XOR cipher

I membri di un'organizzazione segreta hanno inventato un nuovo algoritmo crittografico basato sull'operatore bitwise XOR per scambiarsi messaggi cifrati. Un detective ha intercettato diversi messaggi ed è riuscito ad ottenere lo pseudocodice dell'algoritmo.

I messaggi sono codificati in liste concatenate di caratteri, ad ognuno dei quali è stato applicato l'algoritmo di codifica con una chiave segreta. La chiave è una stringa alfanumerica di lunghezza variabile.

Lo pseudocodice dell'algoritmo è il seguente:

Algorithm 1 listEncodec($list, key, seed$)

Input: una lista di caratteri *list*, la stringa *key* usata per decifrare il messaggio, un intero *seed*: $seed \in \mathbb{N}$.
Output: Nessuno. **Side effect**: *list* contiene i caratteri decifrati.
if $list = \text{nil}$ **then**
    return
end if
$list.value \leftarrow$ charEncodec($list.value$, $key[seed \mod \text{len}(key)]$)
listEncodec($list.next$, key, seed + 1)

Algorithm 2 charEncodec(c, key)

Input: c carattere contenente l'informazione da decifrare, il carattere *key* usato per decifrare l'informazione.
Output: un carattere decifrato.
$result \leftarrow c \oplus key$
if not isPrintable($result$) **then**
    $result \leftarrow c$
end if
return *result*

Il simbolo $\oplus$ è l'operatore bitwise xor (`^`), mentre `mod` corrisponde all'operatore modulo (`%`).

Aiuta il detective implementando lo pseudocodice nelle seguenti funzioni C:

1. `char charEncodec(const char c, const char key)`
2. `void listEncodec(Node *list, const char *key, const int seed)`

Se necessario, si possono aggiungere funzioni di supporto, ma è necessario che le funzioni richieste siano implementate esattamente con i prototipi forniti.

> Lo pseudocodice non è codice completo in C: non considera i tipi dei vari oggetti, l'uso di puntatori/riferimenti, e dunque dell'operatore "->" anziché l'operatore ".": 'e lasciato allo studente il compito di determinare questi dettagli come adeguato.

Note

L'implementazione della lista in C è quella utilizzata durante le lezioni del corso e viene fornita già implementata nei file dell'esercizio come segue:

```
typedef struct node {
    char value;
    struct node *next;
} Node;
```

La funzione `isPrintable` viene fornita con il testo dell'esercizio, accetta in input un singolo carattere e restituisce un intero:

- 0 se non può essere stampato a terminale (ad esempio se si tratta dei caratteri `\0`, `\n`, `\t`, ...)
- 1 altrimenti

La funzione `len` prende in input un stringa e restituisce la lunghezza della stringa. Viene fornita già implementata nel testo dell'esercizio.

Soluzione (fare click per visualizzare)

La soluzione proposta è la seguente:

```
char charEncodec(const char c, const char key)
{
    char result = c ^ key;
    if (!isPrintable(result))
        result = c;
    return result;
}

void listEncodec(Node *list, const char *key, const int seed)
{
    if (list == NULL)
        return;
    // È necessario l'operatore -> poiché list è un puntatore
    // in tal modo si accede al campo value della struttura
    // puntata da list
    list->value = charEncodec(list->value, key[seed % len(key)]);
    // Chiamata ricorsiva
    listEncodec(list->next, key, seed + 1);
}
```