

Álgebra A

Trabalho Prático 1: Aritmética

Questão 1. Implemente o algoritmo de Euclides estendido: Dados números a e b , sua função deve computar números x , y e g tais que $\text{mdc}(a, b) = g$ e $ax + by = g$.

Valor de retorno Não há.

Assinatura `void mdc_estendido(mpz_t g, mpz_t x, mpz_t y,
const mpz_t a, const mpz_t b)`

	Nome	Tipo	Descrição
Entrada	a	mpz_t	
	b	mpz_t	
Saída	g	mpz_t	O maior divisor comum de a e b.
	x	mpz_t	Inteiro tal que $ax + by = g$.
	y	mpz_t	Inteiro tal que $ax + by = g$.

Como testar: Só existe um valor de g correto, que você pode conferir com a função `mpz_gcd(mpz_t g, mpz_t a, mpz_t b)`. Quanto a (x, y) , existe a função `mpz_gcdext`, com mesma assinatura da função `mdc_estendido`, mas é fato que existem vários pares (x, y) válidos. Você pode facilmente verificar se a equação $ax + by = g$ é satisfeita pelos seus números.

Questão 2. Use a questão anterior para implementar uma função que calcula o inverso modular de um número. Sua função deve retornar um inteiro que indica se o número tem inverso modular; caso o número tenha inverso modular, sua função deve preencher o argumento r com o valor.

Valor de retorno 1 se o inverso modular existe e foi calculado,
0 se o inverso modular não existe.

Assinatura `int inverso_modular(mpz_t r,
const mpz_t a,
const mpz_t n)`

	Nome	Tipo	Descrição
Entrada	a	mpz_t	
	n	mpz_t	
Saída	r	mpz_t	Um número tal que $ar \equiv 1 \pmod{n}$, caso exista.

Como testar: Você pode comparar sua resposta com a função `mpz_invert`, que tem mesma assinatura.

Questão 3. Implemente o algoritmo de exponenciação rápida visto em sala, usando que

$$b^e = \begin{cases} (b^{e/2})^2 & \text{se } e \text{ for par} \\ b \cdot (b^{\lfloor e/2 \rfloor})^2 & \text{se } e \text{ for ímpar.} \end{cases}$$

O número de chamadas à função `mpz_mul` deverá ser proporcional ao número de bits do argumento `e`.

Como vimos em sala, tanto a versão recursiva quanto a versão iterativa dessa função são razoavelmente simples. Se você preferir, pode implementar essa função iterativamente; isso irá acelerar sua função descriptografa do TP 3.

Valor de retorno Não há.

Assinatura `void exp_binaria(mpz_t r,`
`const mpz_t b,`
`const mpz_t e,`
`const mpz_t n)`

	Nome	Tipo	Descrição
Entrada	<code>b</code>	<code>mpz_t</code>	
	<code>e</code>	<code>mpz_t</code>	
	<code>n</code>	<code>mpz_t</code>	
Saída	<code>r</code>	<code>mpz_t</code>	Um número tal que $b^e \equiv r \pmod{n}$ e $0 \leq r < n$.

Como testar: Você pode comparar sua resposta com a função `mpz_powm`, que tem mesma assinatura.