Crittografia RSA (`rsa`)

Per rendere ancora più sicure le comunicazioni segrete tra gli smartphone durante l'esame di maturità, gli studenti stanno ponderando l'eventualità di *cifrare* tutti i messaggi che si scambieranno. Nel corso dell'ultimo anno scolastico hanno infatti studiato l'algoritmo di cifratura asimmetrica RSA. L'idea è semplice: realizzare un'app che cifri i messaggi e un'altra che li decifri; in questo modo anche qualora i commissari dovessero intercettarli non riuscirebbero a leggerne il contenuto!

La maturità si avvicina e i lavori per l'app di decifratura sono ancora in alto mare. In estrema sintesi[1], ti viene fornita una coppia di interi N e d che costituiscono la "chiave privata". Ogni intero c, che rappresenta un carattere cifrato, si decifra calcolando

$$c^d \bmod N$$

> ☞ L'operazione di modulo restituisce il resto della divisione intera tra due numeri. In C/C++, si calcola con l'operatore `%`. Questa operazione gode, inoltre, delle seguenti proprietà (molto utili per evitare *integer overflow* quando si vogliono calcolare numeri molto grandi):
>
> - $(A + B) \bmod M = (A \bmod M + B \bmod M) \bmod M$
> - $(A \cdot B) \bmod M = (A \bmod M \cdot B \bmod M) \bmod M$

Aiuta gli studenti a scrivere l'app per la decifratura di un intero messaggio lungo L caratteri, ciascuno da decifrare singolarmente per produrre il testo in chiaro e mettere al sicuro il buon esito dell'esame!

Implementazione

Dovrai sottoporre un unico file con estensione `.cpp` o `.c`.

> ☞ Tra gli allegati a questo task troverai un template (`rsa.cpp` e `rsa.c`) con un esempio di implementazione.

Dovrai implementare la seguente funzione:

■ Funzione `decifra`

C/C++	`void decifra(int N, int d, int L, int messaggio[], char plaintext[]);`

- Gli interi N e d costituiscono la chiave privata necessaria per decifrare il messaggio.
- L'intero L rappresenta la lunghezza del messaggio che è stato cifrato e trasmesso.
- L'array `messaggio`, indicizzato da 0 a $L-1$, contiene alla posizione i l'intero c che rappresenta l'i-esima lettera cifrata.
- La funzione dovrà riempire l'array `plaintext` in modo che le posizioni da 0 a $L-1$ contengano ciascuna il risultato della decifrazione del messaggio contenuto a quella rispettiva posizione nell'array `messaggio`.
  La posizione L dell'array `plaintext` dovrà contenere il carattere di fine stringa `'\0'`.

Il grader chiamerà la funzione `decifra` e stamperà l'array `plaintext` sul file di output.

[1] Un'introduzione all'algoritmo completo, *non necessaria per risolvere questo problema*, si trova per chi volesse saperne di più presso `https://it.wikipedia.org/wiki/RSA`.

Grader di prova

Allegata a questo problema è presente una versione semplificata del grader usato durante la correzione, che potete usare per testare le vostre soluzioni in locale. Il grader di esempio legge i dati da `stdin`, chiama la funzione che dovete implementare e scrive su `stdout`, secondo il seguente formato.

Il file di input è composto da 2 righe, contenenti:

- Riga 1: gli interi N, d e L, separati da uno spazio.
- Riga 2: i numeri interi `messaggio[`i`]`, per $i = 0, \ldots, L-1$.

Il file di output è composto da un'unica riga, contenente:

- Riga 1: il contenuto dell'array `plaintext`, così come modificato dalla funzione `decifra`.

Assunzioni

- $128 \leq N \leq 2^{31} - 1$ (ovvero N è rappresentabile con un intero, anche con segno, a 32 bit).
- $1 \leq d < N$.
- $1 \leq L \leq 100$.
- $0 \leq$ `messaggio[`i`]` $< N$ per ogni $i = 0 \ldots L-1$.
- È garantito che una corretta decifratura del messaggio porta ad avere un plaintext costitutito da numeri interi, in base 10, rappresentanti caratteri ASCII validi (in particolare, lettere minuscole).

Assegnazione del punteggio

Il tuo programma verrà testato su diversi test case raggruppati in subtask. Per ottenere il punteggio relativo ad un subtask, è necessario risolvere correttamente tutti i test relativi ad esso.

- **Subtask 1 [0 punti]**: Casi d'esempio.
- **Subtask 2 [25 punti]**: $L = 1$, `messaggio[0]` $\leq 1\,000$, $d = 3$.
- **Subtask 3 [25 punti]**: $N \leq 1\,000\,000$, $L = 1$.
- **Subtask 4 [10 punti]**: $N \leq 1\,000\,000$.
- **Subtask 5 [40 punti]**: Nessuna limitazione specifica.

Esempi di input/output

stdin	stdout
145 3 1 119	r
391 3 3 295 123 129	abc

Spiegazione

Nel **primo caso di esempio** dobbiamo decifrare un messaggio composto da un unico carattere cifrato: l'intero 119. Calcoliamo quindi $119^3 = 1\,685\,159$ e prendiamo il resto della divsione per $N = 145$. Il risultato è l'intero 114, che nella codifica ASCII rappresenta la lettera `'r'`.

Nel **secondo caso di esempio** procediamo in modo analogo, decifrando singolarmente ciascun intero.