TP4_Ex1_Dilithium

May 30, 2023

0.1 Trabalho Prático nº4 - Grupo 8

0.1.1 Dilithium

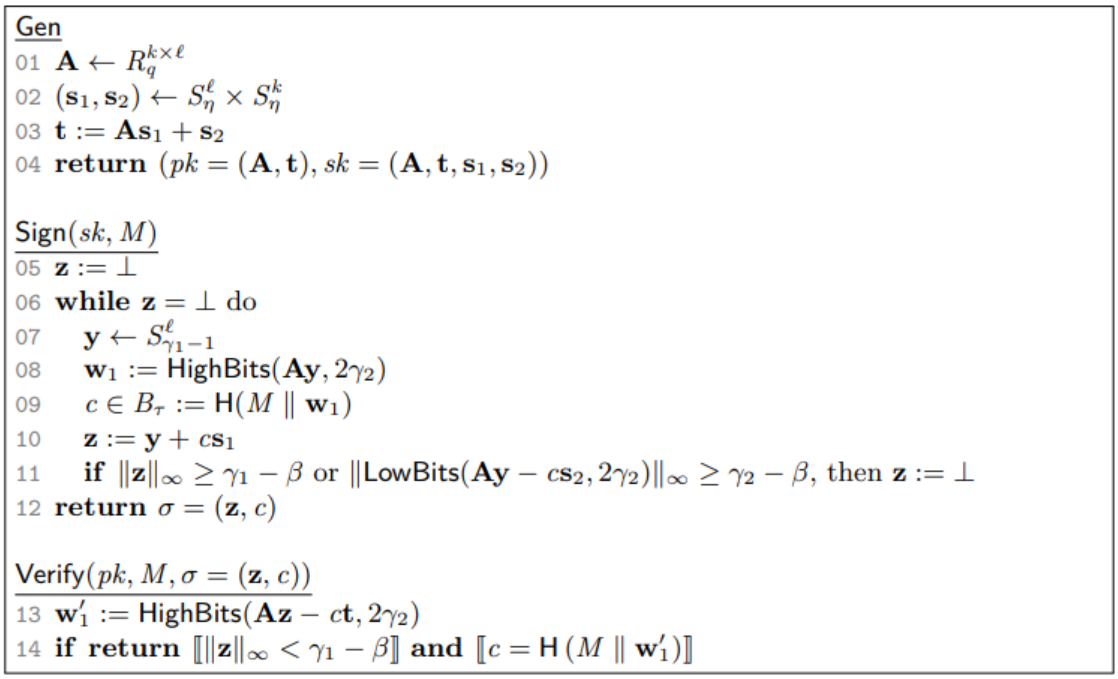
No âmbito do último trabalho prático, foi proposta a implementação da técnica **Dilithium**, na forma de um protótipo. Este é um **esquema da assinatura digital** presente no concurso *NIST PQC* que utiliza o esquema *LWE* básico (com ponto de partida).

Tal como recomendado no enunciado, seguir-se-á o *paper* **Dilithium**.

Este *paper* descreve as diferenças entre a primeira e segunda ronda (*round 1*, *round 2*) submetidas para o algoritmo *Dilithium* e a corrente versão atualizada (*round 3*); podendo agora o esquema ser aleatório, para além de determinístico.

O nível de segurança, do algoritmo *Dilithium*, tem por base a dificuldade em encontrar **pequenos vetores em *lattices***. É projetado para ser: - Simples de implementar (com segurança); - Conservador com os parâmetros; - Um esquema que utiliza uma chave pública e assinaturas de tamanho menor (melhor *performance*); - Modular (fácil de variar os seus parâmetros de segurança).

O *design* do esquema é baseado na abordagem *Fiat-Shamir with Aborts*, sendo uma versão simplificada (e menos eficiente) a seguinte:



Neste caso vê-se os algoritmos para **geração de chaves**, **assinatura** e **verificação**.

Key Generation No processo de geração de chaves, tem-se:

1. Geração de uma matriz k x l, **A** (cada um dos seus valores é um **polinómio** no anel Rq = Zq{X}/(Xn + 1)).
2. Criação das *samples* vetoriais **s1** e **s2** aleatórias, para a **chave privada** (cada coeficiente destes vetores é um elemento de Rq, com coeficientes de tamanho n, no máximo).
3. A segunda parte da chave pública é computada como t = As1 + s2.

Adicionalmente, supõe-se que todas as operações algébricas, aqui, são feitas sobre o anel polinomial, Rq.

Signing Procedure O método `Sign` será o responsável por criar a assinatura final, com base na chave privada (sobre uma dada menagem).

Relativamente a este, tem-se:

1. O procedimento de assinatura começa com a geração de um vetor máscara de polinômios **y** com coeficientes menores que $\gamma 1$, onde $\gamma 1$ é escolhido estrategicamente para equilibrar segurança e resistência contra falsificação (parametrização definida no documento); de forma que a assinatura não revele a chave secreta e não seja facilmente falsificada.

2. Calcular o produto Ay, onde A é a matriz gerada na etapa ExpandA e y é o vetor máscara. Extrair os bits de ordem alta dos coeficientes em Ay para obter w1. Cada coeficiente w em Ay pode ser expresso como $\mathbf{w} = \mathbf{w1} \cdot \mathbf{2\gamma 2} + \mathbf{w0}$, onde $|\mathbf{w0}| \quad \boldsymbol{\gamma}\mathbf{2}$. w1 é um vetor composto por todos os valores w1.

3. Criar um desafio *c* aplicando uma função de *hash* à mensagem e a *w1*. O *output c* é um polinômio em Rq com exatamente τ ± 1's e o restante 0's. A distribuição de *c* é escolhida de forma a que ela tenha uma norma pequena e tenha origem num domínio de tamanho log2(256τ + τ), com o objetivo de estar entre 128 e 256.

4. Calcular a assinatura potencial como $\mathbf{z} = \mathbf{y} + \mathbf{cs1}$, onde s1 é um vetor secreto.

5. Realizar amostragem por rejeição para garantir que a assinatura não revele a chave secreta. Definir o parâmetro β como o coeficiente máximo possível de csi. Se algum coeficiente de z for maior que $\boldsymbol{\gamma}\mathbf{1} \quad \boldsymbol{\beta}$ ou se algum coeficiente dos bits de ordem baixa de $\mathbf{Az} \quad \mathbf{ct}$ (onde t é um termo de ruído) for maior que $\boldsymbol{\gamma}\mathbf{2} \quad \boldsymbol{\beta}$, o procedimento é **reiniciado**. A primeira condição garante a segurança, enquanto a segunda condição garante a correção e a segurança.

6. Repetir o procedimento de assinatura (passos 1-5) até que as condições para amostragem por rejeição sejam satisfeitas. Os parâmetros são ajustados de forma que o número esperado de repetições não seja muito alto.

O objetivo da amostragem por rejeição é evitar que a assinatura dependa da chave secreta e garantir que a assinatura resultante seja segura. Ao repetir o procedimento até que as condições de rejeição sejam atendidas, o assinante pode produzir uma assinatura válida sem revelar informações sensíveis.

Verification Por fim, o `Verify` será responsável por avaliar a correção da assinatura previamente gerada (e se esta encontra-se válida para uma dada mensagem); recebe, para isso, a chave pública.

O verificador primeiro calcula w1 como os bits de ordem alta de Az - ct e, em seguida, aceita a assinatura se todos os coeficientes de z forem menores que γ1 - β e se c for o hash da mensagem e de w1.

[1]:
```
#imports
import os, sys
from cryptography.hazmat.primitives import hashes
from pickle import load, dumps
import random as rn
```

0.1.2 Classe 'Dilithium'

Numa versão mais atualizada, a corrente (atual), do *Dilithium*, apresenta-se numa classe (do mesmo nome), uma implementação do esquema de assinatura.

Utiliza-se a versão mais atualizada, uma vez que o *template* apresentado em cima é bastante **inefeciente**. O maior problema encontra-se na **representação da chave pública**, numa matriz **A**ˆk x l (sendo que esta pode ser penosa, em termos de tamanho). A solução, como iremos ver, parte na utilização de uma *seed* p, utilizando *SHAKE-128* para a criação da matriz **A**. A chave pública será (ρ, t) e o seu tamanho será dominado por t.

Para tal seguiu-se os algoritmos para **geração de chaves**, **assinatura** e **verificação** presentes na **página 19** do *paper* anteriormente mencionado. Estes algoritmos, formalmente, encontram-se apresentados como:

$\underline{\mathsf{Gen}}$

$\zeta \leftarrow \{0,1\}^{256}$

$(\rho, \varsigma, K) \in \{0,1\}^{256\times 3} := \mathrm{H}(\zeta)$ ▷ H is instantiated as SHAKE-256 throughout

$(\mathbf{s}_1, \mathbf{s}_2) \in S_\eta^\ell \times S_\eta^k := \mathrm{H}(\varsigma)$

$\mathbf{A} \in R_q^{k\times\ell} := \mathsf{ExpandA}(\rho)$ ▷ $\mathbf{A}$ is generated and stored in NTT Representation as $\hat{\mathbf{A}}$

$\mathbf{t} := \mathbf{A}\mathbf{s}_1 + \mathbf{s}_2$ ▷ Compute $\mathbf{A}\mathbf{s}_1$ as $\mathtt{NTT}^{-1}(\hat{\mathbf{A}} \cdot \mathtt{NTT}(\mathbf{s}_1))$

$(\mathbf{t}_1, \mathbf{t}_0) := \mathsf{Power2Round}_q(\mathbf{t}, d)$

$tr \in \{0,1\}^{384} := \mathsf{CRH}(\rho \parallel \mathbf{t}_1)$

return $(pk = (\rho, \mathbf{t}_1), sk = (\rho, K, tr, \mathbf{s}_1, \mathbf{s}_2, \mathbf{t}_0))$

$\underline{\mathsf{Sign}(sk, M)}$

$\mathbf{A} \in R_q^{k\times\ell} := \mathsf{ExpandA}(\rho)$ ▷ $\mathbf{A}$ is generated and stored in NTT Representation as $\hat{\mathbf{A}}$

$\mu \in \{0,1\}^{384} := \mathsf{CRH}(tr \parallel M)$

$\kappa := 0$, $(\mathbf{z}, \mathbf{h}) := \perp$

$\rho' \in \{0,1\}^{384} := \mathsf{CRH}(K \parallel \mu)$ (or $\rho' \leftarrow \{0,1\}^{384}$ for randomized signing)

while $(\mathbf{z}, \mathbf{h}) = \perp$ **do** ▷ Pre-compute $\hat{\mathbf{s}}_1 := \mathtt{NTT}(\mathbf{s}_1)$, $\hat{\mathbf{s}}_2 := \mathtt{NTT}(\mathbf{s}_2)$, and $\hat{\mathbf{t}}_0 := \mathtt{NTT}(\mathbf{t}_0)$

  $\mathbf{y} \in \tilde{S}_{\gamma_1}^\ell := \mathsf{ExpandMask}(\rho', \kappa)$

  $\mathbf{w} := \mathbf{A}\mathbf{y}$ ▷ $\mathbf{w} := \mathtt{NTT}^{-1}(\hat{\mathbf{A}} \cdot \mathtt{NTT}(\mathbf{y}))$

  $\mathbf{w}_1 := \mathsf{HighBits}_q(\mathbf{w}, 2\gamma_2)$

  $\tilde{c} \in \{0,1\}^{256} := \mathsf{H}(\mu \parallel \mathbf{w}_1)$

  $c \in B_\tau := \mathsf{SampleInBall}(\tilde{c})$ ▷ Store c in NTT representation as $\hat{c} = \mathtt{NTT}(c)$

  $\mathbf{z} := \mathbf{y} + c\mathbf{s}_1$ ▷ Compute $c\mathbf{s}_1$ as $\mathtt{NTT}^{-1}(\hat{c} \cdot \hat{\mathbf{s}}_1)$

  $\mathbf{r}_0 := \mathsf{LowBits}_q(\mathbf{w} - c\mathbf{s}_2, 2\gamma_2)$ ▷ Compute $c\mathbf{s}_2$ as $\mathtt{NTT}^{-1}(\hat{c} \cdot \hat{\mathbf{s}}_2)$

  if $\|\mathbf{z}\|_\infty \geq \gamma_1 - \beta$ or $\|\mathbf{r}_0\|_\infty \geq \gamma_2 - \beta$, then $(\mathbf{z}, \mathbf{h}) := \perp$

  else

    $\mathbf{h} := \mathsf{MakeHint}_q(-c\mathbf{t}_0, \mathbf{w} - c\mathbf{s}_2 + c\mathbf{t}_0, 2\gamma_2)$ ▷ Compute $c\mathbf{t}_0$ as $\mathtt{NTT}^{-1}(\hat{c} \cdot \hat{\mathbf{t}}_0)$

    if $\|c\mathbf{t}_0\|_\infty \geq \gamma_2$ **or** the # of 1's in $\mathbf{h}$ is greater than ω, then $(\mathbf{z}, \mathbf{h}) := \perp$

  $\kappa := \kappa + \ell$

return $\sigma = (\mathbf{z}, \mathbf{h}, \tilde{c})$

$\underline{\mathsf{Verify}(pk, M, \sigma = (\mathbf{z}, \mathbf{h}, \tilde{c}))}$

$\mathbf{A} \in R_q^{k\times\ell} := \mathsf{ExpandA}(\rho)$ ▷ $\mathbf{A}$ is generated and stored in NTT Representation as $\hat{\mathbf{A}}$

$\mu \in \{0,1\}^{384} := \mathsf{CRH}(\mathsf{CRH}(\rho \parallel \mathbf{t}_1) \parallel M)$

$c := \mathsf{SampleInBall}(\tilde{c})$

$\mathbf{w}_1' := \mathsf{UseHint}_q(\mathbf{h}, \mathbf{A}\mathbf{z} - c\mathbf{t}_1 \cdot 2^d, 2\gamma_2)$ ▷ Compute as $\mathtt{NTT}^{-1}(\hat{\mathbf{A}} \cdot \mathtt{NTT}(\mathbf{z}) - \mathtt{NTT}(c) \cdot \mathtt{NTT}(\mathbf{t}_1 \cdot 2^d))$

return $[\![\|\mathbf{z}\|_\infty < \gamma_1 - \beta]\!]$ **and** $[\![\tilde{c} = \mathsf{H}(\mu \parallel \mathbf{w}_1')]\!]$ **and** $[\![\#$ of 1's in $\mathbf{h}$ is $\leq \omega]\!]$

Como se tinha mencionado previamente, o algoritmo sofreu alterações e, no documento, é apresentado de forma it erativa estas. Sendo assim, os algoritmos atuais para a implementação do esquema de assinatura *Dilithium* são:

Os parâmetros utilizados encontram-se em:

Challenge and Extended Sets	1- -	1-	5+	5++
q	8380417	8380417	8380417	8380417
d	10	13	13	13
weight of c	24	30	60	60
challenge entropy	135	160	257	257
γ_1	2^{17}	2^{17}	2^{19}	2^{19}
γ_2	$(q-1)/128$	$(q-1)/128$	$(q-1)/32$	$(q-1)/32$
(k,ℓ)	$(2,2)$	$(3,3)$	$(9,8)$	$(10,9)$
η	6	3	2	2
β	144	90	120	120
ω	10	80	85	90
pk size (bytes)	864	992	2912	3232
sig size (bytes)	1196	1843	5246	5892
Exp. reps (from Eq. (5))	5.2	4.87	4.59	5.48
BKZ block-size b to break SIS	190 (165)	305 (305)	1055 (1005)	1200 (1145)
Core-SVP Classical	55 (49)	89 (89)	308 (293)	360 (334)
BKZ block-size b to break LWE	200	305	1020	1175
Core-SVP Classical	58	89	298	343

Key generation Tal como mencionado anteriormente, será necessária a criação de chaves públicas-privadas para a criação e verificação das assinaturas. Para tal seguir-se-á o algoritmo para a função `Gen`, como apresentada no *paper*.

1. Cria-se um valor aleatório de 32 bytes, ζ, para gerar as variáveis ρ, ς, K com o método `H3`.
2. Cria-se as samples *s1* e *s2* (vetores), usando a variável ς a partir do método `createSamplesH`.
3. Cria-se a matriz A Rq, a partir de ρ e o método `ExpandA`.
4. O vetor t é calculado através da expressão $t := A$s1 + s2, *a utilizar no método* `Power2Round` *para partir* bit wise* os elementos do anel Rq (o *output* é o par (t1,t0)).
5. A variável *t1* e ρ será utilizada na função de *hash CRH* para gerar *tr*.
6. A chave pública *pk = (p, t1)* e a chave privada *sk=(p,k,tr,s1,s2,t0)*.

...A matriz A tem por base a criação de um inteiro 2 bytes em formato *little endian* (n i + j, i {o..k}* e j *{0..l}*). Vão ser utilizados no método `hash128`, para depois gerar um *integer* a partir do método `generate_Int`. Os 3 bytes (resultantes desta função) serão interpretados como um inteiro em formato *little endian*. Todos coeficentes dos polinómios da matriz A serão assim calculados.

Signing procedure O método `Sign` é responsável por assinar uma mensagem para envio, retornando a assinatura gerada; recorrendo à chave privada *sk* e a mensagem em *bytes M*.

1. O método inicia-se com a geração da matriz A (`ExpandA` e a variável ρ).
2. Gera-se a variável μ usando a função *CRH* com as variáveis *tr* e *M*
3. Do mesmo modo, gera-se a variável ρ', neste caso, com as variáveis K e μ.

São geradas as variáveis z e h, dentro de um ciclo, e serão atualizadas até satisfazerem a condição de saída: *||z||* *γ1* *β ou ||r0||* *γ2* *β* e o número de 1's em h ω.

5. Começa-se por criar a variável y recorrendo ao método `ExpandMask` (com as variáveis ρ' e k).

6. Gera-se o vetor w ($w = A$ y*). Com base neste, e no método* `HighBits` *(extração dos* high-order bits* do `Decompose`), calcula-se *w1*; que terá uso no *sampling in a ball.*

7. Antes de realizar o *sampling in a ball* (que resulta na variável c), tem-se de calcular a variável *c2.* Esta será a partir do uso do método `H1` em conjunto com μ, como *seed.*

8. z é o resultado de $z = y + c$ s1*.

9. Calcula-se *r0* com o método `LowBits` (extração os *low-order bits* do`Decompose`), tal como descrito no *paper.*

10. A variável h é o resultado de `MakeHint` (dica a ser usada por `UseHint`).

11. Caso a condição não for satisfeita -> $k = k + l$.

12. Caso a condição seja satisfeita, a assinatura é dada por *a=(z,h,c2).*

A geração do vetor y é realizada da seguinte maneira:

1. Primeiro, é gerado um número inteiro de 2 bytes em formato *little endian* a partir da expressão *k+i*, onde *i* pertence ao conjunto *{0...l}.*
2. Esses 2 *bytes* são combinados com a variável *ρ'* e passados para a função de hash `hash256`.
3. O output da função `hash256` é, então, utilizado como entrada para a função `generate_Int_Mask`.
4. A função generate_Int_ask processa os primeiros 3 bytes do output recebido. Dependendo se γ1 é igual a 2^17 ou 2^19, ela transforma os dois primeiros bytes em 2 bits ou os 4 bits do terceiro byte.
5. Os 3 bytes resultantes são interpretados como um número inteiro em formato little endian, gerando assim os coeficientes dos polinômios que compõem o vetor y. Dessa forma, o vetor y é gerado com base nos passos descritos acima.

Por outro lado, a geração do vetor c é feita da seguinte forma:

1. Utiliza-se a função `hash256` para gerar uma *hash* de *n bytes* a partir da variável recebida como parâmetro.
2. Em seguida, são gerados os *sign bits* utilizando a função `generate_Bits`.
3. Essa função usa os primeiros *8 bytes*, onde os primeiros *r bits* (peso de c) correspondem aos *sign bits.*
4. Cada *byte* do *output* de *hash256* corresponde a um número inteiro em formato *little endian*, que será gerado pela função `generate_Int_J`.
5. Após obter o valor inteiro j e os *sign bits s*, aplica-se o algoritmo *"sample in ball"* conforme descrito na página 10 da documentação. O resultado final é o vetor c gerado.

Verification O método `Verify` tem como objetivo verificar a autenticidade da assinatura recebida como parâmetro quando associada à mensagem M, utilizando a chave pública.

Para verificar a autenticidade da assinatura, verifica-se a seguinte condição: *//z// < γ1 β e c' = H(μ // w') e o número de 1's em h é ω.*

Começa-se por calcular a matriz A (`ExpandA` e a variável ρ). Calcula-se a variável μ usando o método `CRH` sobre aplicada a *CRH(ρ + t1) + M.*

c é o resultado da utilização do método `SampleInBall` e a variável *c2.* Calcula-se a dica com `UseHint` para determinar *w1 = UseHint(h, A* z c * t1 * 2^d, 2 * γ2)*.

[2]:
```
class Dilithium:

  ␣
↪#############################################################################################
   ## ---------------------- Parametrização Dilithium - nível 5+ (NIST)␣
↪---------------------- ##
  ␣
↪#############################################################################################

   def __init__(self):
       self.q = 8380417
       self.d = 13
       self.c = 60
       self.n = 256
       self.y1 = 2**19
       self.y2 = (self.q - 1)//32
       self.k = 9
       self.l = 8
       self.nn = 2
       self.b = 120
       self.w = 85

       Z.<x> = ZZ[]
       R.<x> = QuotientRing(Z,Z.ideal(x^self.n+1))

       self.R = R

       Zq.<x> = GF(self.q)[]
       fi = x^self.n + 1
       Rq.<x> = QuotientRing(Zq,Zq.ideal(fi))

       self.Rq = Rq

  ␣
↪#############################################################################################
   ## ---------------------------------- Funções Principais␣
↪---------------------------------- ##
  ␣
↪#############################################################################################

   # Função que cria um par de chaves privada-pública (pk, sk)
   def Gen(self):

       # ζ ← {0, 1}^256 (256 bits = 32 bytes)
       z = os.urandom(32)

       #(ρ, ς, K)  {0, 1}^256×3 := H(ζ)
```

```
        (ro, c, K) = self.H3(z)

        #(s1,s2)  Sη^l × Sη^k := H(ς)
        (s1,s2) = self.createSamplesH(c)

        #A  Rq^k×l := ExpandA(ρ)
        A = self.ExpandA(ro)

        #t := A*s1 + s2
        t = A * s1 + s2

        #(t1,t0) := Power2Round(t, d)
        (t1, t0) = self.Power2Round(t, self.d)

        #tr  {0, 1}^384 := CRH(ρ || t1)
        pt1 = ro + dumps(t1)
        tr = self.CRH(pt1)

        #return (pk = (ρ, t1), sk = (ρ, K, tr, s1, s2, t0))
        #return pk, sk
        return (ro, t1),(ro, K, tr, s1, s2, t0)

    #Implementação do método para assinar uma dada mensagem, usando a chave␣
↪privada sk
    def Sign(self, sk, M):

        (ro, K, tr, s1, s2, t0) = sk

        #A  Rq^k×l := ExpandA(ρ)
        A = self.ExpandA(ro)

        #μ  {0, 1}^384:= CRH(tr || M)
        u = self.CRH(tr + M)

        #κ := 0
        k = 0

        #(z, h) :=
        (z, h) = (None, None)

        #ρ'  {0,1}^384 := CRH(K || μ)
        ro2 = self.CRH(K + u)

        # while (z, h) =  do
        while z == None or h == None:

            #y  Sγ^l := ExpandMask(ρ', κ)
```

```
            y = self.ExpandMask(ro2, k)

            #w := Ay
            w = A * y

            #w1 := HighBits(w, 2*γ2 )
            w1 = self.HighBits(w, 2*self.y2)

            #c'  {0,1}^256 := H(μ || w1)
            c2 = self.H1(u + dumps(w1), 32)

            #c  Bτ := SampleInBall(c')
            c = self.SampleInBall(c2)

            #z := y + c * s1
            z = y + c * s1

            #r0 := LowBits(w  c*s2 , 2*γ2)
            r0 = self.LowBits(w - c * s2 , 2 * self.y2)

            #if ||z||  γ1  β or ||r0||  γ2  β
            if self.normInf(z) >= (self.y1 - self.b) and self.normInf(r0) >=
↪(self.y2 - self.b):

                #then (z, h) :=
                (z, h) = (None, None)

            else:

                #h := MakeHint(c*t0 , w  c*s2 + c*t0 , 2*γ2)
                h = self.MakeHint(-c*t0 , w - c*s2 + c*t0 , 2 * self.y2)

                #||c*t0||  γ2 or the # of 1's in h is greater than ω
                if self.normInf(c * t0) >= self.y2 or h.count(True) > self.w:

                    #(z, h) :=
                    (z, h) = (None, None)

            #κ := κ + l
            k = k + self.l

        # return σ = (z, h, ~c)
        return (z,h,c2)

    #Implementação do método para verificar uma assinatura, a partir da chave
↪pública
    def Verify(self, pk, M, sig):
```

```
        (ro, t1) = pk
        (z,h,c2) = sig

        #A  Rq^k×l := ExpandA(ρ)
        A = self.ExpandA(ro)

        #μ  {0, 1}^384 := CRH(CRH(ρ + t1) + M )
        y = self.CRH(self.CRH(ro + dumps(t1)) + M)

        #c := SampleInBall(c')
        c = self.SampleInBall(c2)

        #w' = UseHint(h, A*z  c*t1 * 2^d , 2*γ2)
        w1 = self.UseHint(h, (self.Rq2R(A) * z) - (c * t1 * 2^self.d), 2 * self.
↪y2)

        #return ||z|| < γ1  β and c' = H(μ || w') and J# of 1's in h is  ω
        return self.normInf(z) < (self.y1-self.b) and c2 == self.H1(y +␣
↪dumps(w1), 32) and h.count(True) <= self.w


  ␣
↪#############################################################################################
    ## ---------------------------------- Funções Auxiliares␣
↪---------------------------------- ##
  ␣
↪#############################################################################################

    ## ---------------- Método H3 ---------------- ##

    #Implementação do método H - criação do triplo (ρ, ς, K)
    def H3(self, C):

        digest = hashes.Hash(hashes.SHAKE256(int(96)))
        digest.update(C)
        buffer = digest.finalize()

        # 96 bytes (256bits x 3)
        return (buffer[:32], buffer[32:64], buffer[64:])

    ## ---------------- Método H1 ---------------- ##

    #Implementação do método H - um único output
    def H1(self, m, length):

        digest = hashes.Hash(hashes.SHAKE256(int(length)))
        digest.update(m)
```

```
        h = digest.finalize()

        return h

    ## ---------------- Método createSamplesH ---------------- ##

    #Implementação do mod+- (Modular Reductions)
    def modPMAux(self,r,a):

        if a%2 == 0:
            lim = a/2
        else:
            lim = (a-1)/2

        mod = r % a

        if mod > lim:
            mod -= a

        return mod

    # Geração de s1 e s2 (samples), com base numa seed c
    def createSamplesH(self, c):

        h = self.H1(c, self.l*self.n + self.k*self.n)
        ind = 0
        s1 = []

        for i in range(self.l):
            p = []
            for j in range(self.n):
                num = h[ind]
                p.append(self.modPMAux(num,self.nn))
                ind += 1

            s1.append(self.R(p))
            res_s1 = vector(s1)

        s2 = []

        for i in range(self.k):
            p = []
            for j in range(self.n):
                num = h[ind]
                p.append(self.modPMAux(num,self.nn))
                ind += 1
```

```
            s2.append(self.R(p))
            res_s2 = vector(s2)

        return (res_s1, res_s2)

    ## ---------------- Método ExpandA ---------------- ##

    # Método Hash - Utiliza SHAKE128
    def hash128(self, seed):

        digest = hashes.Hash(hashes.SHAKE128(int(self.n-1)))
        digest.update(seed)

        return digest.finalize()

    # Geração do integer como especificado
    #(This is done by setting the highest bit of every third byte to zero and␣
↪interpreting blocks of
    #3 consecutive bytes in little endian byte order. So for example the three␣
↪bytes b0, b1 and b2
    #are used to get the integer 0  b2· 2^16 + b1 · 2^8 + b0  2^23  1 where b2␣
↪is the logical
    #AND of b2 and 2^128  1.)
    def generate_Int(self, offset, seed):

        if(len(offset)<3):
            offset = seed

        thr = bytearray(offset[:3])
        thr[2]  &= 0x7f
        thr = bytes(thr)

        res = int.from_bytes(thr, "little")

        return res, offset[3:]

    # Mapeia uma seed uniforme (ρ) para uma matriz A (pertencente a Rq^(kxl)) em␣
↪NTT
    # A matriz será utilizada para multiplicações nos algoritmos do esquema
    # O output será  A  Zq^256, ou seja, a representação de A no domínio NTT
    def ExpandA(self,ro):

        A = []

        for i in range(self.k):
            row = []
            for j in range(self.l):
```

```
                coefs = []
                intTwoBytes = int(self.n*i + j).to_bytes(2, "little")
                hash_res = self.hash128(ro + intTwoBytes)
                seed = hash_res

                for cof in range(self.n):
                    number, hash_res = self.generate_Int(hash_res, seed)

                    coefs.append(number)
                row.append(self.Rq(coefs))
            A.append(row)

        res = Matrix(A)

        return res

    ## ---------------- Método Power2Round ---------------- ##

    # Implementação do mod+- num vetor
    def modPM(self, v,a):

        r = []

        for i in v:
            iAux = []
            for j in i:
                iAux.append(self.modPMAux(j,a))

            r.append(self.R(iAux))

        res = vector(r)

        return res

    # Implementação do mod+ num vetor
    def modP(self, v,a):

        r = []

        for i in v:
            iAux = []
            for j in i:
                iAux.append(mod(j,a))

            r.append(self.R(iAux))

        res = vector(r)
```

```
        return res

    # Método para  realizar o arredondamento de potência de 2 durante a
    # geração de chaves e a assinatura de mensagens.
    # Ela ajuda a garantir a segurança do esquema e a resistência a ataques.
    def Power2Round(self,r,d):

        #r := r mod+ q
        r = self.modP(r,self.q)

        #r0 := r mod± 2^d
        r0 = self.modPM(r,2^d)

        # return ((r  r0)/2^d, r0)
        return ((r - r0)/2^d, r0)

    ## ---------------- Método Collision Resistant hashing ---------------- ##

    #Implementação do Collision resistant hashing - SHAKE-256
    def CRH(self,seed):

        digest = hashes.Hash(hashes.SHAKE256(int(48)))
        digest.update(seed)

        return digest.finalize()

    ## ---------------- Método ExpandMask ---------------- ##

    # Método Hash - SHAKE-256
    def hash256(self,seed):

        digest = hashes.Hash(hashes.SHAKE256(int(self.n-1)))
        digest.update(seed)

        return digest.finalize()

    # Método que gera o inteiro segundo regras apresentadas
    def generate_Int_Mask(self, offset, seed):

        if(len(offset)<3):
            offset = seed

        thr = bytearray(offset[:3])

        if self.y1 == 2**17:
            thr[2]  &= 0x3
```

```
        else:
            thr[2]  &= 0xf

        thr = bytes(thr)
        res = int.from_bytes(thr, "little")

        return res, offset[3:]

    # Método para realizar o sampling dos vetores
    def ExpandMask(self, ro, k):

        yAux = []

        for i in range(self.l):
            coefs = []
            two = int(k + i).to_bytes(2, "little")
            hash_res = self.hash256(ro + two)
            seed = hash_res

            for cof in range(self.n):
                res, hash_res = self.generate_Int_Mask(hash_res, seed)

                coefs.append(res - (self.y1 -1))
            yAux.append(self.R(coefs))

        y = vector(yAux)

        return y

    ## ---------------- Método normInf ---------------- ##

    # Implementação de um método para normalização (infinito)
    def normInf(self, v):

        maxsV = []
        for vi in v:
            maxsVi = []
            for i in vi:
                maxsVi.append(self.modPMAux(i,self.q))
            maxsV.append(max(maxsVi))
        res = max(maxsV)

        return res

    ## ---------------- Método SampleInBall ---------------- ##

    # Método que retira dos primeiros 8 bytes os sign bits
```

```
    def generate_Bits(self,seed):

        digest = int.from_bytes(seed[:8], 'little')

        bits = [int(digit) for digit in list(ZZ(digest).binary())]

        return bits[:self.c], seed[8:]

    # Método que gera o valor de j
    def generate_Int_J(self,offset, seed):

        if(len(offset)<1):
            offset = seed

        res = int.from_bytes(bytes(offset[0]), "little")

        return res, offset[1:]

    # Implementação do método para hashing to a Ball
    # Essa função é usada para amostrar um vetor aleatório em uma bola
    # centrada na origem, onde alguns dos elementos são trocados entre
    # si com base em índices aleatórios e bits gerados aleatoriamente.
    def SampleInBall(self,ro):

        c = [0]*self.n
        hash_res = self.hash256(ro)
        seed = hash_res

        sign, hash_res = self.generate_Bits(hash_res)

        for i in range(self.n - self.c, self.n):

            j, hashS_res = self.generate_Int_J(hash_res, seed)
            s = sign[i-self.n + self.c]
            c[i] = c[j]
            c[j] = (-1)^s

        return self.R(c)

    ## ---------------- Método Decompose ---------------- ##

    # Essa função é usada para decompor um vetor r em dois componentes r0 e r1,
    # que são usados em várias operações do esquema Dilithium, como a geração
    # de dicas e a verificação de assinaturas.
    def Decompose(self,r,alpha):

        r0_list = []
```

```
        r1_list = []

        r0 = self.modP(r,self.q)
        r1 = self.modPM(r0,alpha)

        for r0i, r1i in zip(r0, r1):
            r0_coefs = []
            r1_coefs = []
            for r0ij, r1ij in zip(r0i, r1i):
                if (r0ij - r1ij == self.q - 1):
                    r2ij = 0
                    r1ij = r1ij-1
                else:
                    r2ij = (r0ij - r1ij)/alpha
                r0_coefs.append(r1ij)
                r1_coefs.append(r2ij)
            r0_list.append(self.R(r1_coefs))
            r1_list.append(self.R(r0_coefs))

        return (vector(r0_list),vector(r1_list))

    ## ---------------- Método UseHint ---------------- ##

    # Método que utiliza o hint para recuperar high-order bits
    # Gerar o vetor resultante para posterior comparação com o valor esperado
    def UseHint(self,h,r,alpha):

        m = (self.q - 1)/alpha
        (r1,r0) = self.Decompose(r,alpha)

        rAux = []
        i = 0

        for r0i, r1i in zip(r0,r1):
            elem = []
            for r0ij, r1ij in zip(r0i,r1i):
                if h[i] :
                    if r0ij > 0:
                        elem.append(mod((r1ij + 1),m))
                    else:
                         elem.append(mod((r1ij - 1),m))
                else:
                    elem.append(r1ij)
                i += 1
            rAux.append(self.R(elem))

        res = vector(rAux)
```

```
        return res

    ## ---------------- Método Rq2R ---------------- ##

    # Método que converte uma Matriz de Rq para R
    def Rq2R(self, A):

        mtx = []
        for row in A:
            newRow = []
            for elem in row:
                newRow.append(self.R(elem))
            mtx.append(newRow)

        res = Matrix(mtx)
        return res

    ## ---------------- Método HighBits ---------------- ##

    # Método para extrair higher-order bits (depois de aplicar o Decompose)
    def HighBits(self,r,alpha):

        (r1,r0) = self.Decompose(r,alpha)

        return r1

    ## ---------------- Método LowBits ---------------- ##

    # Método para extrair lower-order bits (depois do resultado do decompose)
    def LowBits(self,r,alpha):

        (r1,r0) = self.Decompose(r,alpha)

        return r0

    ## ---------------- Método MakeHint ---------------- ##

    # Método que calcula um hint (para recuperar high-order bits)
    # Verificar se uma assinatura é válida ou não.
    def MakeHint(self,z,r,alpha):

        #  r1 := HighBitsq(r, a)
        r1 = self.HighBits(r,alpha)

        #  v1 := HighBitsq(r + z, a)
        v1 = self.HighBits(r + z,alpha)
```

```
        #  return [r1 /= v1]
        res = []
        for r1Auxi, v1Auxi in zip(r1, v1):
            for r1Auxj, v1Auxj in zip(r1Auxi, v1Auxi):
                res.append(r1Auxj != v1Auxj)

        return res
```

0.1.3 Teste do esquema de assinatura - DILITHIUM

[3]:
```
# Classe que implementa os métodos do esquema
d = Dilithium()

# Geração das chaves pública-privada
pk, sk = d.Gen()

print("Chaves pública-privada criadas...")
print()
print("Pública: ", pk)
print()
print("Privada: ", sk)
print()

msg1 = "Mensagem aleatória"
msg2 = "Outra mensagem"

print("Mensagem a assinar: ", msg1)
print()

print("A assinar mensagem...")

# Assinatura da mensagem usando a chave publica
sig = d.Sign(sk, msg1.encode())

print("Assinatura gerada: ")
print(sig)

# Verificação da assinatura usando a chave pública
ver = d.Verify(pk, msg1.encode(), sig)

if ver == True:
    print("Assinatura validada!")

else:
    print("Ocorreu um erro!")

ver2 = d.Verify(pk, msg2.encode(), sig)
```

```
print("A testar com mensagem inválida... ", msg2)

if ver2 == True:
    print("Assinatura validada!")

else:
    print("Ocorreu um erro!")
```

```
Chaves pública-privada criadas...

Pública:  (b'\xd1\x97\xbf\xc0U\xa0\xc42\x19\xb8D\xff=
\xd6y\xb2;\xa9\x1d\xb6P\x9c\x8d\xfc%\xdd\xbc\xef\xd0p\x08', (815*x^255 +
993*x^254 + 516*x^253 + 253*x^252 + 84*x^251 + 438*x^250 + 701*x^249 + 587*x^248
+ 288*x^247 + 867*x^246 + 21*x^245 + 822*x^244 + 209*x^243 + 171*x^242 +
477*x^241 + 46*x^240 + 190*x^239 + 604*x^238 + 535*x^237 + 889*x^236 + 89*x^235
+ 296*x^234 + 136*x^233 + 841*x^232 + 972*x^231 + 802*x^230 + 772*x^229 +
737*x^228 + 399*x^227 + 70*x^226 + x^225 + 605*x^224 + 686*x^223 + 553*x^222 +
387*x^221 + 451*x^220 + 406*x^219 + 544*x^218 + 684*x^217 + 645*x^216 + 49*x^215
+ 896*x^214 + 385*x^213 + 958*x^212 + 223*x^211 + 974*x^210 + 244*x^209 +
688*x^208 + 340*x^207 + 366*x^206 + 636*x^205 + 577*x^204 + 537*x^203 + 14*x^202
+ 80*x^201 + 606*x^200 + 966*x^199 + 545*x^198 + 118*x^197 + 617*x^196 +
262*x^195 + 918*x^194 + 461*x^193 + 790*x^192 + 849*x^191 + 885*x^190 +
795*x^189 + 776*x^188 + 438*x^187 + 964*x^186 + 312*x^185 + 269*x^184 +
588*x^183 + 925*x^182 + 407*x^181 + 378*x^180 + 158*x^179 + 731*x^178 +
787*x^177 + 836*x^176 + 825*x^175 + 977*x^174 + 641*x^173 + 914*x^172 +
805*x^171 + 209*x^170 + 377*x^169 + 772*x^168 + 146*x^167 + 331*x^166 +
771*x^165 + 496*x^164 + 421*x^163 + 565*x^162 + 315*x^161 + 319*x^160 +
769*x^159 + 569*x^158 + 387*x^157 + 211*x^156 + 549*x^155 + 376*x^154 +
808*x^153 + 509*x^152 + 435*x^151 + 241*x^150 + 921*x^149 + 577*x^148 +
685*x^147 + 874*x^146 + 774*x^145 + 643*x^144 + 744*x^143 + 79*x^142 + 582*x^141
+ 998*x^140 + 9*x^139 + 896*x^138 + 773*x^137 + 196*x^136 + 415*x^135 +
217*x^134 + 143*x^133 + 583*x^132 + 75*x^131 + 15*x^130 + 534*x^129 + 644*x^128
+ 667*x^127 + 957*x^126 + 103*x^125 + 612*x^124 + 664*x^123 + 24*x^122 +
340*x^121 + 353*x^120 + 203*x^119 + 110*x^118 + 911*x^117 + 649*x^116 +
422*x^115 + 310*x^114 + 288*x^113 + 789*x^112 + 966*x^111 + 868*x^110 +
194*x^109 + 701*x^108 + 686*x^107 + 567*x^106 + 647*x^105 + 77*x^104 + 23*x^103
+ 166*x^102 + 192*x^101 + 889*x^100 + 161*x^99 + 415*x^98 + 532*x^97 + 217*x^96
+ 668*x^95 + 997*x^94 + 841*x^93 + 289*x^92 + 165*x^91 + 547*x^90 + 984*x^89 +
148*x^88 + 148*x^87 + 77*x^86 + 711*x^85 + 918*x^84 + 916*x^83 + 226*x^82 +
618*x^81 + 683*x^80 + 607*x^79 + 1019*x^78 + 452*x^77 + 291*x^76 + 991*x^75 +
584*x^74 + 579*x^73 + 845*x^72 + 15*x^71 + 234*x^70 + 735*x^69 + 788*x^68 +
581*x^67 + 353*x^66 + 618*x^65 + 694*x^64 + 790*x^63 + 374*x^62 + 299*x^61 +
832*x^60 + 321*x^59 + 106*x^58 + 452*x^57 + 1008*x^56 + 613*x^55 + 694*x^54 +
115*x^53 + 754*x^52 + 111*x^51 + 164*x^50 + 284*x^49 + 331*x^48 + 354*x^47 +
502*x^46 + 305*x^45 + 453*x^44 + 196*x^43 + 982*x^42 + 765*x^41 + 303*x^40 +
890*x^39 + 222*x^38 + 346*x^37 + 420*x^36 + 832*x^35 + 661*x^34 + 96*x^33 +
419*x^32 + 406*x^31 + 861*x^30 + 350*x^29 + 757*x^28 + 174*x^27 + 374*x^26 +
```

```
391*x^25 + 343*x^24 + 93*x^23 + 354*x^22 + 574*x^21 + 421*x^20 + 974*x^19 +
525*x^18 + 651*x^17 + 575*x^16 + 866*x^15 + 137*x^14 + 785*x^13 + 346*x^12 +
936*x^11 + 516*x^10 + 339*x^9 + 887*x^8 + 414*x^7 + 215*x^6 + 105*x^5 + 864*x^4
+ 815*x^3 + 720*x^2 + 170*x + 836, 149*x^255 + 183*x^254 + 919*x^253 + 804*x^252
+ 561*x^251 + 918*x^250 + 773*x^249 + 163*x^248 + 131*x^247 + 252*x^246 +
604*x^245 + 173*x^244 + 329*x^243 + 125*x^242 + 950*x^241 + 683*x^240 + 10*x^239
+ 851*x^238 + 194*x^237 + 922*x^236 + 173*x^235 + 365*x^234 + 63*x^233 +
925*x^232 + 753*x^231 + 483*x^230 + 838*x^229 + 319*x^228 + 581*x^227 +
817*x^226 + 616*x^225 + 670*x^224 + 690*x^223 + 365*x^222 + 789*x^221 +
298*x^220 + 551*x^219 + 600*x^218 + 875*x^217 + 997*x^216 + 188*x^215 +
554*x^214 + 480*x^213 + 90*x^212 + 872*x^211 + 853*x^210 + 905*x^209 + 564*x^208
+ 477*x^207 + 348*x^206 + 444*x^205 + 619*x^204 + 383*x^203 + 687*x^202 +
241*x^201 + 503*x^200 + 773*x^199 + 169*x^198 + 292*x^197 + 58*x^196 + 800*x^195
+ 576*x^194 + 13*x^193 + 842*x^192 + 752*x^191 + 20*x^190 + 747*x^189 +
549*x^188 + 443*x^187 + 119*x^186 + 185*x^185 + 198*x^184 + 420*x^183 +
691*x^182 + 190*x^181 + 161*x^180 + 787*x^179 + 684*x^178 + 698*x^177 +
260*x^176 + 105*x^175 + 855*x^174 + 1015*x^173 + 749*x^172 + 445*x^171 +
716*x^170 + 169*x^169 + 885*x^168 + 638*x^167 + 1006*x^166 + 154*x^165 +
225*x^164 + 506*x^163 + 462*x^162 + 752*x^161 + 782*x^160 + 538*x^159 +
1011*x^158 + 312*x^157 + 1019*x^156 + 1013*x^155 + 922*x^154 + 541*x^153 +
714*x^152 + 24*x^151 + 347*x^150 + 995*x^149 + 929*x^148 + 90*x^147 + 301*x^146
+ 428*x^145 + 761*x^144 + 731*x^143 + 820*x^142 + 452*x^141 + 958*x^140 +
970*x^139 + 556*x^138 + 933*x^137 + 236*x^136 + 107*x^135 + 977*x^134 + 85*x^133
+ 824*x^132 + 569*x^131 + 492*x^130 + 539*x^129 + 943*x^128 + 25*x^127 +
351*x^126 + 814*x^125 + 739*x^124 + 713*x^123 + 978*x^122 + 76*x^121 + 146*x^120
+ 785*x^119 + 109*x^118 + 982*x^117 + 438*x^116 + 255*x^115 + 512*x^114 +
51*x^113 + 368*x^112 + 842*x^111 + 335*x^110 + 372*x^109 + 104*x^108 + 898*x^107
+ 468*x^106 + 691*x^105 + 479*x^104 + 5*x^103 + 606*x^102 + 531*x^101 +
853*x^100 + 632*x^99 + 944*x^98 + 883*x^97 + 466*x^96 + 306*x^95 + 275*x^94 +
x^93 + 992*x^92 + 162*x^91 + 111*x^90 + 336*x^89 + 30*x^88 + 251*x^87 + 154*x^86
+ 682*x^85 + 291*x^84 + 711*x^83 + 646*x^82 + 12*x^81 + 739*x^80 + 303*x^79 +
223*x^78 + 438*x^77 + 815*x^76 + 545*x^75 + 573*x^74 + 739*x^73 + 308*x^72 +
850*x^71 + 54*x^70 + 711*x^69 + 376*x^68 + 685*x^67 + 280*x^66 + 909*x^65 +
80*x^64 + 770*x^63 + 430*x^62 + 328*x^61 + 530*x^60 + 834*x^59 + 194*x^58 +
947*x^57 + 564*x^56 + 468*x^55 + 494*x^54 + 199*x^53 + 994*x^52 + 238*x^51 +
767*x^50 + 714*x^49 + 127*x^48 + 78*x^47 + 565*x^46 + 140*x^45 + 62*x^44 +
338*x^43 + 129*x^42 + 873*x^41 + 357*x^40 + 717*x^39 + 595*x^38 + 423*x^37 +
212*x^36 + 544*x^35 + 606*x^34 + 294*x^33 + 630*x^32 + 356*x^31 + 350*x^30 +
610*x^29 + 969*x^28 + 491*x^27 + 683*x^26 + 754*x^25 + 778*x^24 + 1012*x^23 +
132*x^22 + 766*x^21 + 777*x^20 + 895*x^19 + 271*x^18 + 351*x^17 + 821*x^16 +
410*x^15 + 906*x^14 + 615*x^13 + 186*x^12 + 892*x^11 + 614*x^10 + 152*x^9 +
416*x^8 + 305*x^7 + 980*x^6 + 410*x^5 + 105*x^4 + 769*x^3 + 552*x^2 + 681*x +
889, 650*x^255 + 39*x^254 + 490*x^253 + 608*x^252 + 840*x^251 + 86*x^250 +
282*x^249 + 271*x^248 + 63*x^247 + 416*x^246 + 936*x^245 + 905*x^244 + 323*x^243
+ 699*x^242 + 197*x^241 + 918*x^240 + 779*x^239 + 97*x^238 + 231*x^237 +
594*x^236 + 819*x^235 + 160*x^234 + 781*x^233 + 677*x^232 + 598*x^231 +
752*x^230 + 184*x^229 + 41*x^228 + 44*x^227 + 385*x^226 + 728*x^225 + 259*x^224
+ 935*x^223 + 204*x^222 + 247*x^221 + 358*x^220 + 505*x^219 + 742*x^218 +
```

```
6*x^217 + 388*x^216 + 539*x^215 + 424*x^214 + 523*x^213 + 495*x^212 + 233*x^211
+ 103*x^210 + x^209 + 790*x^208 + 608*x^207 + 620*x^206 + 980*x^205 + 427*x^204
+ 770*x^203 + 439*x^202 + 22*x^201 + 381*x^200 + 270*x^199 + 316*x^198 +
283*x^197 + 506*x^196 + 653*x^195 + 718*x^194 + 205*x^193 + 23*x^192 + 547*x^191
+ 561*x^190 + 108*x^189 + 5*x^188 + 971*x^187 + 20*x^186 + 918*x^185 + 763*x^184
+ 913*x^183 + 631*x^182 + 363*x^181 + 752*x^180 + 712*x^179 + 902*x^178 +
588*x^177 + 605*x^176 + 453*x^175 + 11*x^174 + 312*x^173 + 769*x^172 + 492*x^171
+ 786*x^170 + 202*x^169 + 956*x^168 + 482*x^167 + 33*x^166 + 24*x^165 +
186*x^164 + 786*x^163 + 169*x^162 + 766*x^161 + 369*x^160 + 410*x^159 +
799*x^158 + 182*x^157 + 53*x^156 + 592*x^155 + 82*x^154 + 787*x^153 + 50*x^152 +
166*x^151 + 840*x^150 + 777*x^149 + 52*x^148 + 1015*x^147 + 218*x^146 +
996*x^145 + 56*x^144 + 793*x^143 + 64*x^142 + 272*x^141 + 168*x^140 + 89*x^139 +
653*x^138 + 490*x^137 + 450*x^136 + 1010*x^135 + 44*x^134 + 509*x^133 +
410*x^132 + 526*x^131 + 837*x^130 + 772*x^129 + 588*x^128 + 77*x^127 + 636*x^126
+ 109*x^125 + 86*x^124 + 68*x^123 + 337*x^122 + 666*x^121 + 848*x^120 +
783*x^119 + 262*x^118 + 228*x^117 + 151*x^116 + 853*x^115 + 25*x^114 + 113*x^113
+ 119*x^112 + 437*x^111 + 39*x^110 + 37*x^109 + 415*x^108 + 817*x^107 +
216*x^106 + 226*x^105 + 781*x^104 + 260*x^103 + 410*x^102 + 90*x^101 + 343*x^100
+ 626*x^99 + 734*x^98 + 547*x^97 + 922*x^96 + 697*x^95 + 18*x^94 + 766*x^93 +
296*x^92 + 984*x^91 + 255*x^90 + 350*x^89 + 687*x^88 + 103*x^87 + 210*x^86 +
240*x^85 + 77*x^84 + 556*x^83 + 225*x^82 + 674*x^81 + 1002*x^80 + 366*x^79 +
351*x^78 + 815*x^77 + 582*x^76 + 311*x^75 + 482*x^74 + 839*x^73 + 280*x^72 +
142*x^71 + 499*x^70 + 233*x^69 + 161*x^68 + 300*x^67 + 545*x^66 + 464*x^65 +
463*x^64 + 862*x^63 + 220*x^62 + 152*x^61 + 821*x^60 + 748*x^59 + 501*x^58 +
167*x^57 + 357*x^56 + 36*x^55 + 443*x^54 + 1018*x^53 + 690*x^52 + 33*x^51 +
654*x^50 + 633*x^49 + 996*x^48 + 916*x^47 + 413*x^46 + 417*x^45 + 216*x^44 +
579*x^43 + 1016*x^42 + 912*x^41 + 225*x^40 + 956*x^39 + 86*x^38 + 301*x^37 +
585*x^36 + 420*x^35 + 117*x^34 + 632*x^33 + 381*x^32 + 292*x^31 + 971*x^30 +
105*x^29 + 284*x^28 + 524*x^27 + 833*x^26 + 965*x^25 + 766*x^24 + 436*x^23 +
1000*x^22 + 428*x^21 + 876*x^20 + 452*x^19 + 751*x^18 + 807*x^17 + 315*x^16 +
607*x^15 + 71*x^14 + 867*x^13 + 805*x^12 + 292*x^11 + 307*x^10 + 88*x^9 +
676*x^8 + 49*x^7 + 684*x^6 + 218*x^5 + 311*x^4 + 901*x^3 + 494*x^2 + 894*x +
215, 961*x^255 + 206*x^254 + 369*x^253 + 1020*x^252 + 91*x^251 + 591*x^250 +
327*x^249 + 41*x^248 + 245*x^247 + 837*x^246 + 452*x^245 + 237*x^244 + 107*x^243
+ 880*x^242 + 259*x^241 + 678*x^240 + 900*x^239 + 789*x^238 + 284*x^237 +
621*x^236 + 45*x^235 + 299*x^234 + 29*x^233 + 724*x^232 + 506*x^231 + 921*x^230
+ 463*x^229 + 126*x^228 + 422*x^227 + 133*x^226 + 962*x^225 + 4*x^224 +
586*x^223 + 760*x^222 + 809*x^221 + 84*x^220 + 79*x^219 + 624*x^218 + 88*x^217 +
838*x^216 + 122*x^215 + 298*x^214 + 375*x^213 + 991*x^212 + 308*x^211 + 78*x^210
+ 391*x^209 + 280*x^208 + 605*x^207 + 384*x^206 + 139*x^205 + 134*x^204 +
372*x^203 + 477*x^202 + 297*x^201 + 831*x^200 + 89*x^199 + 883*x^198 + 339*x^197
+ 903*x^196 + 771*x^195 + 873*x^194 + 285*x^193 + 937*x^192 + 782*x^191 +
308*x^190 + 268*x^189 + 836*x^188 + 326*x^187 + 304*x^186 + 1016*x^185 +
230*x^184 + 850*x^183 + 270*x^182 + 55*x^181 + 198*x^180 + 385*x^179 + 139*x^178
+ 862*x^177 + 52*x^176 + 119*x^175 + 31*x^174 + 996*x^173 + 875*x^172 +
417*x^171 + 291*x^170 + 320*x^169 + 207*x^168 + 459*x^167 + 536*x^166 +
231*x^165 + 309*x^164 + 59*x^163 + 425*x^162 + 81*x^161 + 919*x^160 + 834*x^159
+ 667*x^158 + 592*x^157 + 45*x^156 + 788*x^155 + 485*x^154 + 966*x^153 +
```

```
69*x^152 + 407*x^151 + 939*x^150 + 874*x^149 + 891*x^148 + 642*x^147 + 561*x^146
+ 285*x^145 + 182*x^144 + 669*x^143 + 17*x^142 + 41*x^141 + 600*x^140 +
541*x^139 + 627*x^138 + 830*x^137 + 296*x^136 + 51*x^135 + 811*x^134 + 521*x^133
+ 850*x^132 + 434*x^131 + 368*x^130 + 90*x^129 + 74*x^128 + 121*x^127 +
493*x^126 + 316*x^125 + 334*x^124 + 659*x^123 + 616*x^122 + 86*x^121 + 649*x^120
+ 172*x^119 + 952*x^118 + 652*x^117 + 495*x^116 + 989*x^115 + 623*x^114 +
908*x^113 + 154*x^112 + 242*x^111 + 968*x^110 + 702*x^109 + 537*x^108 +
581*x^107 + 519*x^106 + 509*x^105 + 880*x^104 + 743*x^103 + 248*x^102 +
161*x^101 + 289*x^100 + 406*x^99 + 708*x^98 + 303*x^97 + 134*x^96 + 363*x^95 +
969*x^94 + 993*x^93 + 1005*x^92 + 984*x^91 + 598*x^90 + 705*x^89 + 933*x^88 +
983*x^87 + 559*x^86 + 317*x^85 + 561*x^84 + 283*x^83 + 663*x^82 + 842*x^81 +
517*x^80 + 289*x^79 + 754*x^78 + 69*x^77 + 751*x^76 + 915*x^75 + 942*x^74 +
752*x^73 + 894*x^72 + 794*x^71 + 409*x^70 + 377*x^69 + 28*x^68 + 364*x^67 +
376*x^66 + 566*x^65 + 1019*x^64 + 301*x^63 + 838*x^62 + 229*x^61 + 630*x^60 +
450*x^59 + 83*x^58 + 878*x^57 + 472*x^56 + 115*x^55 + 811*x^54 + 154*x^53 +
711*x^52 + 218*x^51 + 199*x^50 + 992*x^49 + 157*x^48 + 661*x^47 + 1008*x^46 +
51*x^45 + 115*x^44 + 59*x^43 + 533*x^42 + 1005*x^41 + 805*x^40 + 863*x^39 +
579*x^38 + 119*x^37 + 37*x^36 + 700*x^35 + 1003*x^34 + 281*x^33 + 399*x^32 +
105*x^31 + 40*x^30 + 517*x^29 + 957*x^28 + 907*x^27 + 494*x^26 + 889*x^25 +
294*x^24 + 94*x^23 + 796*x^22 + 801*x^21 + 926*x^20 + 607*x^19 + 822*x^18 +
309*x^17 + 612*x^16 + 306*x^15 + 344*x^14 + 86*x^13 + 509*x^12 + 318*x^11 +
841*x^10 + 267*x^9 + 656*x^8 + 352*x^7 + 769*x^6 + 51*x^5 + 826*x^4 + 670*x^3 +
922*x^2 + 691*x + 248, 974*x^255 + 137*x^254 + 629*x^253 + 750*x^252 + 79*x^251
+ 870*x^250 + 412*x^249 + 702*x^248 + 346*x^247 + 359*x^246 + 427*x^245 +
551*x^244 + 1016*x^243 + 686*x^242 + 309*x^241 + 717*x^240 + 736*x^239 +
865*x^238 + 577*x^237 + 88*x^236 + 979*x^235 + 350*x^234 + 847*x^233 + 349*x^232
+ 410*x^231 + 586*x^230 + 975*x^229 + 382*x^228 + 466*x^227 + 344*x^226 +
297*x^225 + 621*x^224 + 825*x^223 + 206*x^222 + 564*x^221 + 384*x^220 +
692*x^219 + 369*x^218 + 19*x^217 + 154*x^216 + 362*x^215 + 30*x^214 + 399*x^213
+ 474*x^212 + 294*x^211 + 471*x^210 + 861*x^209 + 29*x^208 + 519*x^207 +
619*x^206 + 624*x^205 + 592*x^204 + 5*x^203 + 722*x^202 + 80*x^201 + 13*x^200 +
725*x^199 + 672*x^198 + 361*x^197 + 674*x^196 + 615*x^195 + 356*x^194 +
918*x^193 + 16*x^192 + 759*x^191 + 109*x^190 + 351*x^189 + 808*x^188 + 882*x^187
+ 129*x^186 + 1021*x^185 + 90*x^184 + 300*x^183 + 520*x^182 + 213*x^181 +
18*x^180 + 315*x^179 + 24*x^178 + 59*x^177 + 728*x^176 + 271*x^175 + 769*x^174 +
215*x^173 + 466*x^172 + 145*x^171 + 775*x^170 + 979*x^169 + 477*x^168 + 29*x^167
+ 312*x^166 + 224*x^165 + 168*x^164 + 238*x^163 + 296*x^162 + 441*x^161 +
244*x^160 + 286*x^159 + 209*x^158 + 75*x^157 + 598*x^156 + 882*x^155 + 418*x^154
+ 562*x^153 + 979*x^152 + 85*x^151 + 939*x^150 + 826*x^149 + 16*x^148 +
918*x^147 + 362*x^146 + 573*x^145 + 374*x^144 + 548*x^143 + 237*x^142 +
589*x^141 + 416*x^140 + 974*x^139 + 826*x^138 + 709*x^137 + 163*x^136 +
941*x^135 + 821*x^134 + 4*x^133 + 210*x^132 + 461*x^131 + 269*x^130 + 725*x^129
+ 181*x^128 + 24*x^127 + 234*x^126 + 167*x^125 + 650*x^124 + 734*x^123 +
694*x^122 + 50*x^121 + 730*x^120 + 517*x^119 + 349*x^118 + 537*x^117 + 452*x^116
+ 69*x^115 + 631*x^114 + 11*x^113 + 857*x^112 + 731*x^111 + 1011*x^110 +
36*x^109 + 349*x^108 + 680*x^107 + 523*x^106 + 244*x^105 + 791*x^104 + 640*x^103
+ 235*x^102 + 596*x^101 + 721*x^100 + 279*x^99 + 733*x^98 + 440*x^97 + 377*x^96
+ 87*x^95 + 370*x^94 + 229*x^93 + 455*x^92 + 685*x^91 + 369*x^90 + 997*x^89 +
```

```
93*x^88 + 304*x^87 + 232*x^86 + 69*x^85 + 626*x^84 + 810*x^83 + 530*x^82 +
397*x^81 + 314*x^80 + 871*x^79 + 371*x^78 + 854*x^77 + 57*x^76 + 222*x^75 +
437*x^74 + 828*x^73 + 676*x^72 + 974*x^71 + 934*x^70 + 426*x^69 + 833*x^68 +
403*x^67 + 13*x^66 + 248*x^65 + 284*x^64 + 614*x^63 + 934*x^62 + 467*x^61 +
254*x^60 + 191*x^59 + 125*x^58 + 48*x^57 + 213*x^56 + 199*x^55 + 211*x^54 +
338*x^53 + 484*x^52 + 372*x^51 + 251*x^50 + 115*x^49 + 866*x^48 + 557*x^47 +
608*x^46 + 467*x^45 + 189*x^44 + 141*x^43 + 919*x^42 + 711*x^41 + 1005*x^40 +
10*x^39 + 1009*x^38 + 959*x^37 + 876*x^36 + 220*x^35 + 308*x^34 + 88*x^33 +
827*x^32 + 721*x^31 + 871*x^30 + 981*x^29 + 376*x^28 + 1008*x^27 + 917*x^26 +
1000*x^25 + 79*x^24 + 467*x^23 + 195*x^22 + 819*x^21 + 775*x^20 + 641*x^19 +
785*x^18 + 479*x^17 + 827*x^16 + 629*x^15 + 525*x^14 + 652*x^13 + 618*x^12 +
686*x^11 + 10*x^10 + 420*x^9 + 167*x^8 + 309*x^7 + 513*x^6 + 994*x^5 + 434*x^4 +
910*x^3 + 272*x^2 + 441*x + 951, 717*x^255 + 62*x^254 + 968*x^253 + 424*x^252 +
449*x^251 + 832*x^250 + 285*x^249 + 151*x^248 + 287*x^247 + 468*x^246 +
438*x^245 + 486*x^244 + 792*x^243 + 754*x^242 + 282*x^241 + 707*x^240 +
275*x^239 + 741*x^238 + 60*x^237 + 529*x^236 + 954*x^235 + 310*x^234 + 206*x^233
+ 744*x^232 + 1020*x^231 + 725*x^230 + 645*x^229 + 253*x^228 + 981*x^227 +
723*x^226 + 203*x^225 + 309*x^224 + 681*x^223 + 713*x^222 + 876*x^221 +
947*x^220 + 582*x^219 + 335*x^218 + 868*x^217 + 35*x^216 + 137*x^215 + 903*x^214
+ 959*x^213 + 30*x^212 + 972*x^211 + 145*x^210 + 299*x^209 + 892*x^208 +
417*x^207 + 505*x^206 + 15*x^205 + 183*x^204 + 657*x^203 + 3*x^202 + 762*x^201 +
717*x^200 + 76*x^199 + 441*x^198 + 782*x^197 + 550*x^196 + 97*x^195 + 335*x^194
+ 771*x^193 + 927*x^192 + 532*x^191 + 171*x^190 + 323*x^189 + 797*x^188 +
617*x^187 + 1008*x^186 + 302*x^185 + 290*x^184 + 797*x^183 + 809*x^182 +
240*x^181 + 322*x^180 + 457*x^179 + 543*x^178 + 944*x^177 + 500*x^176 +
722*x^175 + 645*x^174 + 14*x^173 + 355*x^172 + 612*x^171 + 369*x^170 +
1022*x^169 + 723*x^168 + 60*x^167 + 774*x^166 + 853*x^165 + 60*x^164 + 728*x^163
+ 659*x^162 + 877*x^161 + 331*x^160 + 265*x^159 + 906*x^158 + 560*x^157 +
567*x^156 + 583*x^155 + 88*x^154 + 744*x^153 + 23*x^152 + 312*x^151 + 421*x^150
+ 1013*x^149 + 831*x^148 + 947*x^147 + 924*x^146 + 209*x^145 + 133*x^144 +
342*x^143 + 972*x^142 + 354*x^141 + 32*x^140 + 147*x^139 + 415*x^138 + 683*x^137
+ 761*x^136 + 447*x^135 + 900*x^134 + 31*x^133 + 872*x^132 + 649*x^131 +
310*x^130 + 694*x^129 + 701*x^128 + 630*x^127 + 715*x^126 + 48*x^125 + 938*x^124
+ 175*x^123 + 822*x^122 + 606*x^121 + 398*x^120 + 966*x^119 + 691*x^118 +
660*x^117 + 422*x^116 + 873*x^115 + 376*x^114 + 861*x^113 + 561*x^112 +
495*x^111 + 194*x^110 + 364*x^109 + 363*x^108 + 885*x^107 + 964*x^106 +
845*x^105 + 556*x^104 + 132*x^103 + 484*x^102 + 497*x^101 + 1001*x^100 +
860*x^99 + 43*x^98 + 805*x^97 + 826*x^96 + 46*x^95 + 903*x^94 + 302*x^93 +
621*x^92 + 501*x^91 + 304*x^90 + 911*x^89 + 38*x^88 + 705*x^87 + 896*x^86 +
515*x^85 + 174*x^84 + 718*x^83 + 559*x^82 + 997*x^81 + 664*x^80 + 714*x^79 +
87*x^78 + 837*x^77 + 922*x^76 + 143*x^75 + 954*x^74 + 477*x^73 + 196*x^72 +
844*x^71 + 710*x^70 + 530*x^69 + 989*x^68 + 782*x^67 + 234*x^66 + 632*x^65 +
8*x^64 + 83*x^63 + 835*x^62 + 552*x^61 + 846*x^60 + 707*x^59 + 343*x^58 +
166*x^57 + 343*x^56 + 1022*x^55 + 642*x^54 + 302*x^53 + 132*x^52 + 419*x^51 +
468*x^50 + 222*x^49 + 143*x^48 + 354*x^47 + 888*x^46 + 719*x^45 + 296*x^44 +
282*x^43 + 889*x^42 + 911*x^41 + 480*x^40 + 249*x^39 + 836*x^38 + 248*x^37 +
753*x^36 + 620*x^35 + 830*x^34 + 62*x^33 + 110*x^32 + 594*x^31 + 286*x^30 +
16*x^29 + 1009*x^28 + 857*x^27 + 991*x^26 + 834*x^25 + 582*x^24 + 22*x^23 +
```

```
435*x^22 + 337*x^21 + 847*x^20 + 977*x^19 + 979*x^18 + 320*x^17 + 406*x^16 +
86*x^15 + 351*x^14 + 491*x^13 + 23*x^12 + 89*x^11 + 194*x^10 + 538*x^9 + 71*x^8
+ 813*x^7 + 990*x^6 + 678*x^5 + 652*x^4 + 702*x^3 + 926*x^2 + 537*x + 22,
51*x^255 + 466*x^254 + 215*x^253 + 607*x^252 + 873*x^251 + 467*x^250 + 97*x^249
+ 644*x^248 + 680*x^247 + 591*x^246 + 952*x^245 + 721*x^244 + 978*x^243 +
302*x^242 + 792*x^241 + 153*x^240 + 401*x^239 + 303*x^238 + 337*x^237 +
192*x^236 + 724*x^235 + 360*x^234 + 505*x^233 + 86*x^232 + 350*x^231 + 868*x^230
+ 717*x^229 + 353*x^228 + 594*x^227 + 815*x^226 + 121*x^225 + 289*x^224 +
65*x^223 + 972*x^222 + 156*x^221 + 996*x^220 + 976*x^219 + 838*x^218 + 19*x^217
+ 246*x^216 + 943*x^215 + 198*x^214 + 857*x^213 + 36*x^212 + 652*x^211 +
102*x^210 + 728*x^209 + 607*x^208 + 367*x^207 + 255*x^206 + 826*x^205 +
755*x^204 + 984*x^203 + 34*x^202 + 328*x^201 + 761*x^200 + 793*x^199 + 153*x^198
+ 178*x^197 + 340*x^196 + 754*x^195 + 357*x^194 + 135*x^193 + 704*x^192 +
976*x^191 + 428*x^190 + 582*x^189 + 262*x^188 + 609*x^187 + 547*x^186 +
963*x^185 + 334*x^184 + 896*x^183 + 629*x^182 + 493*x^181 + 613*x^180 +
588*x^179 + 17*x^178 + 518*x^177 + 137*x^176 + 445*x^175 + 715*x^174 + 585*x^173
+ 323*x^172 + 341*x^171 + 235*x^170 + 565*x^169 + 295*x^168 + 749*x^167 +
462*x^166 + 780*x^165 + 976*x^164 + 494*x^163 + 768*x^162 + 70*x^161 + 160*x^160
+ 714*x^159 + 651*x^158 + 354*x^157 + 312*x^156 + 826*x^155 + 392*x^154 +
630*x^153 + 42*x^152 + 734*x^151 + 1001*x^150 + 238*x^149 + 404*x^148 +
414*x^147 + 280*x^146 + 554*x^145 + 300*x^144 + 567*x^143 + 541*x^142 +
373*x^141 + 476*x^140 + 94*x^139 + 351*x^138 + 622*x^137 + 422*x^136 + 960*x^135
+ 797*x^134 + 201*x^133 + 802*x^132 + 60*x^131 + 433*x^130 + 310*x^129 +
184*x^128 + 154*x^127 + 873*x^126 + 73*x^125 + 115*x^124 + 551*x^123 + 248*x^122
+ 553*x^121 + 68*x^120 + 187*x^119 + 726*x^118 + 926*x^117 + 513*x^116 +
369*x^115 + 657*x^114 + 205*x^113 + 643*x^112 + 224*x^111 + 693*x^110 +
293*x^109 + 848*x^108 + 636*x^107 + 500*x^106 + 60*x^105 + 726*x^104 + 795*x^103
+ 970*x^102 + 723*x^101 + 677*x^100 + 572*x^99 + 384*x^98 + 688*x^97 + 103*x^96
+ 699*x^95 + 737*x^94 + 982*x^93 + 144*x^92 + 91*x^91 + 189*x^90 + 792*x^89 +
725*x^88 + 429*x^87 + 386*x^86 + 447*x^85 + 459*x^84 + 897*x^83 + 530*x^82 +
160*x^81 + 24*x^80 + 802*x^79 + 868*x^78 + 720*x^77 + 1014*x^76 + 479*x^75 +
30*x^74 + 231*x^73 + 407*x^72 + 110*x^71 + 361*x^70 + 111*x^69 + 726*x^68 +
199*x^67 + 542*x^66 + 443*x^65 + 830*x^64 + 57*x^63 + 712*x^62 + 930*x^61 +
623*x^60 + 604*x^59 + 115*x^58 + 512*x^57 + 228*x^56 + 374*x^55 + 497*x^54 +
169*x^53 + 616*x^52 + 638*x^51 + 142*x^50 + 341*x^49 + 100*x^48 + 285*x^47 +
970*x^46 + 213*x^45 + 634*x^44 + 230*x^43 + 574*x^42 + 898*x^41 + 646*x^40 +
119*x^39 + 705*x^38 + 468*x^37 + 197*x^36 + 758*x^35 + 301*x^34 + 118*x^33 +
419*x^32 + 502*x^31 + 901*x^30 + 332*x^29 + 482*x^28 + 195*x^27 + 706*x^26 +
194*x^25 + 274*x^24 + 813*x^23 + 241*x^22 + 260*x^21 + 703*x^20 + 705*x^19 +
413*x^18 + 898*x^17 + 969*x^16 + 1002*x^15 + 761*x^14 + 924*x^13 + 236*x^12 +
62*x^11 + 169*x^10 + 957*x^9 + 855*x^8 + 467*x^7 + 449*x^6 + 357*x^5 + 993*x^4 +
749*x^3 + 973*x^2 + 1012*x + 798, 479*x^255 + 274*x^254 + 202*x^253 + 578*x^252
+ 950*x^251 + 728*x^250 + 1010*x^249 + 66*x^248 + 162*x^247 + 924*x^246 +
665*x^245 + 752*x^244 + 746*x^243 + 517*x^242 + 217*x^241 + 672*x^240 +
575*x^239 + 369*x^238 + 760*x^237 + 826*x^236 + 806*x^235 + 383*x^234 +
832*x^233 + 934*x^232 + 962*x^231 + 225*x^230 + 413*x^229 + 809*x^228 +
553*x^227 + 35*x^226 + 111*x^225 + 367*x^224 + 551*x^223 + 473*x^222 + 740*x^221
+ 112*x^220 + 452*x^219 + 391*x^218 + 394*x^217 + 520*x^216 + 165*x^215 +
```

```
175*x^214 + 812*x^213 + 61*x^212 + 746*x^211 + 1011*x^210 + 768*x^209 +
863*x^208 + 837*x^207 + 237*x^206 + 619*x^205 + 914*x^204 + 654*x^203 +
310*x^202 + 813*x^201 + 935*x^200 + 51*x^199 + 300*x^198 + 851*x^197 + 722*x^196
+ 319*x^195 + 618*x^194 + 615*x^193 + 862*x^192 + 214*x^191 + 154*x^190 +
300*x^189 + 48*x^188 + 222*x^187 + 770*x^186 + 1017*x^185 + 775*x^184 + 77*x^183
+ 932*x^182 + 330*x^181 + 437*x^180 + 379*x^179 + 618*x^178 + 442*x^177 +
1018*x^176 + 566*x^175 + 706*x^174 + 200*x^173 + 481*x^172 + 169*x^171 +
115*x^170 + 48*x^169 + 1005*x^168 + 169*x^167 + 604*x^166 + 120*x^165 +
342*x^164 + 458*x^163 + 652*x^162 + 207*x^161 + 862*x^160 + 141*x^159 +
247*x^158 + 354*x^157 + 376*x^156 + 949*x^155 + 465*x^154 + 34*x^153 + 765*x^152
+ 690*x^151 + 520*x^150 + 136*x^149 + 428*x^148 + 742*x^147 + 163*x^146 +
404*x^145 + 7*x^144 + 683*x^143 + 1013*x^142 + 1020*x^141 + 753*x^140 +
160*x^139 + 220*x^138 + 47*x^137 + 554*x^136 + 921*x^135 + 756*x^134 + 355*x^133
+ 96*x^132 + 753*x^131 + 546*x^130 + 111*x^129 + 509*x^128 + 502*x^127 +
988*x^126 + 170*x^125 + 727*x^124 + 914*x^123 + 883*x^122 + 957*x^121 +
122*x^120 + 265*x^119 + 253*x^118 + 902*x^117 + 469*x^116 + 421*x^115 +
1005*x^114 + 516*x^113 + 620*x^112 + 118*x^111 + 28*x^110 + 166*x^109 + 95*x^108
+ 247*x^107 + 202*x^106 + 295*x^105 + 288*x^104 + 371*x^103 + 834*x^102 +
357*x^101 + 248*x^100 + 97*x^99 + 9*x^98 + 48*x^97 + 750*x^96 + 569*x^95 +
734*x^94 + 323*x^93 + 602*x^92 + 84*x^91 + 346*x^90 + 725*x^89 + 645*x^88 +
117*x^87 + 527*x^86 + 886*x^85 + 44*x^84 + 977*x^83 + 376*x^82 + 692*x^81 +
166*x^80 + 311*x^79 + 308*x^78 + 923*x^77 + 587*x^76 + 831*x^75 + 20*x^74 +
953*x^73 + 587*x^72 + 554*x^71 + 287*x^70 + 341*x^69 + 41*x^68 + 668*x^67 +
951*x^66 + 772*x^65 + 579*x^64 + 577*x^63 + 937*x^62 + 743*x^61 + 867*x^60 +
208*x^59 + 797*x^58 + 354*x^57 + 583*x^56 + 608*x^55 + 731*x^54 + 274*x^53 +
813*x^52 + 453*x^51 + 219*x^50 + 159*x^49 + 236*x^48 + 104*x^47 + 64*x^46 +
221*x^45 + 155*x^44 + 743*x^43 + 558*x^42 + 947*x^41 + 247*x^40 + 378*x^39 +
379*x^38 + 272*x^37 + 558*x^36 + 880*x^35 + 674*x^34 + 845*x^33 + 13*x^32 +
575*x^31 + 842*x^30 + 136*x^29 + 755*x^28 + 95*x^27 + 133*x^26 + 453*x^25 +
126*x^24 + 404*x^23 + 499*x^22 + 187*x^21 + 685*x^20 + 495*x^19 + 759*x^18 +
960*x^17 + 674*x^16 + 789*x^15 + 732*x^14 + 790*x^13 + 759*x^12 + 708*x^11 +
253*x^10 + 208*x^9 + 102*x^8 + 429*x^7 + 886*x^6 + 759*x^5 + 619*x^4 + 473*x^3 +
468*x^2 + 763*x + 814, 472*x^255 + 626*x^254 + 661*x^253 + 148*x^252 + 610*x^251
+ 813*x^250 + 50*x^249 + 829*x^248 + 248*x^247 + 426*x^246 + 876*x^245 +
955*x^244 + 952*x^243 + 723*x^242 + 697*x^241 + 267*x^240 + 663*x^239 +
694*x^238 + 214*x^237 + 713*x^236 + 672*x^235 + 614*x^234 + 570*x^233 +
731*x^232 + 159*x^231 + 363*x^230 + 545*x^229 + 757*x^228 + 77*x^227 + 917*x^226
+ 624*x^225 + 672*x^224 + 98*x^223 + 115*x^222 + 805*x^221 + 658*x^220 +
22*x^219 + 608*x^218 + 572*x^217 + 838*x^216 + 514*x^215 + 41*x^214 + 152*x^213
+ 554*x^212 + 211*x^211 + 105*x^210 + 935*x^209 + 574*x^208 + 753*x^207 +
981*x^206 + 690*x^205 + 894*x^204 + 831*x^203 + 289*x^202 + 573*x^201 +
674*x^200 + 469*x^199 + 579*x^198 + 243*x^197 + 834*x^196 + 251*x^195 +
570*x^194 + 125*x^193 + 284*x^192 + 372*x^191 + 374*x^190 + 945*x^189 +
989*x^188 + 255*x^187 + 571*x^186 + 278*x^185 + 402*x^184 + 289*x^183 +
1011*x^182 + 956*x^181 + 78*x^180 + 315*x^179 + 234*x^178 + 312*x^177 +
865*x^176 + 904*x^175 + 500*x^174 + 877*x^173 + 876*x^172 + 216*x^171 +
786*x^170 + 788*x^169 + 103*x^168 + 153*x^167 + 393*x^166 + 232*x^165 +
605*x^164 + 104*x^163 + 393*x^162 + 520*x^161 + 357*x^160 + 193*x^159 +
```

```
118*x^158 + 938*x^157 + 834*x^156 + 615*x^155 + 816*x^154 + 117*x^153 +
260*x^152 + 108*x^151 + 500*x^150 + 592*x^149 + 583*x^148 + 468*x^147 +
784*x^146 + 719*x^145 + 445*x^144 + 134*x^143 + 214*x^142 + 728*x^141 +
1022*x^140 + 324*x^139 + 533*x^138 + 966*x^137 + 511*x^136 + 635*x^135 +
510*x^134 + 340*x^133 + 801*x^132 + 133*x^131 + 755*x^130 + 632*x^129 +
1019*x^128 + 453*x^127 + 502*x^126 + 265*x^125 + 688*x^124 + 606*x^123 +
585*x^122 + 372*x^121 + 135*x^120 + 658*x^119 + 763*x^118 + 143*x^117 +
513*x^116 + 485*x^115 + 874*x^114 + 127*x^113 + 789*x^112 + 188*x^111 +
883*x^110 + 721*x^109 + 774*x^108 + 150*x^107 + 902*x^106 + 156*x^105 +
596*x^104 + 167*x^103 + 609*x^102 + 319*x^101 + 664*x^100 + 358*x^99 + 594*x^98
+ 978*x^97 + 98*x^96 + 520*x^95 + 30*x^94 + 636*x^93 + 386*x^92 + 401*x^91 +
803*x^90 + 776*x^89 + 149*x^88 + 7*x^87 + 619*x^86 + 519*x^85 + 1020*x^84 +
225*x^83 + 872*x^82 + 927*x^81 + 905*x^80 + 907*x^79 + 451*x^78 + 559*x^77 +
326*x^76 + 836*x^75 + 366*x^74 + 561*x^73 + 254*x^72 + 445*x^71 + 107*x^70 +
348*x^69 + 788*x^68 + 73*x^67 + 685*x^66 + 998*x^65 + 64*x^64 + 391*x^63 +
513*x^62 + 250*x^61 + 646*x^60 + 134*x^59 + 695*x^58 + 876*x^57 + 520*x^56 +
987*x^55 + 2*x^54 + 752*x^53 + 216*x^52 + 719*x^51 + 13*x^50 + 55*x^49 +
904*x^48 + 267*x^47 + 482*x^46 + 1000*x^45 + 398*x^44 + 57*x^43 + 16*x^42 +
53*x^41 + 695*x^40 + 724*x^39 + 505*x^38 + 746*x^37 + 303*x^36 + 161*x^35 +
854*x^34 + 901*x^33 + 980*x^32 + 906*x^31 + 660*x^30 + 584*x^29 + 178*x^28 +
245*x^27 + 53*x^26 + 229*x^25 + 773*x^24 + 373*x^23 + 410*x^22 + 54*x^21 +
294*x^20 + 265*x^19 + 181*x^18 + 643*x^17 + 95*x^16 + 771*x^15 + 930*x^14 +
548*x^13 + 492*x^12 + 776*x^11 + 251*x^10 + 495*x^9 + 348*x^8 + 771*x^7 +
592*x^6 + 517*x^5 + 159*x^4 + 952*x^3 + 298*x^2 + 464*x + 347))

Privada:  (b'\xd1\x97\xbf\xc0U\xa0\xc42\x19\xb8D\xff=
\xd6y\xb2;\xa9\x1d\xb6P\x9c\x8d\xfc%\xdd\xbc\xef\xd0p\x08', b"\xab\x9f\xbe\xb1N\
x81TpA\xca*\x7fQ\x1c\xbb\xb6'7\x07N\x1es}\x98\xeazv\x1a\xf1\xbf\x1bN", b"s\x9e\x
e3\xcb\xe5_^/\xaf\xf3\xc1\x8eR18\xd8L\x10\x0b\xbb\x95\xb4y\x90&<t#w'\xb4'y\xc6\x
17\xe0;\x06\xcf\xa9\xaf\xfbk\x01A<\xa6N", (x^255 + x^254 + x^252 + x^250 + x^249
+ x^243 + x^242 + x^231 + x^230 + x^229 + x^228 + x^226 + x^222 + x^218 + x^217
+ x^216 + x^215 + x^214 + x^212 + x^210 + x^209 + x^208 + x^205 + x^204 + x^202
+ x^201 + x^199 + x^198 + x^194 + x^193 + x^190 + x^188 + x^187 + x^183 + x^180
+ x^179 + x^176 + x^175 + x^172 + x^169 + x^167 + x^163 + x^161 + x^158 + x^155
+ x^151 + x^150 + x^146 + x^145 + x^144 + x^143 + x^142 + x^141 + x^139 + x^137
+ x^135 + x^134 + x^128 + x^127 + x^126 + x^123 + x^122 + x^121 + x^120 + x^118
+ x^114 + x^111 + x^109 + x^108 + x^107 + x^105 + x^104 + x^102 + x^99 + x^98 +
x^97 + x^94 + x^93 + x^89 + x^88 + x^86 + x^84 + x^83 + x^79 + x^77 + x^76 +
x^75 + x^74 + x^73 + x^71 + x^69 + x^62 + x^61 + x^60 + x^58 + x^57 + x^53 +
x^52 + x^50 + x^48 + x^46 + x^45 + x^43 + x^41 + x^40 + x^37 + x^36 + x^35 +
x^32 + x^31 + x^27 + x^26 + x^24 + x^22 + x^20 + x^17 + x^13 + x^10 + x^8 + x^7
+ x^2 + x, x^255 + x^251 + x^249 + x^242 + x^239 + x^237 + x^236 + x^235 + x^234
+ x^233 + x^231 + x^230 + x^227 + x^226 + x^225 + x^224 + x^223 + x^217 + x^216
+ x^215 + x^214 + x^211 + x^210 + x^209 + x^205 + x^201 + x^200 + x^199 + x^197
+ x^193 + x^187 + x^186 + x^184 + x^181 + x^180 + x^179 + x^178 + x^177 + x^175
+ x^172 + x^171 + x^170 + x^166 + x^165 + x^164 + x^163 + x^161 + x^160 + x^159
+ x^158 + x^157 + x^152 + x^151 + x^147 + x^146 + x^141 + x^138 + x^136 + x^135
+ x^134 + x^132 + x^131 + x^129 + x^128 + x^125 + x^123 + x^122 + x^121 + x^116
```

```
+ x^115 + x^113 + x^112 + x^111 + x^110 + x^109 + x^107 + x^106 + x^105 + x^101
+ x^100 + x^98 + x^97 + x^96 + x^95 + x^93 + x^92 + x^91 + x^88 + x^87 + x^85 +
x^84 + x^82 + x^81 + x^80 + x^74 + x^73 + x^72 + x^67 + x^66 + x^63 + x^59 +
x^56 + x^55 + x^54 + x^52 + x^51 + x^50 + x^49 + x^46 + x^44 + x^42 + x^41 +
x^40 + x^35 + x^32 + x^27 + x^25 + x^24 + x^21 + x^20 + x^18 + x^16 + x^15 +
x^14 + x^13 + x^12 + x^11 + x^9 + x^7 + x^6 + x^2 + x, x^255 + x^254 + x^250 +
x^249 + x^248 + x^245 + x^244 + x^243 + x^242 + x^241 + x^240 + x^238 + x^237 +
x^235 + x^232 + x^231 + x^230 + x^227 + x^225 + x^223 + x^219 + x^216 + x^215 +
x^213 + x^207 + x^206 + x^204 + x^201 + x^199 + x^198 + x^192 + x^191 + x^189 +
x^188 + x^187 + x^185 + x^183 + x^181 + x^180 + x^177 + x^176 + x^173 + x^171 +
x^169 + x^166 + x^164 + x^163 + x^159 + x^156 + x^155 + x^154 + x^152 + x^151 +
x^148 + x^147 + x^146 + x^141 + x^140 + x^137 + x^135 + x^132 + x^130 + x^129 +
x^126 + x^125 + x^124 + x^121 + x^120 + x^118 + x^114 + x^113 + x^111 + x^108 +
x^104 + x^101 + x^100 + x^94 + x^92 + x^91 + x^90 + x^84 + x^83 + x^82 + x^79 +
x^71 + x^70 + x^69 + x^66 + x^65 + x^62 + x^57 + x^55 + x^54 + x^53 + x^52 +
x^50 + x^48 + x^47 + x^46 + x^43 + x^41 + x^39 + x^37 + x^36 + x^35 + x^33 +
x^32 + x^31 + x^28 + x^26 + x^24 + x^23 + x^22 + x^21 + x^19 + x^17 + x^16 +
x^15 + x^14 + x^11 + x^7 + x^6 + x^5 + x^4 + x^2 + x, x^254 + x^252 + x^251 +
x^250 + x^249 + x^248 + x^245 + x^243 + x^241 + x^240 + x^239 + x^237 + x^234 +
x^233 + x^231 + x^229 + x^228 + x^225 + x^224 + x^222 + x^221 + x^220 + x^215 +
x^214 + x^213 + x^212 + x^210 + x^209 + x^207 + x^206 + x^203 + x^202 + x^197 +
x^196 + x^195 + x^194 + x^192 + x^191 + x^190 + x^188 + x^187 + x^183 + x^182 +
x^179 + x^177 + x^172 + x^171 + x^164 + x^163 + x^158 + x^153 + x^152 + x^150 +
x^148 + x^147 + x^145 + x^143 + x^142 + x^141 + x^140 + x^139 + x^138 + x^136 +
x^134 + x^132 + x^129 + x^127 + x^125 + x^123 + x^122 + x^121 + x^120 + x^117 +
x^116 + x^114 + x^113 + x^111 + x^108 + x^106 + x^105 + x^104 + x^103 + x^102 +
x^100 + x^98 + x^97 + x^96 + x^95 + x^94 + x^93 + x^92 + x^91 + x^85 + x^84 +
x^83 + x^82 + x^79 + x^76 + x^75 + x^72 + x^70 + x^69 + x^62 + x^61 + x^60 +
x^57 + x^54 + x^52 + x^51 + x^50 + x^47 + x^46 + x^45 + x^44 + x^42 + x^39 +
x^36 + x^34 + x^33 + x^32 + x^31 + x^30 + x^21 + x^18 + x^14 + x^13 + x^12 +
x^11 + x^10 + x^9 + x^8 + x^5 + x^3 + 1, x^253 + x^252 + x^251 + x^250 + x^246 +
x^245 + x^244 + x^242 + x^239 + x^237 + x^235 + x^233 + x^231 + x^226 + x^225 +
x^222 + x^220 + x^215 + x^214 + x^213 + x^211 + x^210 + x^208 + x^207 + x^206 +
x^202 + x^200 + x^199 + x^198 + x^196 + x^194 + x^193 + x^192 + x^190 + x^189 +
x^188 + x^187 + x^186 + x^184 + x^183 + x^181 + x^175 + x^174 + x^173 + x^171 +
x^168 + x^162 + x^161 + x^159 + x^156 + x^154 + x^151 + x^148 + x^146 + x^144 +
x^142 + x^141 + x^139 + x^138 + x^136 + x^131 + x^128 + x^126 + x^125 + x^124 +
x^121 + x^117 + x^115 + x^114 + x^110 + x^108 + x^106 + x^105 + x^103 + x^100 +
x^99 + x^97 + x^96 + x^95 + x^94 + x^92 + x^88 + x^87 + x^85 + x^83 + x^82 +
x^81 + x^80 + x^79 + x^78 + x^77 + x^76 + x^72 + x^70 + x^69 + x^64 + x^60 +
x^59 + x^53 + x^51 + x^49 + x^48 + x^47 + x^45 + x^44 + x^43 + x^39 + x^35 +
x^33 + x^31 + x^30 + x^29 + x^27 + x^25 + x^24 + x^22 + x^21 + x^20 + x^19 +
x^18 + x^17 + x^15 + x^14 + x^13 + x^12 + x^11 + x^10 + x^9 + x^8 + x^6 + x^5 +
1, x^255 + x^252 + x^251 + x^250 + x^248 + x^246 + x^245 + x^241 + x^240 + x^237
+ x^235 + x^232 + x^231 + x^230 + x^229 + x^227 + x^225 + x^224 + x^218 + x^217
+ x^216 + x^215 + x^214 + x^212 + x^210 + x^207 + x^206 + x^205 + x^202 + x^199
+ x^198 + x^197 + x^194 + x^192 + x^189 + x^185 + x^182 + x^180 + x^179 + x^176
+ x^171 + x^170 + x^168 + x^167 + x^162 + x^160 + x^158 + x^157 + x^156 + x^155
```

```
+ x^153 + x^152 + x^149 + x^148 + x^146 + x^144 + x^143 + x^142 + x^141 + x^137
+ x^136 + x^135 + x^134 + x^132 + x^129 + x^128 + x^127 + x^126 + x^124 + x^123
+ x^116 + x^115 + x^112 + x^110 + x^103 + x^102 + x^99 + x^88 + x^87 + x^85 +
x^81 + x^77 + x^76 + x^74 + x^72 + x^66 + x^63 + x^62 + x^56 + x^53 + x^50 +
x^47 + x^45 + x^41 + x^39 + x^37 + x^34 + x^33 + x^31 + x^30 + x^29 + x^22 +
x^18 + x^14 + x^12 + x^7 + x^4 + x^3 + x^2 + 1, x^253 + x^252 + x^251 + x^250 +
x^247 + x^242 + x^241 + x^239 + x^238 + x^236 + x^235 + x^233 + x^230 + x^229 +
x^228 + x^226 + x^225 + x^223 + x^217 + x^216 + x^214 + x^209 + x^207 + x^206 +
x^204 + x^201 + x^200 + x^195 + x^191 + x^190 + x^189 + x^186 + x^185 + x^183 +
x^182 + x^180 + x^177 + x^176 + x^175 + x^174 + x^173 + x^172 + x^171 + x^170 +
x^168 + x^167 + x^166 + x^165 + x^164 + x^158 + x^156 + x^154 + x^150 + x^149 +
x^146 + x^144 + x^143 + x^142 + x^141 + x^139 + x^138 + x^136 + x^134 + x^133 +
x^131 + x^129 + x^125 + x^123 + x^122 + x^117 + x^115 + x^113 + x^112 + x^105 +
x^102 + x^101 + x^99 + x^98 + x^96 + x^95 + x^94 + x^88 + x^86 + x^82 + x^81 +
x^78 + x^77 + x^76 + x^74 + x^73 + x^72 + x^71 + x^70 + x^69 + x^68 + x^67 +
x^66 + x^64 + x^62 + x^57 + x^53 + x^46 + x^45 + x^41 + x^37 + x^36 + x^34 +
x^33 + x^31 + x^28 + x^24 + x^22 + x^19 + x^14 + x^12 + x^8 + x^3 + x, x^255 +
x^254 + x^253 + x^252 + x^251 + x^249 + x^248 + x^244 + x^243 + x^242 + x^241 +
x^240 + x^239 + x^236 + x^235 + x^234 + x^230 + x^229 + x^227 + x^223 + x^222 +
x^221 + x^220 + x^218 + x^216 + x^214 + x^213 + x^212 + x^210 + x^207 + x^204 +
x^203 + x^201 + x^200 + x^199 + x^196 + x^195 + x^194 + x^193 + x^192 + x^191 +
x^190 + x^189 + x^188 + x^187 + x^186 + x^184 + x^181 + x^179 + x^176 + x^175 +
x^173 + x^170 + x^168 + x^165 + x^164 + x^163 + x^162 + x^161 + x^160 + x^155 +
x^153 + x^152 + x^151 + x^150 + x^149 + x^145 + x^141 + x^140 + x^139 + x^133 +
x^132 + x^129 + x^127 + x^126 + x^124 + x^123 + x^116 + x^115 + x^114 + x^113 +
x^109 + x^108 + x^104 + x^103 + x^102 + x^99 + x^94 + x^90 + x^89 + x^88 + x^86
+ x^85 + x^84 + x^83 + x^82 + x^81 + x^79 + x^78 + x^74 + x^73 + x^72 + x^70 +
x^66 + x^65 + x^62 + x^59 + x^58 + x^56 + x^53 + x^52 + x^51 + x^50 + x^48 +
x^47 + x^46 + x^44 + x^37 + x^35 + x^33 + x^29 + x^27 + x^26 + x^25 + x^24 +
x^22 + x^20 + x^19 + x^13 + x^12 + x^10 + x^8 + x^4 + x^3 + x^2 + x), (x^255 +
x^254 + x^252 + x^248 + x^247 + x^244 + x^243 + x^241 + x^240 + x^239 + x^235 +
x^233 + x^230 + x^228 + x^227 + x^224 + x^222 + x^221 + x^220 + x^219 + x^217 +
x^215 + x^214 + x^207 + x^206 + x^205 + x^204 + x^200 + x^199 + x^197 + x^195 +
x^193 + x^190 + x^185 + x^184 + x^181 + x^176 + x^175 + x^171 + x^168 + x^167 +
x^166 + x^163 + x^161 + x^160 + x^159 + x^157 + x^156 + x^153 + x^152 + x^150 +
x^149 + x^147 + x^144 + x^143 + x^142 + x^141 + x^134 + x^132 + x^131 + x^130 +
x^129 + x^128 + x^127 + x^124 + x^123 + x^121 + x^120 + x^119 + x^118 + x^117 +
x^116 + x^114 + x^113 + x^112 + x^110 + x^108 + x^103 + x^101 + x^100 + x^97 +
x^96 + x^93 + x^92 + x^90 + x^88 + x^81 + x^76 + x^75 + x^72 + x^68 + x^66 +
x^64 + x^61 + x^60 + x^58 + x^57 + x^55 + x^54 + x^53 + x^52 + x^51 + x^46 +
x^45 + x^43 + x^42 + x^40 + x^39 + x^37 + x^36 + x^35 + x^33 + x^32 + x^31 +
x^30 + x^28 + x^27 + x^24 + x^23 + x^22 + x^20 + x^16 + x^15 + x^14 + x^12 +
x^11 + x^10 + x^9 + x^8 + x^6 + x^5 + x^4 + x^2 + x + 1, x^254 + x^252 + x^251 +
x^250 + x^249 + x^247 + x^244 + x^243 + x^240 + x^238 + x^233 + x^231 + x^227 +
x^226 + x^225 + x^224 + x^223 + x^222 + x^220 + x^218 + x^217 + x^215 + x^213 +
x^211 + x^210 + x^208 + x^207 + x^202 + x^200 + x^199 + x^198 + x^195 + x^194 +
x^192 + x^191 + x^190 + x^188 + x^187 + x^183 + x^181 + x^179 + x^178 + x^177 +
x^171 + x^165 + x^164 + x^163 + x^161 + x^160 + x^159 + x^158 + x^157 + x^154 +
```

```
x^150 + x^149 + x^147 + x^146 + x^145 + x^144 + x^143 + x^142 + x^139 + x^138 +
x^135 + x^132 + x^129 + x^128 + x^127 + x^126 + x^124 + x^123 + x^122 + x^120 +
x^118 + x^117 + x^116 + x^115 + x^114 + x^112 + x^107 + x^106 + x^104 + x^103 +
x^102 + x^101 + x^100 + x^99 + x^97 + x^95 + x^93 + x^92 + x^91 + x^90 + x^89 +
x^88 + x^87 + x^86 + x^85 + x^84 + x^83 + x^82 + x^81 + x^80 + x^75 + x^73 +
x^72 + x^69 + x^68 + x^66 + x^65 + x^64 + x^63 + x^61 + x^59 + x^58 + x^55 +
x^54 + x^50 + x^49 + x^45 + x^44 + x^42 + x^41 + x^40 + x^39 + x^37 + x^35 +
x^34 + x^32 + x^30 + x^29 + x^26 + x^25 + x^24 + x^23 + x^21 + x^20 + x^19 +
x^18 + x^17 + x^13 + x^12 + x^11 + x^10 + x^5 + x^4 + x^2 + 1, x^255 + x^252 +
x^245 + x^243 + x^240 + x^236 + x^235 + x^234 + x^231 + x^230 + x^227 + x^226 +
x^223 + x^221 + x^220 + x^217 + x^216 + x^214 + x^212 + x^211 + x^207 + x^205 +
x^196 + x^195 + x^194 + x^192 + x^191 + x^190 + x^188 + x^186 + x^179 + x^173 +
x^169 + x^166 + x^165 + x^163 + x^162 + x^161 + x^160 + x^158 + x^157 + x^156 +
x^153 + x^151 + x^149 + x^148 + x^146 + x^144 + x^142 + x^141 + x^140 + x^137 +
x^136 + x^135 + x^134 + x^133 + x^132 + x^131 + x^130 + x^127 + x^125 + x^124 +
x^123 + x^122 + x^117 + x^116 + x^115 + x^114 + x^112 + x^111 + x^108 + x^107 +
x^106 + x^104 + x^100 + x^99 + x^97 + x^95 + x^93 + x^92 + x^91 + x^90 + x^88 +
x^84 + x^83 + x^82 + x^77 + x^76 + x^75 + x^72 + x^69 + x^67 + x^66 + x^63 +
x^61 + x^60 + x^59 + x^58 + x^56 + x^53 + x^52 + x^51 + x^49 + x^46 + x^44 +
x^42 + x^41 + x^40 + x^39 + x^38 + x^33 + x^32 + x^30 + x^28 + x^27 + x^26 +
x^25 + x^22 + x^18 + x^17 + x^16 + x^15 + x^14 + x^12 + x^11 + x^10 + x^8 + x^7
+ x^6 + x^5 + x^2 + 1, x^255 + x^254 + x^252 + x^251 + x^250 + x^249 + x^248 +
x^246 + x^244 + x^243 + x^242 + x^241 + x^240 + x^239 + x^237 + x^235 + x^232 +
x^230 + x^228 + x^227 + x^224 + x^219 + x^216 + x^211 + x^210 + x^208 + x^203 +
x^202 + x^199 + x^198 + x^197 + x^195 + x^194 + x^190 + x^187 + x^186 + x^183 +
x^181 + x^178 + x^175 + x^174 + x^173 + x^172 + x^171 + x^170 + x^168 + x^167 +
x^165 + x^160 + x^159 + x^157 + x^151 + x^149 + x^146 + x^145 + x^144 + x^143 +
x^142 + x^141 + x^139 + x^137 + x^135 + x^134 + x^132 + x^131 + x^130 + x^129 +
x^128 + x^127 + x^123 + x^119 + x^118 + x^117 + x^114 + x^113 + x^111 + x^110 +
x^109 + x^108 + x^106 + x^104 + x^102 + x^98 + x^97 + x^93 + x^88 + x^87 + x^86
+ x^84 + x^82 + x^81 + x^73 + x^69 + x^68 + x^66 + x^65 + x^63 + x^61 + x^60 +
x^58 + x^57 + x^56 + x^55 + x^54 + x^53 + x^50 + x^48 + x^47 + x^45 + x^44 +
x^42 + x^38 + x^37 + x^31 + x^30 + x^28 + x^27 + x^26 + x^25 + x^23 + x^22 +
x^20 + x^17 + x^16 + x^12 + x^11 + x^9 + x^8 + x^5 + x^4 + x^3 + x^2, x^255 +
x^254 + x^248 + x^247 + x^244 + x^243 + x^241 + x^237 + x^236 + x^234 + x^231 +
x^230 + x^229 + x^226 + x^225 + x^224 + x^223 + x^221 + x^220 + x^216 + x^214 +
x^211 + x^209 + x^208 + x^207 + x^204 + x^202 + x^198 + x^197 + x^195 + x^194 +
x^193 + x^192 + x^191 + x^188 + x^184 + x^180 + x^179 + x^177 + x^176 + x^175 +
x^173 + x^171 + x^170 + x^169 + x^163 + x^162 + x^161 + x^159 + x^155 + x^154 +
x^153 + x^152 + x^150 + x^148 + x^146 + x^143 + x^142 + x^141 + x^139 + x^134 +
x^133 + x^131 + x^129 + x^125 + x^123 + x^122 + x^120 + x^119 + x^117 + x^115 +
x^114 + x^111 + x^110 + x^108 + x^107 + x^105 + x^104 + x^103 + x^101 + x^99 +
x^96 + x^95 + x^91 + x^90 + x^89 + x^86 + x^85 + x^84 + x^82 + x^79 + x^78 +
x^77 + x^75 + x^73 + x^72 + x^68 + x^67 + x^65 + x^63 + x^62 + x^61 + x^60 +
x^56 + x^55 + x^53 + x^51 + x^48 + x^46 + x^45 + x^44 + x^43 + x^42 + x^39 +
x^38 + x^37 + x^35 + x^34 + x^33 + x^31 + x^30 + x^26 + x^23 + x^22 + x^21 +
x^20 + x^18 + x^16 + x^10 + x^9 + x^6 + x^5 + x^4 + x^3 + x, x^255 + x^254 +
x^252 + x^251 + x^248 + x^247 + x^246 + x^243 + x^242 + x^239 + x^238 + x^237 +
```

```
x^236 + x^231 + x^230 + x^229 + x^228 + x^226 + x^223 + x^222 + x^221 + x^219 +
x^217 + x^212 + x^210 + x^208 + x^205 + x^204 + x^201 + x^198 + x^196 + x^194 +
x^193 + x^192 + x^190 + x^186 + x^184 + x^183 + x^181 + x^178 + x^174 + x^173 +
x^170 + x^168 + x^166 + x^163 + x^155 + x^153 + x^150 + x^148 + x^146 + x^145 +
x^144 + x^143 + x^142 + x^141 + x^139 + x^135 + x^134 + x^133 + x^132 + x^130 +
x^129 + x^128 + x^127 + x^126 + x^123 + x^121 + x^118 + x^113 + x^108 + x^105 +
x^104 + x^103 + x^101 + x^100 + x^98 + x^97 + x^95 + x^90 + x^89 + x^88 + x^84 +
x^83 + x^82 + x^80 + x^79 + x^78 + x^75 + x^74 + x^73 + x^72 + x^71 + x^70 +
x^69 + x^68 + x^65 + x^63 + x^62 + x^57 + x^56 + x^54 + x^53 + x^51 + x^50 +
x^49 + x^46 + x^45 + x^44 + x^43 + x^42 + x^41 + x^40 + x^39 + x^38 + x^36 +
x^33 + x^30 + x^29 + x^27 + x^26 + x^25 + x^24 + x^22 + x^21 + x^20 + x^18 +
x^17 + x^16 + x^14 + x^13 + x^11 + x^7 + x^5 + x^4 + x + 1, x^255 + x^253 +
x^252 + x^250 + x^249 + x^248 + x^245 + x^239 + x^236 + x^234 + x^233 + x^232 +
x^230 + x^229 + x^224 + x^223 + x^221 + x^220 + x^217 + x^216 + x^215 + x^211 +
x^206 + x^202 + x^201 + x^198 + x^194 + x^193 + x^190 + x^187 + x^184 + x^181 +
x^178 + x^176 + x^174 + x^173 + x^172 + x^171 + x^170 + x^166 + x^162 + x^161 +
x^159 + x^158 + x^157 + x^156 + x^153 + x^149 + x^147 + x^143 + x^139 + x^138 +
x^137 + x^135 + x^133 + x^132 + x^131 + x^130 + x^129 + x^127 + x^126 + x^124 +
x^122 + x^119 + x^117 + x^116 + x^115 + x^106 + x^105 + x^104 + x^101 + x^100 +
x^98 + x^97 + x^96 + x^95 + x^94 + x^93 + x^91 + x^88 + x^85 + x^84 + x^83 +
x^79 + x^76 + x^75 + x^74 + x^73 + x^71 + x^67 + x^64 + x^63 + x^61 + x^60 +
x^59 + x^58 + x^57 + x^56 + x^50 + x^49 + x^45 + x^44 + x^42 + x^39 + x^33 +
x^30 + x^27 + x^24 + x^23 + x^22 + x^21 + x^20 + x^19 + x^18 + x^15 + x^14 +
x^11 + x^10 + x^8 + x^3, x^255 + x^254 + x^253 + x^250 + x^249 + x^247 + x^245 +
x^244 + x^243 + x^242 + x^241 + x^240 + x^239 + x^237 + x^236 + x^235 + x^232 +
x^231 + x^230 + x^229 + x^226 + x^220 + x^219 + x^218 + x^217 + x^215 + x^214 +
x^213 + x^212 + x^211 + x^208 + x^207 + x^205 + x^204 + x^203 + x^201 + x^199 +
x^198 + x^197 + x^196 + x^194 + x^192 + x^191 + x^187 + x^185 + x^184 + x^183 +
x^182 + x^180 + x^179 + x^176 + x^174 + x^173 + x^172 + x^171 + x^170 + x^169 +
x^166 + x^163 + x^162 + x^160 + x^158 + x^156 + x^155 + x^154 + x^152 + x^148 +
x^145 + x^144 + x^143 + x^140 + x^134 + x^133 + x^131 + x^130 + x^128 + x^127 +
x^126 + x^124 + x^123 + x^122 + x^120 + x^119 + x^118 + x^116 + x^115 + x^109 +
x^105 + x^104 + x^103 + x^101 + x^100 + x^96 + x^95 + x^91 + x^90 + x^89 + x^88
+ x^87 + x^86 + x^85 + x^82 + x^81 + x^79 + x^78 + x^74 + x^73 + x^72 + x^70 +
x^69 + x^68 + x^67 + x^65 + x^62 + x^61 + x^60 + x^59 + x^58 + x^57 + x^53 +
x^49 + x^48 + x^47 + x^46 + x^45 + x^42 + x^38 + x^37 + x^36 + x^35 + x^30 +
x^25 + x^24 + x^22 + x^19 + x^16 + x^13 + x^12 + x^11 + x^9 + x^7 + x^4 + x^3 +
1, x^255 + x^254 + x^251 + x^250 + x^247 + x^244 + x^242 + x^241 + x^240 + x^236
+ x^234 + x^226 + x^225 + x^223 + x^219 + x^217 + x^216 + x^215 + x^213 + x^212
+ x^210 + x^209 + x^207 + x^205 + x^203 + x^201 + x^200 + x^199 + x^196 + x^194
+ x^193 + x^190 + x^189 + x^186 + x^185 + x^184 + x^181 + x^179 + x^178 + x^177
+ x^176 + x^174 + x^173 + x^171 + x^167 + x^165 + x^162 + x^160 + x^159 + x^158
+ x^157 + x^156 + x^155 + x^153 + x^152 + x^151 + x^150 + x^149 + x^147 + x^146
+ x^144 + x^143 + x^142 + x^141 + x^140 + x^139 + x^138 + x^135 + x^134 + x^130
+ x^129 + x^128 + x^127 + x^126 + x^125 + x^118 + x^114 + x^113 + x^111 + x^104
+ x^103 + x^100 + x^97 + x^96 + x^95 + x^93 + x^91 + x^89 + x^86 + x^85 + x^83 +
x^82 + x^81 + x^80 + x^78 + x^76 + x^75 + x^74 + x^73 + x^72 + x^70 + x^69 +
x^66 + x^60 + x^59 + x^55 + x^54 + x^52 + x^51 + x^50 + x^49 + x^47 + x^45 +
```

```
x^44 + x^42 + x^41 + x^39 + x^37 + x^35 + x^34 + x^33 + x^31 + x^29 + x^28 +
x^26 + x^22 + x^21 + x^20 + x^17 + x^16 + x^15 + x^13 + x^11 + x^10 + x^8 + x^6
+ x^3 + x^2 + 1), (790*x^255 - 530*x^254 + 1452*x^253 + 513*x^252 - 1790*x^251 +
3924*x^250 - 2362*x^249 - 1111*x^248 + 622*x^247 + 3987*x^246 - 1715*x^245 +
582*x^244 - 2743*x^243 - 2007*x^242 + 2177*x^241 - 123*x^240 - 1994*x^239 -
1812*x^238 - 3625*x^237 - 3613*x^236 + 2653*x^235 + 1615*x^234 + 2223*x^233 -
2918*x^232 - 3702*x^231 + 1897*x^230 - 1601*x^229 + 3740*x^228 + 2502*x^227 -
2139*x^226 - 2384*x^225 - 2427*x^224 - 369*x^223 + 3330*x^222 + 2338*x^221 -
3819*x^220 - 2593*x^219 + 2425*x^218 - 3759*x^217 + 1958*x^216 - 2305*x^215 +
1480*x^214 + 2489*x^213 + 226*x^212 + 3550*x^211 - 3366*x^210 - 3117*x^209 +
3904*x^208 - 1039*x^207 + 1646*x^206 - 703*x^205 - 1425*x^204 - 2071*x^203 +
110*x^202 - 1631*x^201 - 2740*x^200 - 2513*x^199 - 2088*x^198 - 680*x^197 -
3238*x^196 + 3395*x^195 + 1884*x^194 - 1759*x^193 + 1712*x^192 - 3498*x^191 -
1957*x^190 - 1178*x^189 - 1120*x^188 - 1234*x^187 + 613*x^186 + 3823*x^185 -
1870*x^184 - 1901*x^183 - 2159*x^182 - 2222*x^181 - 569*x^180 + 3111*x^179 -
2022*x^178 + 460*x^177 - 3491*x^176 - 3809*x^175 - 1442*x^174 - 2799*x^173 -
2412*x^172 - 2387*x^171 - 618*x^170 - 3611*x^169 - 1437*x^168 - 2221*x^167 +
2590*x^166 + 808*x^165 - 319*x^164 + 765*x^163 - 3308*x^162 + 3808*x^161 -
3827*x^160 + 1822*x^159 + 2924*x^158 + 2736*x^157 - 2944*x^156 + 35*x^155 +
450*x^154 + 2477*x^153 + 513*x^152 - 1773*x^151 - 989*x^150 + 2715*x^149 -
299*x^148 - 2640*x^147 - 238*x^146 + 2863*x^145 - 253*x^144 + 779*x^143 +
3492*x^142 + 3636*x^141 - 2770*x^140 - 1687*x^139 - 861*x^138 + 1316*x^137 -
889*x^136 - 2934*x^135 - 2621*x^134 + 701*x^133 - 1014*x^132 - 3904*x^131 +
3264*x^130 + 996*x^129 + 1906*x^128 + 1077*x^127 - 1377*x^126 + 1323*x^125 -
3866*x^124 - 532*x^123 - 3680*x^122 + 3383*x^121 - 851*x^120 - 2289*x^119 -
1853*x^118 - 2590*x^117 + 2751*x^116 + 1552*x^115 + 3396*x^114 + 731*x^113 +
2422*x^112 + 1261*x^111 + 3202*x^110 + 687*x^109 - 203*x^108 - 537*x^107 +
1393*x^106 + 516*x^105 - 2984*x^104 - 200*x^103 + 1087*x^102 - 2084*x^101 -
680*x^100 - 3058*x^99 - 3817*x^98 + 3484*x^97 + 1779*x^96 - 724*x^95 - 3402*x^94
+ 748*x^93 + 2363*x^92 + 1027*x^91 - 2602*x^90 + 3871*x^89 - 96*x^88 - 3128*x^87
+ 392*x^86 + 1171*x^85 + 3082*x^84 - 3216*x^83 - 3948*x^82 + 1522*x^81 +
1754*x^80 - 1434*x^79 + 605*x^78 - 2753*x^77 - 3663*x^76 + 3006*x^75 - 2755*x^74
- 1231*x^73 + 1702*x^72 + 537*x^71 + 1564*x^70 + 2609*x^69 - 1254*x^68 +
207*x^67 + 464*x^66 + 2692*x^65 + 21*x^64 + 854*x^63 + 517*x^62 + 2025*x^61 +
2368*x^60 - 3934*x^59 - 1246*x^58 + 1487*x^57 + 1368*x^56 - 1883*x^55 +
3843*x^54 - 129*x^53 - 2851*x^52 - 4039*x^51 - 1874*x^50 + 721*x^49 - 2854*x^48
- 92*x^47 + 1602*x^46 - 1809*x^45 - 2075*x^44 - 1475*x^43 - 2837*x^42 +
2393*x^41 + 1670*x^40 + 2976*x^39 + 652*x^38 + 3767*x^37 - 1924*x^36 - 3180*x^35
+ 452*x^34 - 3878*x^33 + 3412*x^32 - 816*x^31 + 2256*x^30 - 2797*x^29 +
2283*x^28 + 1442*x^27 + 1144*x^26 + 2611*x^25 - 1976*x^24 + 210*x^23 + 2453*x^22
- 3472*x^21 + 2437*x^20 + 2948*x^19 + 2736*x^18 + 3034*x^17 - 513*x^16 +
843*x^15 - 3352*x^14 + 1805*x^13 + 511*x^12 + 643*x^11 - 2769*x^10 + 3327*x^9 -
965*x^8 - 2043*x^7 - 4049*x^6 - 1292*x^5 - 3446*x^4 + 1306*x^3 + 2683*x^2 +
3412*x + 3312, 2107*x^255 + 3365*x^254 - 637*x^253 - 310*x^252 - 344*x^251 -
1524*x^250 + 982*x^249 + 899*x^248 - 2329*x^247 + 2341*x^246 - 13*x^245 -
2874*x^244 + 2340*x^243 + 900*x^242 + 3687*x^241 - 561*x^240 - 113*x^239 -
978*x^238 + 1728*x^237 + 3003*x^236 - 1437*x^235 - 345*x^234 + 535*x^233 +
1724*x^232 + 3874*x^231 + 3778*x^230 - 2241*x^229 - 2373*x^228 - 2183*x^227 +
```

```
2212*x^226 - 96*x^225 + 3802*x^224 - 2238*x^223 - 823*x^222 + 3214*x^221 -
2471*x^220 - 1294*x^219 + 2547*x^218 + 588*x^217 - 3746*x^216 - 2569*x^215 -
3720*x^214 + 1829*x^213 - 2255*x^212 + 3014*x^211 - 3377*x^210 + 3790*x^209 -
4004*x^208 + 484*x^207 + 1811*x^206 - 1780*x^205 + 1451*x^204 - 201*x^203 -
817*x^202 + 1573*x^201 + 1874*x^200 + 2960*x^199 - 290*x^198 + 21*x^197 -
3998*x^196 - 3693*x^195 + 1717*x^194 + 3161*x^193 - 1094*x^192 - 3346*x^191 +
1235*x^190 - 1819*x^189 + 1567*x^188 - 534*x^187 - 760*x^186 - 2551*x^185 -
1122*x^184 - 4050*x^183 + 226*x^182 - 1920*x^181 - 908*x^180 - 3674*x^179 -
117*x^178 - 3430*x^177 - 1186*x^176 - 3702*x^175 + 356*x^174 - 1856*x^173 +
1186*x^172 - 1730*x^171 - 142*x^170 - 545*x^169 + 4090*x^168 + 4046*x^167 -
2632*x^166 + 2200*x^165 + 65*x^164 + 3198*x^163 + 3582*x^162 + 329*x^161 +
3152*x^160 - 1395*x^159 - 1487*x^158 + 3921*x^157 - 1857*x^156 - 3644*x^155 -
3519*x^154 + 1570*x^153 - 303*x^152 - 1489*x^151 + 326*x^150 - 693*x^149 +
771*x^148 - 3663*x^147 - 3438*x^146 + 2676*x^145 + 1452*x^144 - 1101*x^143 +
897*x^142 + 2005*x^141 - 2853*x^140 + 2326*x^139 - 2108*x^138 - 2216*x^137 +
722*x^136 - 2490*x^135 + 1724*x^134 + 3581*x^133 + 448*x^132 - 1772*x^131 -
412*x^130 + 1366*x^129 + 3882*x^128 - 2646*x^127 + 2254*x^126 + 1726*x^125 -
948*x^124 + 605*x^123 + 246*x^122 + 1565*x^121 - 1667*x^120 - 770*x^119 -
3768*x^118 - 3918*x^117 + 3974*x^116 + 2426*x^115 - 1426*x^114 - 3209*x^113 -
3331*x^112 + 2157*x^111 + 3139*x^110 + 1789*x^109 + 1800*x^108 + 1413*x^107 -
459*x^106 + 3374*x^105 - 3759*x^104 - 879*x^103 + 2494*x^102 - 1211*x^101 -
2137*x^100 - 1565*x^99 - 1539*x^98 - 1683*x^97 - 1486*x^96 + 1554*x^95 +
3014*x^94 - 1170*x^93 + 2308*x^92 - 1478*x^91 + 871*x^90 - 2535*x^89 - 3215*x^88
+ 135*x^87 - 2494*x^86 + 3786*x^85 + 204*x^84 + 2519*x^83 - 2839*x^82 +
3778*x^81 + 1443*x^80 - 171*x^79 + 722*x^78 + 2852*x^77 - 2025*x^76 - 2631*x^75
+ 64*x^74 + 1398*x^73 - 1375*x^72 + 3841*x^71 - 1380*x^70 + 1095*x^69 -
1287*x^68 + 1523*x^67 - 4088*x^66 - 934*x^65 + 526*x^64 - 1734*x^63 - 1538*x^62
- 2516*x^61 - 2437*x^60 - 110*x^59 + 2026*x^58 + 1204*x^57 - 3589*x^56 +
3497*x^55 - 631*x^54 - 169*x^53 + 2388*x^52 - 3989*x^51 + 3000*x^50 + 1087*x^49
+ 1877*x^48 + 2750*x^47 - 1431*x^46 - 2584*x^45 - 3914*x^44 - 2268*x^43 +
2397*x^42 + 945*x^41 - 1164*x^40 + 2184*x^39 - 237*x^38 - 3797*x^37 - 2905*x^36
- 1549*x^35 + 899*x^34 - 1247*x^33 - 1869*x^32 + 2117*x^31 + 1603*x^30 +
1839*x^29 + 3260*x^28 + 568*x^27 + 3481*x^26 - 2451*x^25 + 2306*x^24 - 1456*x^23
+ 3707*x^22 + 3714*x^21 - 1631*x^20 - 1821*x^19 + 148*x^18 + 2497*x^17 +
368*x^16 - 395*x^15 - 2086*x^14 + 2082*x^13 - 2693*x^12 + 74*x^11 - 2815*x^10 +
2223*x^9 + 1430*x^8 + 2626*x^7 - 3434*x^6 - 2953*x^5 - 184*x^4 - 3687*x^3 +
2897*x^2 - 2045*x - 2374, 2524*x^255 + 1281*x^254 + 4024*x^253 + 984*x^252 +
3406*x^251 + 3794*x^250 - 413*x^249 + 1416*x^248 - 3043*x^247 + 1748*x^246 +
2457*x^245 + 3535*x^244 + 848*x^243 + 2831*x^242 + 3851*x^241 + 2708*x^240 +
69*x^239 - 360*x^238 + 2903*x^237 + 1028*x^236 + 1786*x^235 + 782*x^234 +
603*x^233 + 1599*x^232 + 1805*x^231 - 246*x^230 + 3613*x^229 + 244*x^228 +
2121*x^227 + 412*x^226 + 1812*x^225 - 3974*x^224 + 3281*x^223 + 3056*x^222 +
757*x^221 + 299*x^220 + 3329*x^219 - 2618*x^218 + 654*x^217 + 2234*x^216 -
2210*x^215 + 790*x^214 - 2829*x^213 + 1782*x^212 + 2156*x^211 - 720*x^210 -
2037*x^209 - 2259*x^208 + 809*x^207 + 2653*x^206 - 731*x^205 + 624*x^204 +
3772*x^203 + 3050*x^202 + 1509*x^201 + 1899*x^200 + 2990*x^199 - 1422*x^198 -
4081*x^197 - 2631*x^196 - 2503*x^195 - 981*x^194 + 2619*x^193 + 2185*x^192 -
4002*x^191 + 780*x^190 + 1321*x^189 - 3241*x^188 - 1199*x^187 - 1985*x^186 -
```

```
2886*x^185 - 4081*x^184 - 3814*x^183 + 715*x^182 + 1966*x^181 - 378*x^180 -
717*x^179 + 3905*x^178 + 1000*x^177 + 3409*x^176 - 2451*x^175 - 2798*x^174 +
3608*x^173 + 209*x^172 + 2622*x^171 + 3798*x^170 + 626*x^169 + 1892*x^168 -
1756*x^167 + 378*x^166 + 2932*x^165 + 3520*x^164 + 35*x^163 - 4063*x^162 +
2904*x^161 + 1026*x^160 - 3639*x^159 + 2649*x^158 + 1052*x^157 - 3018*x^156 +
1640*x^155 + 3502*x^154 - 3453*x^153 - 3441*x^152 + 13*x^151 + 1962*x^150 -
2755*x^149 - 2341*x^148 - 3392*x^147 + 180*x^146 - 3313*x^145 + 1420*x^144 -
3080*x^143 - 1444*x^142 - 587*x^141 + 2354*x^140 + 801*x^139 - 2097*x^138 -
1761*x^137 - 3049*x^136 - 994*x^135 + 3798*x^134 - 3504*x^133 + 1001*x^132 -
3063*x^131 + 455*x^130 - 2804*x^129 + 408*x^128 + 502*x^127 + 2500*x^126 +
1147*x^125 + 907*x^124 + 2421*x^123 - 2368*x^122 + 3684*x^121 - 2373*x^120 -
2141*x^119 + 3278*x^118 - 228*x^117 - 54*x^116 + 979*x^115 + 720*x^114 +
909*x^113 + 1913*x^112 - 413*x^111 - 2004*x^110 - 3023*x^109 - 1022*x^108 -
348*x^107 + 412*x^106 + 207*x^105 - 1199*x^104 - 864*x^103 - 2536*x^102 +
933*x^101 - 2876*x^100 + 3283*x^99 - 2302*x^98 + 3912*x^97 - 3363*x^96 +
4018*x^95 - 2891*x^94 + 2408*x^93 + 1119*x^92 + 796*x^91 - 3156*x^90 + 535*x^89
- 75*x^88 - 3568*x^87 + 315*x^86 - 2596*x^85 - 1839*x^84 - 3453*x^83 - 1129*x^82
- 2729*x^81 + 408*x^80 + 1828*x^79 - 3891*x^78 + 1942*x^77 - 3405*x^76 +
614*x^75 - 2019*x^74 + 151*x^73 - 810*x^72 - 1418*x^71 + 1674*x^70 - 1875*x^69 -
1738*x^68 - 1942*x^67 + 4001*x^66 - 3876*x^65 + 3446*x^64 - 3262*x^63 +
3192*x^62 - 3727*x^61 - 2857*x^60 - 1028*x^59 + 3313*x^58 - 1959*x^57 + 325*x^56
+ 2342*x^55 - 2897*x^54 + 116*x^53 + 314*x^52 + 2872*x^51 + 1932*x^50 +
3484*x^49 + 3423*x^48 - 3400*x^47 + 2796*x^46 + 690*x^45 - 1436*x^44 + 3905*x^43
+ 340*x^42 - 2303*x^41 + 3210*x^40 - 2665*x^39 + 445*x^38 + 2569*x^37 +
1337*x^36 - 2276*x^35 - 117*x^34 - 3661*x^33 - 2871*x^32 + 3088*x^31 + 1425*x^30
- 3035*x^29 + 2831*x^28 + 185*x^27 + 936*x^26 - 89*x^25 + 754*x^24 - 198*x^23 +
3956*x^22 + 243*x^21 - 753*x^20 + 1940*x^19 + 496*x^18 - 2392*x^17 - 2173*x^16 +
29*x^15 + 1683*x^14 - 3052*x^13 - 3727*x^12 - 2869*x^11 - 2992*x^10 - 455*x^9 -
2382*x^8 - 3364*x^7 - 3736*x^6 + 3350*x^5 - 1926*x^4 + 2835*x^3 - 1937*x^2 +
2629*x + 1041, 3001*x^255 + 2704*x^254 + 1087*x^253 + 3152*x^252 + 1936*x^251 -
3557*x^250 - 3234*x^249 + 1596*x^248 + 142*x^247 - 2631*x^246 - 1102*x^245 +
1807*x^244 + 228*x^243 - 1624*x^242 + 847*x^241 + 3654*x^240 + 1895*x^239 -
3487*x^238 + 1220*x^237 - 1007*x^236 + 3004*x^235 + 3287*x^234 + 3590*x^233 -
562*x^232 - 3025*x^231 - 2415*x^230 - 165*x^229 + 3785*x^228 - 702*x^227 -
3662*x^226 + 2916*x^225 - 2315*x^224 - 1060*x^223 - 3953*x^222 + 3684*x^221 -
2249*x^220 + 2124*x^219 + 1283*x^218 - 1537*x^217 - 363*x^216 - 3817*x^215 +
1515*x^214 - 925*x^213 + 2385*x^212 + 1000*x^211 + 2078*x^210 - 256*x^209 -
135*x^208 + 3593*x^207 - 336*x^206 - 1360*x^205 - 665*x^204 - 1739*x^203 +
1889*x^202 + 3064*x^201 - 430*x^200 - 3179*x^199 - 2390*x^198 - 2727*x^197 +
1297*x^196 - 419*x^195 - 3438*x^194 - 1646*x^193 + 1079*x^192 + 3450*x^191 -
1898*x^190 - 1925*x^189 + 487*x^188 + 1883*x^187 - 298*x^186 - 2100*x^185 -
1218*x^184 + 2323*x^183 + 3449*x^182 - 3874*x^181 + 2487*x^180 - 262*x^179 -
2887*x^178 - 508*x^177 - 222*x^176 - 2790*x^175 - 3049*x^174 + 3373*x^173 -
725*x^172 + 1839*x^171 - 1671*x^170 - 615*x^169 + 2070*x^168 + 4024*x^167 +
3552*x^166 + 2062*x^165 + 1840*x^164 + 2217*x^163 - 645*x^162 - 3243*x^161 -
1911*x^160 + 2160*x^159 + 1448*x^158 - 1882*x^157 + 2499*x^156 + 860*x^155 +
1727*x^154 - 2924*x^153 + 2888*x^152 - 1126*x^151 - 1494*x^150 - 2583*x^149 -
440*x^148 + 1400*x^147 - 1196*x^146 + 3915*x^145 + 1518*x^144 - 2328*x^143 -
```

```
1713*x^142 + 180*x^141 - 2907*x^140 + 754*x^139 - 3919*x^138 + 2073*x^137 +
515*x^136 + 1427*x^135 + 4088*x^134 + 1594*x^133 + 25*x^132 - 2383*x^131 -
2599*x^130 - 4082*x^129 + 1058*x^128 - 2293*x^127 - 3849*x^126 - 2876*x^125 -
3648*x^124 - 3200*x^123 + 2255*x^122 - 1759*x^121 + 3686*x^120 - 3073*x^119 -
1077*x^118 - 3104*x^117 + 2207*x^116 + 3805*x^115 - 335*x^114 - 1564*x^113 +
2798*x^112 + 2669*x^111 - 1081*x^110 - 2322*x^109 + 1420*x^108 + 2564*x^107 +
3143*x^106 + 408*x^105 - 2293*x^104 - 3606*x^103 + 3846*x^102 - 1819*x^101 +
142*x^100 - 1588*x^99 + 2490*x^98 - 135*x^97 + 315*x^96 + 1320*x^95 - 1732*x^94
+ 1287*x^93 + 2590*x^92 + 2105*x^91 - 2654*x^90 - 2159*x^89 + 3516*x^88 +
1959*x^87 + 2311*x^86 - 1735*x^85 + 1615*x^84 + 3293*x^83 + 1589*x^82 +
1544*x^81 + 238*x^80 + 936*x^79 + 902*x^78 - 3685*x^77 + 3904*x^76 - 2196*x^75 +
106*x^74 + 3620*x^73 + 2284*x^72 - 2536*x^71 + 975*x^70 + 836*x^69 - 3265*x^68 +
3146*x^67 + 2651*x^66 + 2023*x^65 + 873*x^64 + 2111*x^63 - 2156*x^62 - 3374*x^61
+ 2705*x^60 + 1255*x^59 + 2548*x^58 + 118*x^57 + 1770*x^56 - 1052*x^55 +
3660*x^54 - 3777*x^53 + 2073*x^52 - 1907*x^51 - 1103*x^50 + 4062*x^49 +
3746*x^48 - 3399*x^47 - 2681*x^46 - 3227*x^45 - 2500*x^44 - 653*x^43 + 2288*x^42
+ 3889*x^41 + 3385*x^40 - 3595*x^39 + 2342*x^38 + 1897*x^37 + 3296*x^36 -
1889*x^35 - 1712*x^34 - 3567*x^33 + 2344*x^32 - 1334*x^31 + 724*x^30 + 1082*x^29
+ 3141*x^28 + 1322*x^27 - 705*x^26 - 2320*x^25 + 1978*x^24 + 3548*x^23 -
3334*x^22 + 1499*x^21 - 4081*x^20 + 2173*x^19 - 1542*x^18 + 4066*x^17 + 331*x^16
+ 2398*x^15 - 3577*x^14 - 536*x^13 + 3986*x^12 + 250*x^11 + 3786*x^10 + 1141*x^9
+ 2458*x^8 - 1425*x^7 + 623*x^6 - 1657*x^5 + 787*x^4 + 3247*x^3 - 1907*x^2 +
2031*x + 1670, 2134*x^255 + 645*x^254 - 3532*x^253 + 194*x^252 + 189*x^251 +
2071*x^250 - 697*x^249 - 2542*x^248 - 2145*x^247 + 562*x^246 - 2*x^245 +
3173*x^244 + 3378*x^243 + 1054*x^242 + 3417*x^241 + 2769*x^240 + 1744*x^239 +
230*x^238 - 3326*x^237 + 200*x^236 - 3165*x^235 + 539*x^234 + 3936*x^233 -
3721*x^232 - 2075*x^231 + 2120*x^230 - 2645*x^229 - 1206*x^228 - 633*x^227 +
3165*x^226 - 1556*x^225 + 1160*x^224 - 186*x^223 - 3905*x^222 - 1085*x^221 +
3412*x^220 + 1155*x^219 - 2040*x^218 + 190*x^217 + 997*x^216 + 123*x^215 -
3694*x^214 + 391*x^213 + 1287*x^212 - 1024*x^211 + 836*x^210 - 1838*x^209 -
1808*x^208 - 109*x^207 - 3206*x^206 - 2459*x^205 - 1144*x^204 - 203*x^203 +
3446*x^202 - 995*x^201 - 3614*x^200 - 2770*x^199 + 1947*x^198 - 3345*x^197 -
3177*x^196 - 1986*x^195 + 1034*x^194 + 3469*x^193 + 2238*x^192 - 2444*x^191 +
296*x^190 + 1053*x^189 + 3700*x^188 + 1924*x^187 + 1737*x^186 + 1710*x^185 +
733*x^184 - 1249*x^183 - 2634*x^182 - 3394*x^181 + 206*x^180 - 925*x^179 -
28*x^178 + 2818*x^177 - 2419*x^176 + 1859*x^175 + 2157*x^174 + 402*x^173 -
3697*x^172 - 3961*x^171 + 123*x^170 + 1021*x^169 + 389*x^168 + 479*x^167 -
1858*x^166 + 395*x^165 + 2910*x^164 - 3060*x^163 + 865*x^162 + 4087*x^161 -
1880*x^160 - 1834*x^159 - 3783*x^158 + 3631*x^157 - 2298*x^156 - 3767*x^155 +
1538*x^154 - 2003*x^153 + 2567*x^152 + 3533*x^151 + 2130*x^150 + 1141*x^149 +
2695*x^148 + 1216*x^147 - 3595*x^146 - 217*x^145 + 1506*x^144 - 3569*x^143 -
2002*x^142 + 2317*x^141 - 1893*x^140 - 1795*x^139 - 1473*x^138 + 622*x^137 +
3681*x^136 - 3509*x^135 + 1896*x^134 - 997*x^133 - 2467*x^132 - 3189*x^131 +
1685*x^130 - 3341*x^129 + 496*x^128 + 3133*x^127 + 1546*x^126 + 254*x^125 +
857*x^124 + 1670*x^123 + 3834*x^122 + 2057*x^121 - 1659*x^120 - 2034*x^119 -
1961*x^118 + 2882*x^117 + 3267*x^116 - 909*x^115 - 1453*x^114 - 146*x^113 +
1809*x^112 + 1505*x^111 + 2587*x^110 - 1317*x^109 - 160*x^108 + 4061*x^107 -
2217*x^106 + 2359*x^105 - 1052*x^104 + 2405*x^103 + 2117*x^102 + 1597*x^101 +
```

```
2400*x^100 - 4063*x^99 - 706*x^98 + 2040*x^97 + 3496*x^96 - 3585*x^95 + 214*x^94
- 3170*x^93 + 3574*x^92 + 2191*x^91 + 99*x^90 + 2351*x^89 + 2024*x^88 +
2529*x^87 + 946*x^86 + 2752*x^85 + 517*x^84 + 647*x^83 + 602*x^82 - 1862*x^81 -
1885*x^80 - 3243*x^79 + 421*x^78 - 61*x^77 - 2075*x^76 + 3786*x^75 - 94*x^74 -
112*x^73 - 3823*x^72 - 2660*x^71 + 2862*x^70 + 3635*x^69 - 3933*x^68 + 2418*x^67
+ 1797*x^66 - 1550*x^65 + 3433*x^64 + 2328*x^63 + 2522*x^62 - 1895*x^61 -
3240*x^60 - 2540*x^59 - 2310*x^58 - 1992*x^57 + 3214*x^56 + 3791*x^55 -
3960*x^54 - 2726*x^53 - 3811*x^52 - 1778*x^51 + 1019*x^50 + 774*x^49 + 2099*x^48
- 3653*x^47 + 2482*x^46 + 2122*x^45 - 2194*x^44 - 668*x^43 - 1495*x^42 -
2632*x^41 + 666*x^40 + 2383*x^39 + 1792*x^38 - 160*x^37 + 2864*x^36 - 3018*x^35
- 2341*x^34 - 3742*x^33 + 2484*x^32 - 3290*x^31 + 2945*x^30 - 3254*x^29 -
640*x^28 - 815*x^27 + 3560*x^26 - 2558*x^25 - 3179*x^24 + 3615*x^23 - 3707*x^22
+ 1213*x^21 + 720*x^20 + 473*x^19 + 3233*x^18 - 3908*x^17 + 3954*x^16 -
2828*x^15 - 927*x^14 + 1506*x^13 + 989*x^12 + 1252*x^11 - 1479*x^10 + 1865*x^9 -
3005*x^8 - 2136*x^7 - 1558*x^6 - 1657*x^5 + 1742*x^4 - 332*x^3 + 1931*x^2 +
648*x + 1598, 3155*x^255 - 2951*x^254 + 468*x^253 - 408*x^252 + 3425*x^251 -
1974*x^250 + 1477*x^249 + 3579*x^248 + 3503*x^247 + 676*x^246 - 2134*x^245 -
761*x^244 - 498*x^243 - 2952*x^242 + 826*x^241 - 1525*x^240 + 2128*x^239 +
2712*x^238 + 3301*x^237 + 2553*x^236 - 3509*x^235 + 415*x^234 - 3375*x^233 -
1787*x^232 + 1781*x^231 + 2592*x^230 - 3771*x^229 - 3443*x^228 + 2287*x^227 +
1891*x^226 + 3431*x^225 - 4067*x^224 - 3265*x^223 + 1083*x^222 - 1289*x^221 -
1698*x^220 + 3482*x^219 + 877*x^218 + 2920*x^217 - 2914*x^216 - 1659*x^215 +
2700*x^214 - 298*x^213 - 2580*x^212 + 1657*x^211 + 876*x^210 - 1132*x^209 +
1321*x^208 + 2634*x^207 - 4056*x^206 - 1427*x^205 - 570*x^204 + 2541*x^203 -
1345*x^202 - 2070*x^201 + 3280*x^200 + 2480*x^199 + 3166*x^198 - 476*x^197 -
2027*x^196 + 2727*x^195 + 727*x^194 - 1689*x^193 - 2755*x^192 - 1510*x^191 -
4003*x^190 - 2552*x^189 - 3126*x^188 - 2631*x^187 + 1728*x^186 - 659*x^185 +
3102*x^184 + 3642*x^183 + 3187*x^182 - 1358*x^181 + 1662*x^180 + 1463*x^179 -
1532*x^178 + 1151*x^177 - 3903*x^176 + 4036*x^175 + 594*x^174 + 2750*x^173 +
1372*x^172 - 1160*x^171 - 3435*x^170 - 3429*x^169 + 482*x^168 - 3667*x^167 +
3788*x^166 + 3487*x^165 + 457*x^164 + 515*x^163 + 450*x^162 + 2512*x^161 +
1276*x^160 - 71*x^159 - 1260*x^158 - 2074*x^157 + 975*x^156 + 3872*x^155 +
1853*x^154 + 3554*x^153 - 1618*x^152 - 1156*x^151 + 3955*x^150 - 2945*x^149 -
2987*x^148 + 2003*x^147 + 2681*x^146 + 3790*x^145 + 933*x^144 + 2864*x^143 +
1460*x^142 - 3497*x^141 + 1391*x^140 + 1144*x^139 - 2438*x^138 + 160*x^137 -
4049*x^136 + 446*x^135 + 3374*x^134 - 827*x^133 + 179*x^132 - 1019*x^131 -
748*x^130 + 978*x^129 - 3774*x^128 + 718*x^127 + 812*x^126 + 3213*x^125 +
2407*x^124 - 2099*x^123 - 3536*x^122 - 2237*x^121 - 554*x^120 + 629*x^119 +
1959*x^118 + 541*x^117 - 3782*x^116 + 3533*x^115 + 2599*x^114 - 2592*x^113 +
1483*x^112 + 1820*x^111 - 3581*x^110 - 3840*x^109 - 3703*x^108 - 3619*x^107 -
1477*x^106 - 1266*x^105 + 191*x^104 - 2808*x^103 + 2032*x^102 - 1987*x^101 -
3675*x^100 + 1814*x^99 + 2863*x^98 - 2226*x^97 - 1270*x^96 - 2738*x^95 -
3486*x^94 + 740*x^93 - 1915*x^92 - 3700*x^91 + 2347*x^90 - 1835*x^89 - 1547*x^88
- 4012*x^87 - 2409*x^86 - 1215*x^85 - 2349*x^84 + 1373*x^83 + 2951*x^82 -
3550*x^81 - 1849*x^80 - 2595*x^79 + 2751*x^78 - 2410*x^77 + 3379*x^76 -
2828*x^75 + 661*x^74 - 284*x^73 - 1920*x^72 - 3728*x^71 + 3647*x^70 + 1586*x^69
- 3670*x^68 + 567*x^67 + 2823*x^66 + 652*x^65 - 1252*x^64 - 3191*x^63 -
3253*x^62 + 3367*x^61 - 2061*x^60 - 3525*x^59 + 2171*x^58 - 2433*x^57 +
```

```
2095*x^56 + 3004*x^55 + 2349*x^54 - 814*x^53 + 3552*x^52 + 602*x^51 + 1630*x^50
+ 3745*x^49 + 2838*x^48 - 3227*x^47 + 1318*x^46 - 2769*x^45 + 1954*x^44 -
1759*x^43 + 2367*x^42 - 2232*x^41 + 791*x^40 + 2073*x^39 - 760*x^38 + 3731*x^37
- 3701*x^36 - 714*x^35 + 2823*x^34 - 486*x^33 + 4058*x^32 - 530*x^31 + 1356*x^30
+ 1206*x^29 + 326*x^28 + 1331*x^27 - 3460*x^26 + 812*x^25 + 1973*x^24 + 274*x^23
- 2001*x^22 - 3374*x^21 - 3356*x^20 + 2513*x^19 + 3475*x^18 + 2372*x^17 +
781*x^16 + 2971*x^15 + 2194*x^14 - 2095*x^13 + 3376*x^12 - 3309*x^11 - 3645*x^10
+ 1836*x^9 - 3465*x^8 + 3568*x^7 + 4070*x^6 + 1894*x^5 - 1568*x^4 - 2719*x^3 -
2448*x^2 + 1955*x - 737, -3025*x^255 + 2251*x^254 + 20*x^253 - 2135*x^252 +
1600*x^251 - 3757*x^250 - 1763*x^249 - 2342*x^248 + 2117*x^247 - 161*x^246 +
949*x^245 - 949*x^244 + 3312*x^243 - 2909*x^242 + 29*x^241 + 2309*x^240 +
3682*x^239 + 713*x^238 + 1142*x^237 - 3844*x^236 + 707*x^235 - 2847*x^234 -
1062*x^233 - 1205*x^232 + 346*x^231 - 3942*x^230 - 2103*x^229 - 730*x^228 -
658*x^227 + 2997*x^226 + 3324*x^225 - 2779*x^224 - 3945*x^223 - 1868*x^222 -
3028*x^221 + 2009*x^220 - 2274*x^219 + 400*x^218 - 2426*x^217 - 2346*x^216 -
1943*x^215 - 549*x^214 + 3326*x^213 + 1383*x^212 - 970*x^211 + 3034*x^210 +
2108*x^209 + 921*x^208 - 85*x^207 + 947*x^206 + 1215*x^205 - 3554*x^204 -
1302*x^203 - 1579*x^202 - 2700*x^201 + 1471*x^200 + 474*x^199 + 2585*x^198 -
1351*x^197 - 3890*x^196 + 732*x^195 - 3550*x^194 + 1494*x^193 + 1509*x^192 -
3839*x^191 - 3021*x^190 - 1135*x^189 + 2159*x^188 + 1534*x^187 + 487*x^186 -
269*x^185 + 3948*x^184 - 1889*x^183 + 364*x^182 - 2453*x^181 - 1052*x^180 +
405*x^179 + 3649*x^178 + 3267*x^177 - 3784*x^176 - 2228*x^175 + 3530*x^174 +
979*x^173 + 4040*x^172 + 2788*x^171 + 1833*x^170 + 2748*x^169 + 1386*x^168 -
1521*x^167 - 2020*x^166 - 1775*x^165 - 3465*x^164 + 3572*x^163 + 1173*x^162 -
341*x^161 - 59*x^160 - 965*x^159 + 108*x^158 - 2462*x^157 - 2412*x^156 -
3976*x^155 + 3169*x^154 - 2687*x^153 - 1001*x^152 + 3627*x^151 - 1233*x^150 +
1931*x^149 - 2087*x^148 - 2373*x^147 + 2569*x^146 + 1157*x^145 + 132*x^144 -
1635*x^143 - 2213*x^142 + 1888*x^141 + 2951*x^140 - 3532*x^139 + 1639*x^138 +
1068*x^137 - 27*x^136 + 632*x^135 + 593*x^134 + 2924*x^133 - 4023*x^132 +
1733*x^131 + 3399*x^130 - 640*x^129 + 441*x^128 + 3818*x^127 - 177*x^126 +
3062*x^125 - 2900*x^124 - 1983*x^123 - 1222*x^122 - 2909*x^121 + 3057*x^120 -
2450*x^119 + 2124*x^118 - 1812*x^117 + 2156*x^116 - 2781*x^115 - 2031*x^114 -
1244*x^113 - 2772*x^112 + 3870*x^111 - 2040*x^110 + 3436*x^109 + 2825*x^108 -
3313*x^107 + 3098*x^106 + 3171*x^105 + 4072*x^104 + 2622*x^103 + 181*x^102 +
2193*x^101 + 3448*x^100 - 3978*x^99 + 3175*x^98 - 2261*x^97 + 2224*x^96 +
3754*x^95 + 2367*x^94 - 722*x^93 - 1368*x^92 - 802*x^91 - 3209*x^90 + 571*x^89 +
2827*x^88 + 1876*x^87 + 170*x^86 - 3389*x^85 - 3048*x^84 + 2823*x^83 + 3137*x^82
+ 781*x^81 - 2005*x^80 - 653*x^79 + 433*x^78 - 1156*x^77 - 544*x^76 - 407*x^75 +
124*x^74 - 2272*x^73 + 1245*x^72 - 1387*x^71 - 2472*x^70 - 648*x^69 + 873*x^68 -
3215*x^67 + 1544*x^66 + 299*x^65 + 985*x^64 - 1190*x^63 - 956*x^62 - 3960*x^61 -
1656*x^60 + 1884*x^59 - 3013*x^58 - 3977*x^57 - 2755*x^56 + 1620*x^55 +
1444*x^54 - 4085*x^53 + 3597*x^52 - 2405*x^51 + 3708*x^50 + 374*x^49 - 2366*x^48
- 60*x^47 - 3324*x^46 + 3726*x^45 + 3151*x^44 - 637*x^43 - 3319*x^42 + 633*x^41
- 433*x^40 - 2405*x^39 + 213*x^38 + 394*x^37 + 3678*x^36 + 1107*x^35 - 2534*x^34
+ 1453*x^33 - 3808*x^32 + 1505*x^31 + 1976*x^30 + 3634*x^29 + 893*x^28 +
2775*x^27 + 3499*x^26 + 2559*x^25 + 3437*x^24 + 4077*x^23 + 95*x^22 + 2782*x^21
- 3421*x^20 - 2897*x^19 + 4037*x^18 - 2670*x^17 - 1292*x^16 + 3155*x^15 -
1958*x^14 + 2571*x^13 - 2740*x^12 + 4001*x^11 + 3408*x^10 - 4013*x^9 - 1405*x^8
```

```
- 140*x^7 - 1505*x^6 + 772*x^5 + 1389*x^4 - 1431*x^3 + 2027*x^2 + 3646*x - 715,
-1258*x^255 + 3863*x^254 - 2112*x^253 + 2221*x^252 + 174*x^251 + 3829*x^250 -
1793*x^249 + 3600*x^248 - 384*x^247 + 82*x^246 - 4028*x^245 - 3426*x^244 +
2708*x^243 - 3176*x^242 + 1552*x^241 + 2133*x^240 - 3990*x^239 - 1634*x^238 +
3210*x^237 - 2947*x^236 + 1137*x^235 + 2758*x^234 + 14*x^233 + 2754*x^232 +
1228*x^231 - 738*x^230 + 3902*x^229 + 3970*x^228 + 2415*x^227 + 3326*x^226 +
2691*x^225 - 541*x^224 - 2409*x^223 - 4067*x^222 + 3380*x^221 + 1606*x^220 +
830*x^219 - 2138*x^218 - 3738*x^217 + 1640*x^216 - 676*x^215 - 2019*x^214 -
2753*x^213 + 1875*x^212 - 727*x^211 - 3748*x^210 - 3815*x^209 + 1185*x^208 -
1931*x^207 - 1880*x^206 - 1922*x^205 - 2669*x^204 - 3112*x^203 - 2187*x^202 +
2508*x^201 - 661*x^200 + 1615*x^199 - 695*x^198 - 3283*x^197 - 872*x^196 -
3693*x^195 + 2204*x^194 + 1718*x^193 + 2975*x^192 - 1606*x^191 + 3905*x^190 +
1622*x^189 - 1747*x^188 + 3901*x^187 + 1294*x^186 - 479*x^185 - 3841*x^184 +
1925*x^183 - 3289*x^182 - 2718*x^181 + 2499*x^180 + 3108*x^179 + 698*x^178 +
2806*x^177 + 1645*x^176 - 1225*x^175 - 301*x^174 + 294*x^173 + 988*x^172 +
3923*x^171 - 2520*x^170 - 3483*x^169 + 3787*x^168 - 333*x^167 + 2995*x^166 -
928*x^165 - 482*x^164 - 3381*x^163 - 3665*x^162 - 2394*x^161 - 1315*x^160 -
951*x^159 - 3964*x^158 + 1521*x^157 + 3621*x^156 - 896*x^155 - 2717*x^154 +
2731*x^153 + 3943*x^152 - 1189*x^151 - 3985*x^150 + 582*x^149 - 3227*x^148 +
2215*x^147 - 3394*x^146 + 2721*x^145 + 1036*x^144 - 689*x^143 + 2817*x^142 -
1846*x^141 + 660*x^140 + 2847*x^139 - 3341*x^138 - 3676*x^137 + 1732*x^136 +
1970*x^135 - 2530*x^134 - 1155*x^133 - 1963*x^132 + 3642*x^131 - 112*x^130 -
3050*x^129 + 117*x^128 - 1543*x^127 + 2415*x^126 + 2111*x^125 + 3144*x^124 -
2910*x^123 + 1705*x^122 + 3114*x^121 + 2585*x^120 + 1643*x^119 + 2457*x^118 +
1651*x^117 + 3756*x^116 + 881*x^115 - 2752*x^114 - 3329*x^113 - 1427*x^112 -
18*x^111 - 830*x^110 - 3313*x^109 + 2983*x^108 - 1308*x^107 - 1469*x^106 -
1442*x^105 - 3333*x^104 - 3248*x^103 - 3485*x^102 + 1731*x^101 - 3667*x^100 +
2256*x^99 - 1878*x^98 + 1075*x^97 + 3206*x^96 - 125*x^95 - 2324*x^94 - 2761*x^93
+ 3414*x^92 - 3640*x^91 - 2498*x^90 - 810*x^89 + 901*x^88 - 2470*x^87 - 321*x^86
+ 2017*x^85 + 3868*x^84 + 4044*x^83 - 2532*x^82 - 3442*x^81 + 2636*x^80 +
794*x^79 + 1186*x^78 - 2128*x^77 - 3939*x^76 + 4035*x^75 + 458*x^74 + 2058*x^73
+ 3506*x^72 + 3853*x^71 - 1201*x^70 - 3728*x^69 + 441*x^68 + 2837*x^67 -
2665*x^66 + 3358*x^65 - 4051*x^64 - 2315*x^63 + 3101*x^62 + 2077*x^61 +
3469*x^60 - 1431*x^59 + 3618*x^58 + 1573*x^57 - 1268*x^56 + 808*x^55 - 2400*x^54
- 2629*x^53 + 3488*x^52 - 703*x^51 - 3529*x^50 + 2036*x^49 - 363*x^48 - 514*x^47
- 503*x^46 + 2836*x^45 + 2496*x^44 - 2799*x^43 + 3437*x^42 + 3294*x^41 -
2550*x^40 - 857*x^39 - 3857*x^38 - 445*x^37 + 3681*x^36 + 221*x^35 + 531*x^34 +
188*x^33 - 2749*x^32 - 826*x^31 + 1982*x^30 - 1026*x^29 + 2076*x^28 + 4055*x^27
+ 1274*x^26 + 1570*x^25 - 2965*x^24 - 1072*x^23 - 3483*x^22 - 1898*x^21 -
2728*x^20 + 475*x^19 + 2870*x^18 - 3113*x^17 + 2799*x^16 - 2845*x^15 - 1957*x^14
- 1531*x^13 - 1774*x^12 - 2536*x^11 - 2463*x^10 + 93*x^9 - 2079*x^8 - 1441*x^7 +
1078*x^6 + 2442*x^5 + 2623*x^4 + 3891*x^3 + 401*x^2 + 351*x + 2169, 159*x^255 +
2729*x^254 - 4053*x^253 + 1765*x^252 + 287*x^251 - 1412*x^250 + 1732*x^249 -
3405*x^248 - 1611*x^247 + 2685*x^246 - 3674*x^245 - 2679*x^244 + 2258*x^243 -
1259*x^242 - 3416*x^241 + 2634*x^240 + 3914*x^239 + 1862*x^238 + 928*x^237 +
3548*x^236 - 2898*x^235 - 386*x^234 - 4073*x^233 + 4086*x^232 - 2014*x^231 -
1732*x^230 - 3638*x^229 + 2350*x^228 - 779*x^227 + 3767*x^226 - 3799*x^225 +
1708*x^224 - 3827*x^223 - 1274*x^222 - 1480*x^221 - 2826*x^220 + 2078*x^219 -
```

```
3745*x^218 - 3918*x^217 - 1997*x^216 - 3316*x^215 + 3853*x^214 - 1416*x^213 -
1153*x^212 + 1926*x^211 + 3129*x^210 - 1036*x^209 - 1803*x^208 - 2430*x^207 +
1561*x^206 - 305*x^205 - 2790*x^204 + 1737*x^203 - 719*x^202 + 2870*x^201 +
2458*x^200 - 1063*x^199 + 2939*x^198 + 2203*x^197 + 1031*x^196 + 4039*x^195 -
134*x^194 + 426*x^193 + 228*x^192 - 237*x^191 + 1793*x^190 + 3522*x^189 +
2668*x^188 + 1956*x^187 + 3426*x^186 - 2358*x^185 - 1570*x^184 + 3557*x^183 -
3607*x^182 - 1076*x^181 + 2024*x^180 + 3643*x^179 + 2220*x^178 + 409*x^177 +
2368*x^176 + 1263*x^175 - 1586*x^174 - 1662*x^173 - 2891*x^172 - 3440*x^171 +
2173*x^170 + 607*x^169 - 611*x^168 - 1413*x^167 + 2126*x^166 + 1155*x^165 -
3435*x^164 + 3114*x^163 - 324*x^162 + 2227*x^161 - 1091*x^160 - 3389*x^159 +
2911*x^158 - 2574*x^157 - 1259*x^156 - 3627*x^155 + 3396*x^154 - 1134*x^153 +
192*x^152 - 2701*x^151 - 3190*x^150 + 164*x^149 - 1341*x^148 + 3707*x^147 +
2152*x^146 + 872*x^145 - 1590*x^144 - 1529*x^143 + 3354*x^142 - 2769*x^141 -
282*x^140 + 536*x^139 + 1433*x^138 + 3035*x^137 + 1919*x^136 - 1494*x^135 -
1508*x^134 + 3806*x^133 - 723*x^132 + 4067*x^131 + 1000*x^130 - 595*x^129 -
1706*x^128 + 490*x^127 - 2181*x^126 - 2082*x^125 - 4025*x^124 - 3278*x^123 +
1227*x^122 + 3414*x^121 - 1006*x^120 - 3925*x^119 - 1578*x^118 - 2101*x^117 +
2161*x^116 - 3344*x^115 + 1207*x^114 - 794*x^113 + 3986*x^112 - 373*x^111 +
3082*x^110 + 2526*x^109 - 2914*x^108 - 3567*x^107 - 2013*x^106 - 538*x^105 -
1338*x^104 + 2612*x^103 + 3281*x^102 + 595*x^101 + 672*x^100 + 812*x^99 -
494*x^98 + 1487*x^97 + 4007*x^96 + 2662*x^95 - 3356*x^94 + 530*x^93 - 901*x^92 +
525*x^91 - 4058*x^90 + 124*x^89 - 1306*x^88 + 1785*x^87 + 1212*x^86 + 204*x^85 -
3762*x^84 - 718*x^83 + 4008*x^82 + 1482*x^81 - 3149*x^80 + 664*x^79 - 3881*x^78
- 1121*x^77 + 1684*x^76 - 1027*x^75 + 559*x^74 + 1267*x^73 - 2147*x^72 +
2914*x^71 + 154*x^70 + 938*x^69 - 2291*x^68 + 2673*x^67 + 2796*x^66 - 3266*x^65
- 2105*x^64 + 160*x^63 + 3891*x^62 + 1314*x^61 - 2822*x^60 + 1729*x^59 -
4017*x^58 - 3143*x^57 + 1842*x^56 - 1549*x^55 - 926*x^54 - 2678*x^53 - 1268*x^52
- 1071*x^51 + 830*x^50 + 2582*x^49 + 913*x^48 - 3992*x^47 - 2297*x^46 -
2685*x^45 + 3067*x^44 - 3572*x^43 - 3292*x^42 + 3229*x^41 + 1141*x^40 -
1105*x^39 - 3897*x^38 + 2957*x^37 - 1696*x^36 + 380*x^35 + 2644*x^34 + 3216*x^33
- 3213*x^32 - 1130*x^31 + 2088*x^30 - 2349*x^29 + 3379*x^28 + 3984*x^27 -
414*x^26 + 3803*x^25 + 3126*x^24 - 1821*x^23 - 441*x^22 + 3888*x^21 - 2890*x^20
- 1680*x^19 + 68*x^18 + 3567*x^17 - 2195*x^16 + 3546*x^15 + 3194*x^14 -
2544*x^13 - 3874*x^12 - 3862*x^11 + 3644*x^10 + 272*x^9 - 3685*x^8 - 2823*x^7 +
3089*x^6 + 424*x^5 - 3252*x^4 - 2388*x^3 - 3017*x^2 - 2527*x + 3475))

Mensagem a assinar:  Mensagem aleatória

A assinar mensagem...
Assinatura gerada:
((342936*x^255 - 3647*x^254 + 309516*x^253 + 74807*x^252 - 101177*x^251 +
319577*x^250 + 248047*x^249 + 466443*x^248 - 178632*x^247 + 28420*x^246 -
487559*x^245 + 133882*x^244 + 494757*x^243 + 40946*x^242 - 237690*x^241 +
34116*x^240 - 472120*x^239 + 12632*x^238 - 486115*x^237 + 493251*x^236 -
431641*x^235 - 25456*x^234 - 383297*x^233 + 448007*x^232 - 205407*x^231 -
15762*x^230 - 424163*x^229 - 312469*x^228 - 418696*x^227 + 31859*x^226 -
28980*x^225 + 164541*x^224 + 410540*x^223 - 400782*x^222 - 228240*x^221 -
127925*x^220 + 379156*x^219 + 468123*x^218 - 10444*x^217 + 108655*x^216 -
```

```
432668*x^215 + 333061*x^214 + 382619*x^213 + 65026*x^212 - 433044*x^211 -
159347*x^210 + 216102*x^209 + 112078*x^208 - 95465*x^207 + 439075*x^206 -
417652*x^205 - 372480*x^204 - 241658*x^203 - 69392*x^202 - 164659*x^201 +
418367*x^200 + 412357*x^199 + 440091*x^198 + 450461*x^197 - 43004*x^196 -
516585*x^195 + 212417*x^194 - 12531*x^193 + 223805*x^192 + 26264*x^191 -
118269*x^190 + 268580*x^189 + 131719*x^188 - 2079*x^187 - 328960*x^186 -
167314*x^185 - 341722*x^184 - 197425*x^183 + 145164*x^182 + 470713*x^181 -
365740*x^180 - 313744*x^179 - 320723*x^178 - 81186*x^177 + 383058*x^176 +
298957*x^175 - 95150*x^174 - 185964*x^173 + 461976*x^172 - 62173*x^171 +
342941*x^170 - 3640*x^169 + 309507*x^168 + 74806*x^167 - 101173*x^166 +
319576*x^165 + 248062*x^164 + 466440*x^163 - 178630*x^162 + 28418*x^161 -
487541*x^160 + 133881*x^159 + 494751*x^158 + 40955*x^157 - 237685*x^156 +
34113*x^155 - 472125*x^154 + 12633*x^153 - 486112*x^152 + 493258*x^151 -
431640*x^150 - 25444*x^149 - 383299*x^148 + 448006*x^147 - 205401*x^146 -
15761*x^145 - 424163*x^144 - 312464*x^143 - 418696*x^142 + 31856*x^141 -
28977*x^140 + 164544*x^139 + 410541*x^138 - 400792*x^137 - 228238*x^136 -
127927*x^135 + 379156*x^134 + 468122*x^133 - 10446*x^132 + 108661*x^131 -
432674*x^130 + 333054*x^129 + 382613*x^128 + 65023*x^127 - 433048*x^126 -
159342*x^125 + 216111*x^124 + 112080*x^123 - 95464*x^122 + 439070*x^121 -
417650*x^120 - 372469*x^119 - 241655*x^118 - 69390*x^117 - 164663*x^116 +
418358*x^115 + 412348*x^114 + 440094*x^113 + 450468*x^112 - 42988*x^111 -
516584*x^110 + 212422*x^109 - 12532*x^108 + 223801*x^107 + 26270*x^106 -
118270*x^105 + 268579*x^104 + 131722*x^103 - 2082*x^102 - 328968*x^101 -
167328*x^100 - 341723*x^99 - 197420*x^98 + 145163*x^97 + 470714*x^96 -
365731*x^95 - 313745*x^94 - 320733*x^93 - 81190*x^92 + 383060*x^91 + 298961*x^90
- 95153*x^89 - 185970*x^88 + 461981*x^87 - 62185*x^86 + 342941*x^85 - 3634*x^84
+ 309508*x^83 + 74809*x^82 - 101178*x^81 + 319581*x^80 + 248059*x^79 +
466444*x^78 - 178627*x^77 + 28423*x^76 - 487556*x^75 + 133888*x^74 + 494755*x^73
+ 40948*x^72 - 237686*x^71 + 34120*x^70 - 472120*x^69 + 12629*x^68 - 486112*x^67
+ 493262*x^66 - 431638*x^65 - 25451*x^64 - 383304*x^63 + 448005*x^62 -
205401*x^61 - 15753*x^60 - 424155*x^59 - 312469*x^58 - 418698*x^57 + 31862*x^56
- 28975*x^55 + 164539*x^54 + 410530*x^53 - 400781*x^52 - 228240*x^51 -
127925*x^50 + 379153*x^49 + 468120*x^48 - 10439*x^47 + 108659*x^46 - 432663*x^45
+ 333062*x^44 + 382610*x^43 + 65027*x^42 - 433050*x^41 - 159351*x^40 +
216105*x^39 + 112089*x^38 - 95468*x^37 + 439076*x^36 - 417655*x^35 - 372468*x^34
- 241652*x^33 - 69400*x^32 - 164655*x^31 + 418359*x^30 + 412359*x^29 +
440084*x^28 + 450462*x^27 - 43001*x^26 - 516583*x^25 + 212425*x^24 - 12535*x^23
+ 223801*x^22 + 26268*x^21 - 118268*x^20 + 268579*x^19 + 131719*x^18 - 2082*x^17
- 328960*x^16 - 167325*x^15 - 341720*x^14 - 197423*x^13 + 145171*x^12 +
470709*x^11 - 365736*x^10 - 313742*x^9 - 320726*x^8 - 81186*x^7 + 383053*x^6 +
298961*x^5 - 95152*x^4 - 185962*x^3 + 461975*x^2 - 62183*x + 342937,
315688*x^255 + 489302*x^254 + 152591*x^253 - 173800*x^252 - 163825*x^251 +
255350*x^250 + 423776*x^249 - 371540*x^248 + 391311*x^247 - 52150*x^246 +
176322*x^245 - 15917*x^244 - 370912*x^243 - 417853*x^242 - 209231*x^241 -
26637*x^240 - 339239*x^239 - 68507*x^238 - 156160*x^237 - 259530*x^236 +
172924*x^235 + 193800*x^234 - 42111*x^233 + 81032*x^232 - 345644*x^231 +
156058*x^230 + 75827*x^229 + 238650*x^228 + 376268*x^227 - 45666*x^226 +
497836*x^225 + 140527*x^224 - 155975*x^223 + 315665*x^222 - 492482*x^221 -
```

```
177801*x^220 + 139469*x^219 + 120355*x^218 - 43207*x^217 + 296593*x^216 -
491021*x^215 + 441245*x^214 + 273078*x^213 - 140694*x^212 + 344802*x^211 -
48567*x^210 - 156386*x^209 + 427518*x^208 + 19846*x^207 + 390464*x^206 -
361728*x^205 + 274521*x^204 - 441917*x^203 - 425666*x^202 + 473261*x^201 -
513809*x^200 - 130060*x^199 + 437782*x^198 + 182491*x^197 - 112463*x^196 +
294503*x^195 - 197978*x^194 + 154721*x^193 - 195813*x^192 + 27093*x^191 -
48725*x^190 - 277404*x^189 + 2007*x^188 - 291235*x^187 + 359622*x^186 -
143167*x^185 + 123112*x^184 - 513182*x^183 - 186514*x^182 + 294775*x^181 +
47220*x^180 - 207371*x^179 + 356018*x^178 + 159898*x^177 + 352728*x^176 +
362076*x^175 + 178019*x^174 + 431179*x^173 + 282209*x^172 - 118804*x^171 +
315692*x^170 + 489304*x^169 + 152601*x^168 - 173800*x^167 - 163833*x^166 +
255349*x^165 + 423772*x^164 - 371533*x^163 + 391319*x^162 - 52148*x^161 +
176324*x^160 - 15918*x^159 - 370910*x^158 - 417851*x^157 - 209220*x^156 -
26632*x^155 - 339230*x^154 - 68505*x^153 - 156167*x^152 - 259529*x^151 +
172929*x^150 + 193811*x^149 - 42097*x^148 + 81031*x^147 - 345640*x^146 +
156066*x^145 + 75833*x^144 + 238657*x^143 + 376274*x^142 - 45661*x^141 +
497837*x^140 + 140526*x^139 - 155981*x^138 + 315660*x^137 - 492477*x^136 -
177800*x^135 + 139479*x^134 + 120352*x^133 - 43201*x^132 + 296592*x^131 -
491021*x^130 + 441242*x^129 + 273080*x^128 - 140694*x^127 + 344797*x^126 -
48569*x^125 - 156389*x^124 + 427515*x^123 + 19843*x^122 + 390461*x^121 -
361720*x^120 + 274528*x^119 - 441918*x^118 - 425671*x^117 + 473256*x^116 -
513815*x^115 - 130065*x^114 + 437781*x^113 + 182486*x^112 - 112468*x^111 +
294499*x^110 - 197981*x^109 + 154725*x^108 - 195820*x^107 + 27099*x^106 -
48723*x^105 - 277396*x^104 + 2010*x^103 - 291239*x^102 + 359615*x^101 -
143174*x^100 + 123113*x^99 - 513185*x^98 - 186512*x^97 + 294779*x^96 +
47228*x^95 - 207359*x^94 + 356012*x^93 + 159896*x^92 + 352725*x^91 + 362083*x^90
+ 178015*x^89 + 431181*x^88 + 282200*x^87 - 118798*x^86 + 315685*x^85 +
489304*x^84 + 152601*x^83 - 173798*x^82 - 163823*x^81 + 255350*x^80 +
423781*x^79 - 371535*x^78 + 391317*x^77 - 52152*x^76 + 176320*x^75 - 15915*x^74
- 370903*x^73 - 417844*x^72 - 209221*x^71 - 26636*x^70 - 339226*x^69 -
68492*x^68 - 156166*x^67 - 259533*x^66 + 172935*x^65 + 193808*x^64 - 42102*x^63
+ 81026*x^62 - 345637*x^61 + 156060*x^60 + 75832*x^59 + 238658*x^58 +
376273*x^57 - 45656*x^56 + 497835*x^55 + 140534*x^54 - 155979*x^53 + 315655*x^52
- 492481*x^51 - 177795*x^50 + 139475*x^49 + 120347*x^48 - 43204*x^47 +
296586*x^46 - 491024*x^45 + 441237*x^44 + 273088*x^43 - 140700*x^42 +
344798*x^41 - 48559*x^40 - 156384*x^39 + 427510*x^38 + 19832*x^37 + 390467*x^36
- 361727*x^35 + 274520*x^34 - 441918*x^33 - 425660*x^32 + 473257*x^31 -
513812*x^30 - 130066*x^29 + 437774*x^28 + 182487*x^27 - 112463*x^26 +
294510*x^25 - 197983*x^24 + 154729*x^23 - 195810*x^22 + 27095*x^21 - 48729*x^20
- 277401*x^19 + 2013*x^18 - 291232*x^17 + 359615*x^16 - 143169*x^15 +
123105*x^14 - 513181*x^13 - 186514*x^12 + 294783*x^11 + 47225*x^10 - 207365*x^9
+ 356017*x^8 + 159887*x^7 + 352720*x^6 + 362076*x^5 + 178024*x^4 + 431182*x^3 +
282208*x^2 - 118806*x + 315690, 492104*x^255 + 29627*x^254 + 418287*x^253 +
50746*x^252 - 497049*x^251 - 501914*x^250 + 158071*x^249 - 193669*x^248 -
363556*x^247 + 430960*x^246 + 49701*x^245 - 187995*x^244 + 43360*x^243 -
340641*x^242 - 451510*x^241 - 231208*x^240 + 189171*x^239 + 153130*x^238 -
241238*x^237 - 305726*x^236 + 96606*x^235 + 505917*x^234 - 206585*x^233 +
81647*x^232 - 382195*x^231 - 386609*x^230 - 28058*x^229 + 264577*x^228 -
```

```
176895*x^227 - 425194*x^226 - 144833*x^225 - 426936*x^224 + 72986*x^223 -
85270*x^222 - 391579*x^221 + 388012*x^220 - 260447*x^219 + 132206*x^218 -
203751*x^217 + 281807*x^216 + 296907*x^215 - 244357*x^214 + 372180*x^213 -
322786*x^212 - 145713*x^211 + 48325*x^210 - 188212*x^209 + 361161*x^208 +
362007*x^207 - 276932*x^206 + 323311*x^205 - 392755*x^204 - 375340*x^203 -
246952*x^202 + 258431*x^201 + 460942*x^200 + 502386*x^199 - 44478*x^198 +
436163*x^197 - 37460*x^196 - 6642*x^195 + 263181*x^194 - 272744*x^193 +
449866*x^192 + 121971*x^191 + 322377*x^190 + 177080*x^189 + 256081*x^188 -
496894*x^187 - 87255*x^186 - 145375*x^185 + 337923*x^184 + 167185*x^183 -
400375*x^182 + 480294*x^181 - 42994*x^180 + 157490*x^179 + 136202*x^178 -
173686*x^177 + 369493*x^176 - 205415*x^175 - 275290*x^174 - 2023*x^173 -
257540*x^172 - 185290*x^171 + 492103*x^170 + 29628*x^169 + 418289*x^168 +
50737*x^167 - 497042*x^166 - 501912*x^165 + 158080*x^164 - 193670*x^163 -
363547*x^162 + 430962*x^161 + 49712*x^160 - 187994*x^159 + 43367*x^158 -
340634*x^157 - 451504*x^156 - 231209*x^155 + 189178*x^154 + 153135*x^153 -
241238*x^152 - 305725*x^151 + 96609*x^150 + 505932*x^149 - 206586*x^148 +
81656*x^147 - 382194*x^146 - 386601*x^145 - 28046*x^144 + 264576*x^143 -
176894*x^142 - 425209*x^141 - 144832*x^140 - 426927*x^139 + 72989*x^138 -
85275*x^137 - 391580*x^136 + 388017*x^135 - 260446*x^134 + 132202*x^133 -
203752*x^132 + 281806*x^131 + 296899*x^130 - 244365*x^129 + 372177*x^128 -
322791*x^127 - 145724*x^126 + 48329*x^125 - 188209*x^124 + 361165*x^123 +
362005*x^122 - 276935*x^121 + 323310*x^120 - 392765*x^119 - 375342*x^118 -
246948*x^117 + 258436*x^116 + 460934*x^115 + 502390*x^114 - 44480*x^113 +
436162*x^112 - 37461*x^111 - 6638*x^110 + 263189*x^109 - 272741*x^108 +
449866*x^107 + 121971*x^106 + 322372*x^105 + 177077*x^104 + 256077*x^103 -
496885*x^102 - 87261*x^101 - 145380*x^100 + 337932*x^99 + 167186*x^98 -
400375*x^97 + 480303*x^96 - 42994*x^95 + 157490*x^94 + 136205*x^93 - 173685*x^92
+ 369503*x^91 - 205429*x^90 - 275286*x^89 - 2019*x^88 - 257538*x^87 -
185301*x^86 + 492109*x^85 + 29630*x^84 + 418278*x^83 + 50748*x^82 - 497035*x^81
- 501907*x^80 + 158069*x^79 - 193671*x^78 - 363547*x^77 + 430958*x^76 +
49707*x^75 - 187993*x^74 + 43370*x^73 - 340638*x^72 - 451509*x^71 - 231200*x^70
+ 189170*x^69 + 153134*x^68 - 241232*x^67 - 305724*x^66 + 96598*x^65 +
505926*x^64 - 206577*x^63 + 81652*x^62 - 382191*x^61 - 386597*x^60 - 28047*x^59
+ 264570*x^58 - 176901*x^57 - 425200*x^56 - 144833*x^55 - 426933*x^54 +
72998*x^53 - 85268*x^52 - 391580*x^51 + 388017*x^50 - 260442*x^49 + 132196*x^48
- 203761*x^47 + 281803*x^46 + 296903*x^45 - 244368*x^44 + 372170*x^43 -
322785*x^42 - 145721*x^41 + 48327*x^40 - 188213*x^39 + 361170*x^38 + 362001*x^37
- 276940*x^36 + 323312*x^35 - 392762*x^34 - 375343*x^33 - 246952*x^32 +
258437*x^31 + 460934*x^30 + 502387*x^29 - 44484*x^28 + 436168*x^27 - 37466*x^26
- 6637*x^25 + 263189*x^24 - 272738*x^23 + 449869*x^22 + 121974*x^21 +
322379*x^20 + 177074*x^19 + 256079*x^18 - 496896*x^17 - 87254*x^16 - 145375*x^15
+ 337931*x^14 + 167185*x^13 - 400378*x^12 + 480297*x^11 - 42994*x^10 +
157497*x^9 + 136202*x^8 - 173685*x^7 + 369495*x^6 - 205418*x^5 - 275284*x^4 -
2024*x^3 - 257534*x^2 - 185303*x + 492109, 81346*x^255 + 120332*x^254 -
179334*x^253 - 276123*x^252 - 425943*x^251 + 11322*x^250 + 153870*x^249 -
57734*x^248 - 331488*x^247 + 153407*x^246 + 360860*x^245 + 344401*x^244 -
493869*x^243 + 217535*x^242 + 263213*x^241 + 422650*x^240 - 265244*x^239 +
265258*x^238 - 279837*x^237 + 446755*x^236 - 110340*x^235 - 109482*x^234 -
```

```
232953*x^233 - 155057*x^232 - 516978*x^231 + 471664*x^230 - 3141*x^229 +
108058*x^228 + 203256*x^227 + 133615*x^226 + 361353*x^225 - 363928*x^224 -
182769*x^223 - 226647*x^222 + 100045*x^221 - 110016*x^220 - 356987*x^219 +
477647*x^218 - 207751*x^217 - 386436*x^216 + 195690*x^215 + 322312*x^214 -
203646*x^213 - 164951*x^212 - 141022*x^211 + 24893*x^210 - 14364*x^209 -
18114*x^208 + 110933*x^207 - 452499*x^206 + 459516*x^205 - 220512*x^204 -
363391*x^203 - 509858*x^202 - 498822*x^201 - 63824*x^200 - 284735*x^199 +
344810*x^198 - 154267*x^197 - 144535*x^196 - 518259*x^195 + 85007*x^194 -
246226*x^193 + 408603*x^192 + 433062*x^191 - 12401*x^190 + 281927*x^189 +
501504*x^188 - 239662*x^187 - 404837*x^186 - 287770*x^185 - 496808*x^184 +
322118*x^183 + 343841*x^182 - 500126*x^181 - 255807*x^180 - 397684*x^179 +
87083*x^178 - 466619*x^177 + 336791*x^176 - 356467*x^175 + 362483*x^174 +
472498*x^173 + 511730*x^172 - 37142*x^171 + 81345*x^170 + 120331*x^169 -
179333*x^168 - 276131*x^167 - 425944*x^166 + 11332*x^165 + 153873*x^164 -
57730*x^163 - 331482*x^162 + 153411*x^161 + 360866*x^160 + 344405*x^159 -
493868*x^158 + 217529*x^157 + 263213*x^156 + 422663*x^155 - 265241*x^154 +
265264*x^153 - 279828*x^152 + 446760*x^151 - 110334*x^150 - 109480*x^149 -
232947*x^148 - 155052*x^147 - 516977*x^146 + 471661*x^145 - 3130*x^144 +
108059*x^143 + 203260*x^142 + 133619*x^141 + 361359*x^140 - 363932*x^139 -
182770*x^138 - 226656*x^137 + 100029*x^136 - 110010*x^135 - 356999*x^134 +
477651*x^133 - 207754*x^132 - 386436*x^131 + 195694*x^130 + 322311*x^129 -
203657*x^128 - 164965*x^127 - 141024*x^126 + 24880*x^125 - 14357*x^124 -
18116*x^123 + 110941*x^122 - 452502*x^121 + 459521*x^120 - 220509*x^119 -
363395*x^118 - 509857*x^117 - 498829*x^116 - 63821*x^115 - 284732*x^114 +
344812*x^113 - 154267*x^112 - 144527*x^111 - 518256*x^110 + 85010*x^109 -
246230*x^108 + 408597*x^107 + 433061*x^106 - 12401*x^105 + 281934*x^104 +
501511*x^103 - 239663*x^102 - 404832*x^101 - 287775*x^100 - 496807*x^99 +
322115*x^98 + 343840*x^97 - 500131*x^96 - 255810*x^95 - 397688*x^94 + 87085*x^93
- 466617*x^92 + 336786*x^91 - 356457*x^90 + 362476*x^89 + 472501*x^88 +
511721*x^87 - 37141*x^86 + 81340*x^85 + 120338*x^84 - 179331*x^83 - 276129*x^82
- 425931*x^81 + 11321*x^80 + 153876*x^79 - 57741*x^78 - 331484*x^77 +
153408*x^76 + 360871*x^75 + 344405*x^74 - 493861*x^73 + 217536*x^72 +
263206*x^71 + 422656*x^70 - 265245*x^69 + 265264*x^68 - 279821*x^67 +
446767*x^66 - 110334*x^65 - 109479*x^64 - 232946*x^63 - 155055*x^62 -
516975*x^61 + 471670*x^60 - 3129*x^59 + 108062*x^58 + 203255*x^57 + 133624*x^56
+ 361353*x^55 - 363927*x^54 - 182765*x^53 - 226645*x^52 + 100034*x^51 -
110019*x^50 - 356998*x^49 + 477643*x^48 - 207751*x^47 - 386442*x^46 +
195701*x^45 + 322311*x^44 - 203647*x^43 - 164961*x^42 - 141022*x^41 + 24876*x^40
- 14368*x^39 - 18114*x^38 + 110938*x^37 - 452506*x^36 + 459517*x^35 -
220512*x^34 - 363386*x^33 - 509856*x^32 - 498823*x^31 - 63822*x^30 - 284732*x^29
+ 344811*x^28 - 154269*x^27 - 144543*x^26 - 518269*x^25 + 85012*x^24 -
246232*x^23 + 408606*x^22 + 433058*x^21 - 12395*x^20 + 281933*x^19 + 501513*x^18
- 239660*x^17 - 404838*x^16 - 287763*x^15 - 496810*x^14 + 322121*x^13 +
343841*x^12 - 500130*x^11 - 255812*x^10 - 397683*x^9 + 87079*x^8 - 466617*x^7 +
336789*x^6 - 356466*x^5 + 362481*x^4 + 472498*x^3 + 511738*x^2 - 37141*x +
81342, -308086*x^255 - 394549*x^254 - 180155*x^253 - 23613*x^252 + 465332*x^251
- 99565*x^250 - 390375*x^249 + 148255*x^248 - 183807*x^247 - 437534*x^246 -
104303*x^245 + 264406*x^244 - 274038*x^243 + 434135*x^242 - 417841*x^241 +
```

```
391330*x^240 - 238728*x^239 - 325155*x^238 + 231089*x^237 - 522921*x^236 +
178646*x^235 - 316252*x^234 + 350167*x^233 - 326727*x^232 + 193070*x^231 -
450699*x^230 - 521271*x^229 - 314167*x^228 + 427486*x^227 + 114547*x^226 +
196075*x^225 - 400222*x^224 - 374342*x^223 - 287479*x^222 + 502767*x^221 +
445442*x^220 + 362147*x^219 + 300299*x^218 - 259757*x^217 + 136700*x^216 -
369922*x^215 + 57928*x^214 + 259165*x^213 - 234634*x^212 - 445133*x^211 -
144584*x^210 + 518048*x^209 - 33719*x^208 - 462736*x^207 + 136454*x^206 +
354699*x^205 - 363095*x^204 - 365897*x^203 + 387088*x^202 + 42328*x^201 +
207831*x^200 + 513923*x^199 - 23341*x^198 + 428991*x^197 - 203136*x^196 -
398742*x^195 + 280676*x^194 - 505694*x^193 - 295514*x^192 - 133238*x^191 +
175625*x^190 + 52276*x^189 - 498931*x^188 + 457712*x^187 + 402666*x^186 -
200177*x^185 + 32953*x^184 - 256558*x^183 + 63924*x^182 - 331620*x^181 +
187416*x^180 - 253533*x^179 - 499896*x^178 - 200802*x^177 + 85987*x^176 -
378866*x^175 + 349253*x^174 + 65306*x^173 + 171592*x^172 + 3141*x^171 -
308084*x^170 - 394541*x^169 - 180157*x^168 - 23610*x^167 + 465329*x^166 -
99556*x^165 - 390379*x^164 + 148261*x^163 - 183802*x^162 - 437523*x^161 -
104291*x^160 + 264417*x^159 - 274028*x^158 + 434136*x^157 - 417832*x^156 +
391331*x^155 - 238715*x^154 - 325153*x^153 + 231104*x^152 - 522918*x^151 +
178648*x^150 - 316254*x^149 + 350168*x^148 - 326721*x^147 + 193076*x^146 -
450695*x^145 - 521265*x^144 - 314163*x^143 + 427487*x^142 + 114551*x^141 +
196072*x^140 - 400219*x^139 - 374350*x^138 - 287488*x^137 + 502759*x^136 +
445438*x^135 + 362142*x^134 + 300298*x^133 - 259752*x^132 + 136697*x^131 -
369926*x^130 + 57935*x^129 + 259150*x^128 - 234634*x^127 - 445137*x^126 -
144589*x^125 + 518050*x^124 - 33716*x^123 - 462735*x^122 + 136455*x^121 +
354693*x^120 - 363094*x^119 - 365900*x^118 + 387086*x^117 + 42331*x^116 +
207837*x^115 + 513934*x^114 - 23350*x^113 + 429000*x^112 - 203138*x^111 -
398747*x^110 + 280684*x^109 - 505696*x^108 - 295508*x^107 - 133244*x^106 +
175628*x^105 + 52278*x^104 - 498927*x^103 + 457706*x^102 + 402673*x^101 -
200178*x^100 + 32950*x^99 - 256562*x^98 + 63926*x^97 - 331619*x^96 + 187412*x^95
- 253522*x^94 - 499901*x^93 - 200807*x^92 + 85987*x^91 - 378868*x^90 +
349250*x^89 + 65299*x^88 + 171594*x^87 + 3140*x^86 - 308090*x^85 - 394540*x^84 -
180152*x^83 - 23607*x^82 + 465325*x^81 - 99562*x^80 - 390375*x^79 + 148259*x^78
- 183797*x^77 - 437517*x^76 - 104296*x^75 + 264401*x^74 - 274032*x^73 +
434131*x^72 - 417837*x^71 + 391326*x^70 - 238709*x^69 - 325145*x^68 +
231106*x^67 - 522915*x^66 + 178645*x^65 - 316247*x^64 + 350169*x^63 -
326715*x^62 + 193073*x^61 - 450699*x^60 - 521275*x^59 - 314157*x^58 +
427484*x^57 + 114553*x^56 + 196084*x^55 - 400222*x^54 - 374340*x^53 -
287492*x^52 + 502764*x^51 + 445444*x^50 + 362148*x^49 + 300303*x^48 -
259755*x^47 + 136698*x^46 - 369926*x^45 + 57929*x^44 + 259156*x^43 - 234639*x^42
- 445134*x^41 - 144584*x^40 + 518045*x^39 - 33724*x^38 - 462732*x^37 +
136455*x^36 + 354697*x^35 - 363101*x^34 - 365900*x^33 + 387094*x^32 + 42328*x^31
+ 207837*x^30 + 513927*x^29 - 23343*x^28 + 428989*x^27 - 203140*x^26 -
398750*x^25 + 280672*x^24 - 505697*x^23 - 295511*x^22 - 133238*x^21 +
175635*x^20 + 52275*x^19 - 498919*x^18 + 457703*x^17 + 402680*x^16 - 200175*x^15
+ 32950*x^14 - 256577*x^13 + 63932*x^12 - 331622*x^11 + 187420*x^10 - 253530*x^9
- 499904*x^8 - 200799*x^7 + 85983*x^6 - 378861*x^5 + 349244*x^4 + 65311*x^3 +
171587*x^2 + 3140*x - 308091, -252003*x^255 + 397007*x^254 - 105868*x^253 +
448571*x^252 - 335118*x^251 - 411778*x^250 - 271992*x^249 - 494443*x^248 +
```

```
114139*x^247 + 137256*x^246 + 152102*x^245 + 195864*x^244 + 133053*x^243 +
225619*x^242 - 368860*x^241 - 123958*x^240 - 12894*x^239 - 260403*x^238 +
26437*x^237 + 497026*x^236 - 103475*x^235 - 229656*x^234 + 371683*x^233 -
121284*x^232 + 89011*x^231 - 5880*x^230 - 105374*x^229 - 510062*x^228 -
354847*x^227 + 49881*x^226 + 317013*x^225 - 160921*x^224 - 224156*x^223 -
17081*x^222 + 112903*x^221 - 483998*x^220 - 363984*x^219 - 70229*x^218 -
285437*x^217 + 506238*x^216 - 50099*x^215 + 266375*x^214 + 399456*x^213 -
136689*x^212 - 250671*x^211 - 287460*x^210 - 455929*x^209 - 66595*x^208 +
429737*x^207 - 29553*x^206 + 503096*x^205 + 293561*x^204 - 327236*x^203 -
165146*x^202 + 153172*x^201 + 315924*x^200 + 450266*x^199 + 255713*x^198 -
399612*x^197 - 287576*x^196 - 8532*x^195 - 318139*x^194 - 136944*x^193 -
222195*x^192 + 280328*x^191 - 199929*x^190 - 299442*x^189 + 318795*x^188 +
293512*x^187 - 445473*x^186 + 204318*x^185 + 71018*x^184 - 364535*x^183 +
425344*x^182 + 240621*x^181 + 17048*x^180 + 293732*x^179 - 14326*x^178 +
471900*x^177 + 401996*x^176 + 63832*x^175 + 138370*x^174 + 283143*x^173 -
414833*x^172 - 28175*x^171 - 251992*x^170 + 397007*x^169 - 105861*x^168 +
448562*x^167 - 335106*x^166 - 411782*x^165 - 271992*x^164 - 494448*x^163 +
114141*x^162 + 137264*x^161 + 152104*x^160 + 195868*x^159 + 133052*x^158 +
225620*x^157 - 368855*x^156 - 123948*x^155 - 12901*x^154 - 260405*x^153 +
26445*x^152 + 497024*x^151 - 103470*x^150 - 229666*x^149 + 371685*x^148 -
121276*x^147 + 89010*x^146 - 5872*x^145 - 105375*x^144 - 510056*x^143 -
354847*x^142 + 49895*x^141 + 317015*x^140 - 160915*x^139 - 224159*x^138 -
17084*x^137 + 112893*x^136 - 484003*x^135 - 363977*x^134 - 70228*x^133 -
285435*x^132 + 506231*x^131 - 50099*x^130 + 266371*x^129 + 399445*x^128 -
136690*x^127 - 250663*x^126 - 287447*x^125 - 455933*x^124 - 66601*x^123 +
429727*x^122 - 29552*x^121 + 503101*x^120 + 293565*x^119 - 327235*x^118 -
165150*x^117 + 153176*x^116 + 315916*x^115 + 450276*x^114 + 255711*x^113 -
399608*x^112 - 287576*x^111 - 8533*x^110 - 318135*x^109 - 136949*x^108 -
222185*x^107 + 280321*x^106 - 199921*x^105 - 299441*x^104 + 318792*x^103 +
293501*x^102 - 445475*x^101 + 204327*x^100 + 71024*x^99 - 364540*x^98 +
425341*x^97 + 240632*x^96 + 17046*x^95 + 293731*x^94 - 14320*x^93 + 471892*x^92
+ 401999*x^91 + 63820*x^90 + 138367*x^89 + 283134*x^88 - 414822*x^87 -
28159*x^86 - 251995*x^85 + 397012*x^84 - 105868*x^83 + 448564*x^82 - 335124*x^81
- 411787*x^80 - 271990*x^79 - 494442*x^78 + 114140*x^77 + 137259*x^76 +
152109*x^75 + 195867*x^74 + 133055*x^73 + 225621*x^72 - 368846*x^71 -
123959*x^70 - 12896*x^69 - 260408*x^68 + 26437*x^67 + 497022*x^66 - 103473*x^65
- 229653*x^64 + 371677*x^63 - 121278*x^62 + 89014*x^61 - 5863*x^60 - 105377*x^59
- 510059*x^58 - 354840*x^57 + 49885*x^56 + 317014*x^55 - 160921*x^54 -
224157*x^53 - 17091*x^52 + 112903*x^51 - 483998*x^50 - 363978*x^49 - 70229*x^48
- 285433*x^47 + 506239*x^46 - 50104*x^45 + 266376*x^44 + 399450*x^43 -
136697*x^42 - 250674*x^41 - 287457*x^40 - 455930*x^39 - 66597*x^38 + 429733*x^37
- 29550*x^36 + 503104*x^35 + 293556*x^34 - 327233*x^33 - 165152*x^32 +
153173*x^31 + 315917*x^30 + 450270*x^29 + 255708*x^28 - 399617*x^27 -
287580*x^26 - 8536*x^25 - 318126*x^24 - 136957*x^23 - 222181*x^22 + 280326*x^21
- 199928*x^20 - 299444*x^19 + 318793*x^18 + 293509*x^17 - 445480*x^16 +
204321*x^15 + 71027*x^14 - 364542*x^13 + 425342*x^12 + 240626*x^11 + 17047*x^10
+ 293730*x^9 - 14326*x^8 + 471892*x^7 + 402006*x^6 + 63825*x^5 + 138368*x^4 +
283143*x^3 - 414824*x^2 - 28160*x - 252000, 231729*x^255 + 148754*x^254 -
```

```
101024*x^253 + 519071*x^252 + 40480*x^251 - 353390*x^250 + 338074*x^249 +
462525*x^248 - 485289*x^247 + 221113*x^246 - 340316*x^245 + 354745*x^244 +
376886*x^243 - 413597*x^242 - 261453*x^241 - 495666*x^240 - 384827*x^239 +
458855*x^238 + 58497*x^237 + 423880*x^236 - 191746*x^235 + 213195*x^234 -
456426*x^233 - 429628*x^232 + 83943*x^231 - 385411*x^230 - 252974*x^229 +
189199*x^228 - 322531*x^227 + 178900*x^226 - 345383*x^225 - 22761*x^224 -
26436*x^223 + 121580*x^222 - 395489*x^221 - 113166*x^220 + 436400*x^219 -
443995*x^218 + 154770*x^217 + 29802*x^216 + 431628*x^215 - 361014*x^214 +
381469*x^213 - 189874*x^212 + 277569*x^211 + 272140*x^210 + 429220*x^209 +
254036*x^208 - 52473*x^207 + 86564*x^206 + 510473*x^205 - 408635*x^204 +
454987*x^203 - 73625*x^202 - 403171*x^201 - 510945*x^200 + 274151*x^199 +
446877*x^198 + 147161*x^197 + 335619*x^196 + 336957*x^195 - 184266*x^194 -
191298*x^193 + 520072*x^192 - 235403*x^191 + 86359*x^190 + 15783*x^189 +
381158*x^188 + 410167*x^187 + 500922*x^186 + 103789*x^185 - 192817*x^184 +
253106*x^183 - 276924*x^182 - 464206*x^181 - 128618*x^180 + 897*x^179 -
164346*x^178 + 358092*x^177 + 96316*x^176 - 171607*x^175 - 511751*x^174 -
152696*x^173 - 339475*x^172 - 423807*x^171 + 231727*x^170 + 148751*x^169 -
101023*x^168 + 519074*x^167 + 40484*x^166 - 353395*x^165 + 338073*x^164 +
462527*x^163 - 485276*x^162 + 221110*x^161 - 340315*x^160 + 354742*x^159 +
376888*x^158 - 413586*x^157 - 261450*x^156 - 495663*x^155 - 384827*x^154 +
458858*x^153 + 58503*x^152 + 423881*x^151 - 191753*x^150 + 213200*x^149 -
456421*x^148 - 429619*x^147 + 83944*x^146 - 385402*x^145 - 252969*x^144 +
189208*x^143 - 322528*x^142 + 178910*x^141 - 345377*x^140 - 22765*x^139 -
26440*x^138 + 121578*x^137 - 395494*x^136 - 113161*x^135 + 436400*x^134 -
443989*x^133 + 154771*x^132 + 29804*x^131 + 431628*x^130 - 361027*x^129 +
381462*x^128 - 189873*x^127 + 277571*x^126 + 272133*x^125 + 429218*x^124 +
254031*x^123 - 52469*x^122 + 86555*x^121 + 510471*x^120 - 408633*x^119 +
454981*x^118 - 73623*x^117 - 403171*x^116 - 510944*x^115 + 274152*x^114 +
446876*x^113 + 147157*x^112 + 335624*x^111 + 336964*x^110 - 184271*x^109 -
191293*x^108 + 520067*x^107 - 235397*x^106 + 86361*x^105 + 15785*x^104 +
381156*x^103 + 410169*x^102 + 500924*x^101 + 103790*x^100 - 192817*x^99 +
253109*x^98 - 276919*x^97 - 464206*x^96 - 128616*x^95 + 898*x^94 - 164346*x^93 +
358095*x^92 + 96307*x^91 - 171609*x^90 - 511753*x^89 - 152711*x^88 - 339475*x^87
- 423802*x^86 + 231732*x^85 + 148764*x^84 - 101023*x^83 + 519071*x^82 +
40485*x^81 - 353400*x^80 + 338082*x^79 + 462523*x^78 - 485285*x^77 + 221111*x^76
- 340306*x^75 + 354741*x^74 + 376886*x^73 - 413588*x^72 - 261448*x^71 -
495662*x^70 - 384828*x^69 + 458854*x^68 + 58500*x^67 + 423881*x^66 - 191745*x^65
+ 213195*x^64 - 456423*x^63 - 429628*x^62 + 83947*x^61 - 385407*x^60 -
252976*x^59 + 189208*x^58 - 322523*x^57 + 178897*x^56 - 345376*x^55 - 22763*x^54
- 26441*x^53 + 121576*x^52 - 395493*x^51 - 113161*x^50 + 436395*x^49 -
443993*x^48 + 154773*x^47 + 29805*x^46 + 431632*x^45 - 361015*x^44 + 381470*x^43
- 189878*x^42 + 277565*x^41 + 272141*x^40 + 429225*x^39 + 254037*x^38 -
52468*x^37 + 86562*x^36 + 510474*x^35 - 408628*x^34 + 454987*x^33 - 73627*x^32 -
403173*x^31 - 510944*x^30 + 274153*x^29 + 446867*x^28 + 147150*x^27 +
335625*x^26 + 336960*x^25 - 184270*x^24 - 191301*x^23 + 520070*x^22 -
235404*x^21 + 86354*x^20 + 15782*x^19 + 381158*x^18 + 410163*x^17 + 500922*x^16
+ 103784*x^15 - 192816*x^14 + 253109*x^13 - 276915*x^12 - 464211*x^11 -
128617*x^10 + 904*x^9 - 164352*x^8 + 358100*x^7 + 96307*x^6 - 171598*x^5 -
```

```
511750*x^4 - 152700*x^3 - 339471*x^2 - 423811*x + 231731, -92731*x^255 -
505171*x^254 - 214728*x^253 - 226524*x^252 - 390647*x^251 - 154832*x^250 +
403042*x^249 - 107145*x^248 - 116613*x^247 + 225210*x^246 - 337667*x^245 +
178510*x^244 + 138932*x^243 + 291040*x^242 - 403807*x^241 - 277641*x^240 -
138670*x^239 + 156638*x^238 - 4300*x^237 - 41669*x^236 + 423900*x^235 +
220225*x^234 + 187867*x^233 + 289916*x^232 - 150139*x^231 + 139923*x^230 +
331597*x^229 + 18748*x^228 + 122340*x^227 - 366666*x^226 - 354717*x^225 +
517978*x^224 + 35741*x^223 - 436080*x^222 - 476031*x^221 - 514804*x^220 +
24229*x^219 - 452739*x^218 + 89638*x^217 - 302405*x^216 - 254058*x^215 +
434385*x^214 + 388595*x^213 - 1617*x^212 + 29346*x^211 + 183224*x^210 +
500579*x^209 - 408255*x^208 - 2942*x^207 + 145008*x^206 + 1408*x^205 -
83846*x^204 + 427565*x^203 + 402883*x^202 + 498164*x^201 - 485372*x^200 +
455484*x^199 + 481631*x^198 + 273505*x^197 + 134188*x^196 - 435021*x^195 +
117103*x^194 + 230073*x^193 - 521234*x^192 - 452080*x^191 + 12374*x^190 +
183782*x^189 + 493147*x^188 + 387944*x^187 - 359378*x^186 + 73147*x^185 -
51317*x^184 + 522893*x^183 - 140932*x^182 + 309900*x^181 + 220028*x^180 -
144576*x^179 + 179236*x^178 + 327996*x^177 - 276857*x^176 - 481890*x^175 +
76122*x^174 + 397133*x^173 + 9198*x^172 + 269283*x^171 - 92733*x^170 -
505172*x^169 - 214733*x^168 - 226522*x^167 - 390642*x^166 - 154832*x^165 +
403051*x^164 - 107132*x^163 - 116599*x^162 + 225209*x^161 - 337656*x^160 +
178517*x^159 + 138942*x^158 + 291040*x^157 - 403799*x^156 - 277644*x^155 -
138673*x^154 + 156639*x^153 - 4291*x^152 - 41658*x^151 + 423900*x^150 +
220233*x^149 + 187868*x^148 + 289912*x^147 - 150134*x^146 + 139930*x^145 +
331599*x^144 + 18755*x^143 + 122342*x^142 - 366670*x^141 - 354722*x^140 +
517965*x^139 + 35744*x^138 - 436077*x^137 - 476027*x^136 - 514811*x^135 +
24229*x^134 - 452737*x^133 + 89638*x^132 - 302388*x^131 - 254065*x^130 +
434391*x^129 + 388581*x^128 - 1630*x^127 + 29331*x^126 + 183216*x^125 +
500594*x^124 - 408246*x^123 - 2932*x^122 + 145006*x^121 + 1412*x^120 -
83861*x^119 + 427561*x^118 + 402881*x^117 + 498163*x^116 - 485375*x^115 +
455488*x^114 + 481634*x^113 + 273505*x^112 + 134199*x^111 - 435017*x^110 +
117100*x^109 + 230074*x^108 - 521239*x^107 - 452089*x^106 + 12371*x^105 +
183777*x^104 + 493150*x^103 + 387946*x^102 - 359372*x^101 + 73150*x^100 -
51319*x^99 + 522890*x^98 - 140928*x^97 + 309902*x^96 + 220024*x^95 - 144581*x^94
+ 179242*x^93 + 327993*x^92 - 276853*x^91 - 481892*x^90 + 76119*x^89 +
397134*x^88 + 9191*x^87 + 269283*x^86 - 92725*x^85 - 505166*x^84 - 214735*x^83 -
226516*x^82 - 390650*x^81 - 154830*x^80 + 403046*x^79 - 107144*x^78 -
116607*x^77 + 225207*x^76 - 337656*x^75 + 178512*x^74 + 138938*x^73 +
291044*x^72 - 403788*x^71 - 277643*x^70 - 138670*x^69 + 156626*x^68 - 4295*x^67
- 41663*x^66 + 423902*x^65 + 220231*x^64 + 187865*x^63 + 289915*x^62 -
150133*x^61 + 139932*x^60 + 331595*x^59 + 18757*x^58 + 122348*x^57 - 366660*x^56
- 354717*x^55 + 517969*x^54 + 35742*x^53 - 436084*x^52 - 476028*x^51 -
514804*x^50 + 24234*x^49 - 452742*x^48 + 89638*x^47 - 302394*x^46 - 254066*x^45
+ 434388*x^44 + 388588*x^43 - 1620*x^42 + 29333*x^41 + 183216*x^40 + 500583*x^39
- 408256*x^38 - 2941*x^37 + 145002*x^36 + 1415*x^35 - 83846*x^34 + 427561*x^33 +
402881*x^32 + 498165*x^31 - 485374*x^30 + 455488*x^29 + 481629*x^28 +
273499*x^27 + 134193*x^26 - 435018*x^25 + 117108*x^24 + 230077*x^23 -
521238*x^22 - 452083*x^21 + 12377*x^20 + 183779*x^19 + 493153*x^18 + 387953*x^17
- 359380*x^16 + 73145*x^15 - 51318*x^14 + 522887*x^13 - 140933*x^12 +
```

309901*x^11 + 220034*x^10 - 144576*x^9 + 179239*x^8 + 327991*x^7 - 276855*x^6 -
481893*x^5 + 76117*x^4 + 397138*x^3 + 9187*x^2 + 269279*x - 92733), [False,
False, False, False, False, False, False, False, False, False, False, False,
False, False, False, False, False, False, False, False, False, False, False,
False, False, False, False, False, False, False, False, False, False, False,
False, False, False, False, False, False, False, False, False, False, False,
False, False, False, False, False, True, False, False, False, False, False,
False, False, False, False, False, False, False, False, False, False, False,
False, False, False, False, False, False, False, False, False, False, False,
False, False, False, False, False, False, False, False, False, False, False,
False, False, False, True, False, False, False, False, False, False, False,
False, False, False, False, False, False, False, True, False, False, False,
False, False, False, False, False, False, False, False, False, False, False,
False, True, False, False, False, False, False, False, False, False, False,
False, False, False, False, False, False, False, False, False, False, False,
False, False, False, False, False, False, False, False, False, False, False,
False, False, False, False, True, False, False, True, False, False, False,
False, False, False, False, False, False, False, False, False, False, False,
False, False, False, False, False, False, False, False, False, False, False,
False, True, False, False, False, False, False, False, False, False, False,
False, False, False, False, False, False, False, False, False, False, False,
False, False, False, False, False, False, False, False, False, False, False,
False, False, False, False, False, True, False, False, False, False, False,
True, False, False, True, False, False, True, False, True, False, False, False,
False, False, False, False, False, False, False, False, False, True, False,
True, False, False, False, True, False, False, False, False, False, False,
False, False, False, False, False, False, False, False, False, False, False,
False, False, False, False, False, False, False, False, False, False, False,
False, False, True, False, False, False, False, False, False, False, False,
False, False, False, True, False, False, False, False, False, False, False,
False, False, False, False, False, False, False, False, False, False, False,
False, False, False, False, False, False, False, False, False, False, False,
False, False, False, False, False, False, False, False, False, False, False,
False, False, False, False, False, False, False, False, False, False, False,
False, False, False, False, False, False, False, False, False, False, False,
False, False, False, False, False, False, False, False, False, False, False,
False, False, False, False, False, False, False, False, False, False, False,
True, False, False, False, False, False, False, False, False, False, False,
False, False, False, False, False, False, False, False, False, False, False,
False, False, False, False, False, False, False, False, False, False, False,
False, False, False, False, False, False, False, False, False, False, False,
False, False, False, False, False, True, False, False, False, False, False,
False, False, False, False, False, True, False, False, False, False, False,
False, False, False, False, False, False, True, False, False, False, False,
False, False, False, False, False, False, False, False, False, False, False,

False, False, False, False, False, False, False, False, False, False, True,
False, False, False, False, False, False, False, False, False, False, False,
False, False, False, False, False, False, False, False, False, False, False,
False, False, False, False, False, False, False, False, False, False, False,
False, False, False, False, False, False, False, False, False, False, False,
False, False, False, False, False, False, False, False, False, False, False,
False, False, False, False, False, False, False, False, False, True, False,
False, False, False, False, False, False, False, False, False, False, False,
False, False, False, False, False, False, False, False, False, False, False,
False, False, False, False, False, False, False, False, False, False, False,
False, False, False, False, False, False, False, False, False, False, False,
False, False, False, False, False, False, False, False, False, False, False,
False, False, False, False, False, False, False, False, False, True, False,
False, False, False, False, False, False, False, False, False, False, False,
False, False, False, False, False, False, False, False, False, False, False,
False, False, True, False, False, False, False, False, False, False, False,
False, False, False, False, False, False, False, False, False, False, False,
False, False, False, False, False, True, False, False, False, False, False,
False, False, False, False, False, False, False, False, False, False, False,
False, False, False, False, False, False, False, False, False, False, False,
False, False, False, False, False, False, False, True, False, False, False,
False, False, False, False, False, False, False, False, False, False, False,
False, False, False, False, False, False, False, False, False, False, False,
False, False, False, False, False, False, False, False, False, False, False,
False, False, False, False, False, False, False, False, False, False, False,
True, False, False, False, False, False, False, False, False, False, False,
False, False, False, False, False, False, False, False, False, False, False,
False, False, False, False, False, False, False, False, False, False, False,
False, False, False, False, False, False, False, False, False, False, False,
False, False, False, True, True, False, False, False, False, False, False,
False, False, False, True, False, False, False, False, True, False, False,
False, False, False, False, False, False, False, False, False, False, False,
False, False, False, False, False, False, False, False, True, False, True,
False, False, False, False, False, False, False, False, False, False, False,
False, False, False, False, False, False, False, False, False, False, False,
False, False, False, False, False, False, False, False, False, False, False,
False, False, False, False, False, False, False, False, False, False, False,
False, False, False, False, False, False, False, False, False, False, False,
False, True, False, False, False, False, False, False, False, False, False,
False, False, False, False, False, False, False, False, False, False, False,
False, False, False, False, False, False, False, False, False, False, False,
False, False, False, True, False, False, False, False, False, False, False,
True, False, False, False, False, False, False, False, False, False, False,

False, False, False, False, False, False, False, False, False, False, False,
False, False, False, False, False, False, False, False, False, False, False,
False, False, False, False, False, False, False, False, True, False, False,
False, False, False, False, False, False, False, False, False, False, False,
False, False, False, False, False, False, False, False, False, False, False,
False, False, False, False, False, False, False, False, False, False, False,
False, False, False, True, False, False, False, False, False, False, False,
False, False, False, False, False, False, False, False, False, False, False,
False, False, False, True, False, False, False, False, False, False, False,
False, False, False, False, False, False, False, False, False, False, False,
False, False, False, False, False, False, False, False, False, False, False,
False, False, False, False, False, False, False, False, False, False, False,
False, False, False, False, False, False, False, False, False, False, False,
False, False, False, False, False, False, False, False, False, False, False,
False, False, False, False, False, False, False, False, False, False, False,
False, False, False, False, False, False, False, False, False, False, False,
False, False, False, False, False, False, False, True, False, False, False,
False, False, False, False, False, False, False, False, False, False, False,
False, False, False, False, False, False, False, False, False, False, False,
False, False, False, False, False, False, False, False, False, False, False,
False, False, False, False, False, False, False, False, False, True, False,
False, False, False, False, True, False, False, False, False, False, False,
False, False, False, False, False, False, False, False, False, False, False,
False, False, False, False, False, False, False, False, False, False, False,
False, False, False, False, False, False, False, False, False, False, False,
False, True, False, False, True, False, False, False, False, False, False,
False, False, False, False, False, True, False, False, False, False, False,
False, False, False, False, False, False, False, False, True, False, False,
False, False, False, False, False, False, False, False, False, False, False,
False, False, False, False, False, False, False, False, False, False, False,
False, False, False, False, False, False, True, False, False, False, False,
False, False, False, False, False, False, False, False, False, False, False,
False, False, False, False, False, False, False, False, False, False, False,
False, False, False, False, False, False, False, False, False, False, False,
False, False, False, False, False, False, False, False, False, False, False,
False, False, False, False, False, False, False, False, False, False, False,
True, False, False, False, False, False, False, False, False, False, True,
False, False, False, False, False, False, False, False, False, False, False,
False, False, False, False, False, False, False, False, False, False, False,

False, False, False, False, False, False, False, False, False, False, False,
False, False, False, False, False, False, False, False, False, False, False,
False, False, False, False, False, False, False, False, False, False, False,
False, False, False, False, False, False, False, False, False, False, False,
False, False, False, False, False, False, False, False, False, False, False,
False, False, False, False, False, False, False, False, False, False, False,
False, False, False, False, False, False, False, False, False, False, False,
True, False, False, False, False, False, False, False, False, False, False,
False, False, False, False, False, False, False, False, False, False, False,
False, False, False, False, False, False, False, False, False, False, False,
False, False, False, False, False, False, False, False, False, False, False,
False, False, False, False, False, False, False, False, False, False, False,
False, False, False, False, False, False, False, False, True, False, False,
False, False, False, False, False, False, False, False, False, False, False,
False, False, False, False, False, False, False, False, False, False, False,
False, False, False, False, False, False, False, False, False, False, False,
False, False, False, False, False, False, False, False, False, False, False,
False, False, False, True, False, False, False, False, False, False, False,
False, False, False, False, False, False, False, False, False, False, False,
False, False, False, False, False, False, False, False, False, False, False,
False, False, False, False, False, False, False, False, False, False, False,
False, False, False, False, False, False, False, False, False, False, False,
False, False, False, False, False, False, False, False, False, False, False,
True, False, False, False, False, False, False, False, False, False, False,
False, False, False, False, False, False, False, False, False, False, False,
False, False, False, False, False, False, False, False, False, False, False,
False, False, False, False, False, False, False, False, False, False, False,
False, False, False, False, False, False, False, False, False, False, False,
False, False, False, False, False, False, False, False, False, False, False,
False, False, False, False, False, False, False, True, False, False, False,
False, False, False, False, False, False, True, False, False, False, False,
False, False, False, False, False, False, False, False, False, False, False,
False, False, False, False, False, False, False, False, False, False, False,
False, False, False, False, False, False, False, False, False, False, False,
False, False, False, False, False, False, False, False, False, False, False,
False, False, False, False, False, False, False, False, False, False, False,
False, False, True, False, True, False, False, False, False, False, False,
False, False, False, False, False, False, False, False, False, False, False,
False, False, False, False, False, True, False, False, False, False, False,
False, False, False, False, False, False, False, False, False, False, True,
False, False, False, False, False, False, False, False, False, False, False,

```
False, False, False, False, False, False, False, False, False, False, False,
False, True, False, False, False, False, False, False, False, False, False,
False, False, False, False, False, False, False, False, False, False, False,
False, False, False, False, False, False, False, False, False, False, False,
False, False, False, False, False, False, False, False, False, True, False,
False, False, False, False, False, False, True, False, False, False, False,
False, False, False, False, False, False, False, False, False, False, False,
False, False, False, False, False, False, False, False, False, False, False,
False, False, False, False, False, False, False, False, False, False, False,
False, False, False, False, False, False, False, False, False, False, False,
False, False, False, False, False, False, False, False, False, False, False,
True, False, False, False, False, False, False, False, False, False, False,
False, False, False, False, False, False, False, False, False, False, False,
False, True, False, False, False, False, False, False, False, False, False,
False, False, False, False, False, False, False, False, False, False, False,
False, False, False, False, False, False, False, False, False, False, True,
False, False, False, False, False, False, False, False, False, True, False,
False, False, False, False, False, False, False, False, False, False, False,
False, False, False],
b'\xfc\xa1`i;f`\x07\xd8&vur\xb9`\xf3\xa3,\xc0\xf2R(q\x17~\x9a\xcf;\x834\xf9|')
Assinatura validada!
A testar com mensagem inválida...  Outra mensagem
Ocorreu um erro!
```