using type = POLYV<ZPZV<1>, ZPZV<0>, ZPZV<1>, ZPZV<894»; };  // NOLINT
template<> struct ConwayPolynomial<911, 4> { using ZPZ = aerobus::zpz<911>; using type = POLYV<ZPZV<1>, ZPZV<0>, ZPZV<11>, ZPZV<887>, ZPZV<17»; };  // NOLINT
template<> struct ConwayPolynomial<911, 5> { using ZPZ = aerobus::zpz<911>; using type = POLYV<ZPZV<1>, ZPZV<0>, ZPZV<0>, ZPZV<0>, ZPZV<2>, ZPZV<894»; };  // NOLINT
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
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»; };  // NOLINT
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<17»; };  // NOLINT
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<679>, ZPZV<116>, ZPZV<894»; };  // NOLINT
template<> struct ConwayPolynomial<919, 1> { using ZPZ = aerobus::zpz<919>; using type = POLYV<ZPZV<1>, ZPZV<912»; };  // NOLINT
template<> struct ConwayPolynomial<919, 2> { using ZPZ = aerobus::zpz<919>; using type = POLYV<ZPZV<1>, ZPZV<910>, ZPZV<7»; };  // NOLINT
template<> struct ConwayPolynomial<919, 3> { using ZPZ = aerobus::zpz<919>; using type = POLYV<ZPZV<1>, ZPZV<0>, ZPZV<2>, ZPZV<912»; };  // NOLINT
template<> struct ConwayPolynomial<919, 4> { using ZPZ = aerobus::zpz<919>; using type = POLYV<ZPZV<1>, ZPZV<0>, ZPZV<3>, ZPZV<602>, ZPZV<7»; };  // NOLINT
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
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
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<912»; };  // NOLINT
template<> struct ConwayPolynomial<919, 8> { using ZPZ = aerobus::zpz<919>; using type = POLYV<ZPZV<1>, ZPZV<0>, ZPZV<0>, ZPZV<0>, ZPZV<6>, ZPZV<708>, ZPZV<202>, ZPZV<504>, ZPZV<7»; };  // NOLINT
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
template<> struct ConwayPolynomial<929, 1> { using ZPZ = aerobus::zpz<929>; using type = POLYV<ZPZV<1>, ZPZV<926»; };  // NOLINT
template<> struct ConwayPolynomial<929, 2> { using ZPZ = aerobus::zpz<929>; using type = POLYV<ZPZV<1>, ZPZV<917>, ZPZV<3»; };  // NOLINT
template<> struct ConwayPolynomial<929, 3> { using ZPZ = aerobus::zpz<929>; using type = POLYV<ZPZV<1>, ZPZV<0>, ZPZV<5>, ZPZV<926»; };  // NOLINT
template<> struct ConwayPolynomial<929, 4> { using ZPZ = aerobus::zpz<929>; using type = POLYV<ZPZV<1>, ZPZV<0>, ZPZV<0>, ZPZV<787>, ZPZV<3»; };  // NOLINT
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
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
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<7>, ZPZV<926»; };  // NOLINT
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»; };  // NOLINT
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
template<> struct ConwayPolynomial<937, 1> { using ZPZ = aerobus::zpz<937>; using type = POLYV<ZPZV<1>, ZPZV<932»; };  // NOLINT
template<> struct ConwayPolynomial<937, 2> { using ZPZ = aerobus::zpz<937>; using type = POLYV<ZPZV<1>, ZPZV<934>, ZPZV<5»; };  // NOLINT
template<> struct ConwayPolynomial<937, 3> { using ZPZ = aerobus::zpz<937>; using type = POLYV<ZPZV<1>, ZPZV<0>, ZPZV<3>, ZPZV<932»; };  // NOLINT
template<> struct ConwayPolynomial<937, 4> { using ZPZ = aerobus::zpz<937>; using type = POLYV<ZPZV<1>, ZPZV<0>, ZPZV<23>, ZPZV<585>, ZPZV<5»; };  // NOLINT
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
template<> struct ConwayPolynomial<937, 6> { using ZPZ = aerobus::zpz<937>; using type = POLYV<ZPZV<1>, ZPZV<0>, ZPZV<0>, ZPZV<794>, ZPZV<727>, ZPZV<934>, ZPZV<5»; };  // NOLINT
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<24>, ZPZV<932»; };  // NOLINT
template<> struct ConwayPolynomial<937, 8> { using ZPZ = aerobus::zpz<937>; using type = POLYV<ZPZV<1>, ZPZV<0>, ZPZV<0>, ZPZV<0>, ZPZV<0>, ZPZV<658>, ZPZV<26>, ZPZV<53>, ZPZV<5»; };  // NOLINT
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<28>, ZPZV<533>, ZPZV<483>, ZPZV<932»; };  // NOLINT
template<> struct ConwayPolynomial<941, 1> { using ZPZ = aerobus::zpz<941>; using type = POLYV<ZPZV<1>, ZPZV<939»; };  // NOLINT
template<> struct ConwayPolynomial<941, 2> { using ZPZ = aerobus::zpz<941>; using type = POLYV<ZPZV<1>, ZPZV<940>, ZPZV<2»; };  // NOLINT
```

```
template<> struct ConwayPolynomial<941, 3> { using ZPZ = aerobus::zpz<941>; using type = POLYV<ZPZV<1>, ZPZV<0>, ZPZV<3>, ZPZV<939»; };  // NOLINT
template<> struct ConwayPolynomial<941, 4> { using ZPZ = aerobus::zpz<941>; using type = POLYV<ZPZV<1>, ZPZV<0>, ZPZV<3>, ZPZV<505>, ZPZV<2»; };  // NOLINT
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
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
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<4>, ZPZV<939»; };  // NOLINT
template<> struct ConwayPolynomial<941, 8> { using ZPZ = aerobus::zpz<941>; using type = POLYV<ZPZV<1>, ZPZV<0>, ZPZV<0>, ZPZV<0>, ZPZV<0>, ZPZV<805>, ZPZV<675>, ZPZV<590>, ZPZV<2»; };  // NOLINT
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<939»; };  // NOLINT
template<> struct ConwayPolynomial<947, 1> { using ZPZ = aerobus::zpz<947>; using type = POLYV<ZPZV<1>, ZPZV<945»; };  // NOLINT
template<> struct ConwayPolynomial<947, 2> { using ZPZ = aerobus::zpz<947>; using type = POLYV<ZPZV<1>, ZPZV<943>, ZPZV<2»; };  // NOLINT
template<> struct ConwayPolynomial<947, 3> { using ZPZ = aerobus::zpz<947>; using type = POLYV<ZPZV<1>, ZPZV<0>, ZPZV<3>, ZPZV<945»; };  // NOLINT
template<> struct ConwayPolynomial<947, 4> { using ZPZ = aerobus::zpz<947>; using type = POLYV<ZPZV<1>, ZPZV<0>, ZPZV<8>, ZPZV<894>, ZPZV<2»; };  // NOLINT
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
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
template<> struct ConwayPolynomial<947, 7> { using ZPZ = aerobus::zpz<947>; using type = POLYV<ZPZV<1>, ZPZV<0>, ZPZV<0>, ZPZV<0>, ZPZV<0>, ZPZV<0>, ZPZV<6>, ZPZV<945»; };  // NOLINT
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»; };  // NOLINT
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<1>, ZPZV<269>, ZPZV<808>, ZPZV<945»; };  // NOLINT
template<> struct ConwayPolynomial<953, 1> { using ZPZ = aerobus::zpz<953>; using type = POLYV<ZPZV<1>, ZPZV<950»; };  // NOLINT
template<> struct ConwayPolynomial<953, 2> { using ZPZ = aerobus::zpz<953>; using type = POLYV<ZPZV<1>, ZPZV<947>, ZPZV<3»; };  // NOLINT
template<> struct ConwayPolynomial<953, 3> { using ZPZ = aerobus::zpz<953>; using type = POLYV<ZPZV<1>, ZPZV<0>, ZPZV<7>, ZPZV<950»; };  // NOLINT
template<> struct ConwayPolynomial<953, 4> { using ZPZ = aerobus::zpz<953>; using type = POLYV<ZPZV<1>, ZPZV<0>, ZPZV<1>, ZPZV<865>, ZPZV<3»; };  // NOLINT
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
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
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<950»; };  // NOLINT
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»; };  // NOLINT
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
template<> struct ConwayPolynomial<967, 1> { using ZPZ = aerobus::zpz<967>; using type = POLYV<ZPZV<1>, ZPZV<962»; };  // NOLINT
template<> struct ConwayPolynomial<967, 2> { using ZPZ = aerobus::zpz<967>; using type = POLYV<ZPZV<1>, ZPZV<965>, ZPZV<5»; };  // NOLINT
template<> struct ConwayPolynomial<967, 3> { using ZPZ = aerobus::zpz<967>; using type = POLYV<ZPZV<1>, ZPZV<0>, ZPZV<1>, ZPZV<962»; };  // NOLINT
template<> struct ConwayPolynomial<967, 4> { using ZPZ = aerobus::zpz<967>; using type = POLYV<ZPZV<1>, ZPZV<0>, ZPZV<3>, ZPZV<963>, ZPZV<5»; };  // NOLINT
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
template<> struct ConwayPolynomial<967, 6> { using ZPZ = aerobus::zpz<967>; using type = POLYV<ZPZV<1>, ZPZV<0>, ZPZV<0>, ZPZV<805>, ZPZV<948>, ZPZV<831>, ZPZV<5»; };  // NOLINT
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<9>, ZPZV<962»; };  // NOLINT
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
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<3>, ZPZV<512>, ZPZV<783>, ZPZV<962»; };  // NOLINT
template<> struct ConwayPolynomial<971, 1> { using ZPZ = aerobus::zpz<971>; using type = POLYV<ZPZV<1>, ZPZV<965»; };  // NOLINT
template<> struct ConwayPolynomial<971, 2> { using ZPZ = aerobus::zpz<971>; using type = POLYV<ZPZV<1>, ZPZV<970>, ZPZV<6»; };  // NOLINT
template<> struct ConwayPolynomial<971, 3> { using ZPZ = aerobus::zpz<971>; using type = POLYV<ZPZV<1>, ZPZV<0>, ZPZV<3>, ZPZV<965»; };  // NOLINT
template<> struct ConwayPolynomial<971, 4> { using ZPZ = aerobus::zpz<971>; using type = POLYV<ZPZV<1>, ZPZV<0>, ZPZV<2>, ZPZV<527>, ZPZV<6»; };  // NOLINT
template<> struct ConwayPolynomial<971, 5> { using ZPZ = aerobus::zpz<971>; using type = POLYV<ZPZV<1>, ZPZV<0>, ZPZV<0>, ZPZV<0>, ZPZV<14>, ZPZV<965»; };  // NOLINT
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
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
template<> struct ConwayPolynomial<971, 8> { using ZPZ = aerobus::zpz<971>; using type =
POLYV<ZPZV<1>, ZPZV<0>, ZPZV<0>, ZPZV<0>, ZPZV<0>, ZPZV<725>, ZPZV<281>, ZPZV<206>, ZPZV<6»; };  //
NOLINT
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<473>, ZPZV<965»;
};  // NOLINT
template<> struct ConwayPolynomial<977, 1> { using ZPZ = aerobus::zpz<977>; using type =
POLYV<ZPZV<1>, ZPZV<974»; };  // NOLINT
template<> struct ConwayPolynomial<977, 2> { using ZPZ = aerobus::zpz<977>; using type =
POLYV<ZPZV<1>, ZPZV<972>, ZPZV<3»; };  // NOLINT
template<> struct ConwayPolynomial<977, 3> { using ZPZ = aerobus::zpz<977>; using type =
POLYV<ZPZV<1>, ZPZV<0>, ZPZV<1>, ZPZV<974»; };  // NOLINT
template<> struct ConwayPolynomial<977, 4> { using ZPZ = aerobus::zpz<977>; using type =
POLYV<ZPZV<1>, ZPZV<0>, ZPZV<0>, ZPZV<800>, ZPZV<3»; };  // NOLINT
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
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»; };  // NOLINT
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<974»; };  // NOLINT
template<> struct ConwayPolynomial<977, 8> { using ZPZ = aerobus::zpz<977>; using type =
POLYV<ZPZV<1>, ZPZV<0>, ZPZV<0>, ZPZV<0>, ZPZV<0>, ZPZV<855>, ZPZV<807>, ZPZV<77>, ZPZV<3»; };  //
NOLINT
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<974»;
};  // NOLINT
template<> struct ConwayPolynomial<983, 1> { using ZPZ = aerobus::zpz<983>; using type =
POLYV<ZPZV<1>, ZPZV<978»; };  // NOLINT
template<> struct ConwayPolynomial<983, 2> { using ZPZ = aerobus::zpz<983>; using type =
POLYV<ZPZV<1>, ZPZV<981>, ZPZV<5»; };  // NOLINT
template<> struct ConwayPolynomial<983, 3> { using ZPZ = aerobus::zpz<983>; using type =
POLYV<ZPZV<1>, ZPZV<0>, ZPZV<1>, ZPZV<978»; };  // NOLINT
template<> struct ConwayPolynomial<983, 4> { using ZPZ = aerobus::zpz<983>; using type =
POLYV<ZPZV<1>, ZPZV<0>, ZPZV<5>, ZPZV<567>, ZPZV<5»; };  // NOLINT
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
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
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<3>, ZPZV<978»; };  // NOLINT
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»; };  //
NOLINT
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<858>, ZPZV<87>, ZPZV<978»;
};  // NOLINT
template<> struct ConwayPolynomial<991, 1> { using ZPZ = aerobus::zpz<991>; using type =
POLYV<ZPZV<1>, ZPZV<985»; };  // NOLINT
template<> struct ConwayPolynomial<991, 2> { using ZPZ = aerobus::zpz<991>; using type =
POLYV<ZPZV<1>, ZPZV<989>, ZPZV<6»; };  // NOLINT
template<> struct ConwayPolynomial<991, 3> { using ZPZ = aerobus::zpz<991>; using type =
POLYV<ZPZV<1>, ZPZV<0>, ZPZV<4>, ZPZV<985»; };  // NOLINT
template<> struct ConwayPolynomial<991, 4> { using ZPZ = aerobus::zpz<991>; using type =
POLYV<ZPZV<1>, ZPZV<0>, ZPZV<10>, ZPZV<794>, ZPZV<6»; };  // NOLINT
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
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
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
template<> struct ConwayPolynomial<991, 8> { using ZPZ = aerobus::zpz<991>; using type =
POLYV<ZPZV<1>, ZPZV<0>, ZPZV<0>, ZPZV<0>, ZPZV<15>, ZPZV<941>, ZPZV<786>, ZPZV<234>, ZPZV<6»; };  //
NOLINT
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<9>, ZPZV<466>, ZPZV<222>, ZPZV<985»;
};  // NOLINT
template<> struct ConwayPolynomial<997, 1> { using ZPZ = aerobus::zpz<997>; using type =
POLYV<ZPZV<1>, ZPZV<990»; };  // NOLINT
template<> struct ConwayPolynomial<997, 2> { using ZPZ = aerobus::zpz<997>; using type =
POLYV<ZPZV<1>, ZPZV<995>, ZPZV<7»; };  // NOLINT
template<> struct ConwayPolynomial<997, 3> { using ZPZ = aerobus::zpz<997>; using type =
POLYV<ZPZV<1>, ZPZV<0>, ZPZV<2>, ZPZV<990»; };  // NOLINT
template<> struct ConwayPolynomial<997, 4> { using ZPZ = aerobus::zpz<997>; using type =
POLYV<ZPZV<1>, ZPZV<0>, ZPZV<4>, ZPZV<622>, ZPZV<7»; };  // NOLINT
template<> struct ConwayPolynomial<997, 5> { using ZPZ = aerobus::zpz<997>; using type =
POLYV<ZPZV<1>, ZPZV<0>, ZPZV<0>, ZPZV<0>, ZPZV<10>, ZPZV<990»; };  // NOLINT
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
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<1>, ZPZV<990»; };  // NOLINT
template<> struct ConwayPolynomial<997, 8> { using ZPZ = aerobus::zpz<997>; using type =
POLYV<ZPZV<1>, ZPZV<0>, ZPZV<0>, ZPZV<0>, ZPZV<0>, ZPZV<934>, ZPZV<473>, ZPZV<241>, ZPZV<7»; };  //
NOLINT
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<39>, ZPZV<732>, ZPZV<616>, ZPZV<990»;
};  // NOLINT
#endif  // AEROBUS_CONWAY_IMPORTS

#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>

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