Esercizio

Scrivere il programma a linea di comando `lz78encode` in grado di comprimere file binari qualsiasi utilizzando l'algoritmo LZ78. Le posizioni del dizionario vengono codificate con un numero binario a tanti bit quanti quelli necessari a codificare il più grande valore possibile nel dizionario in un dato momento (ricordate che 0 significa stringa vuota e che le stringhe aggiunte partono da 1), mentre i byte aggiuntivi sono codificati con il loro valore a 8 bit.

Il software deve supportare la seguente sintassi:

```
lz78encode <max bits> <input filename> <output filename>
```

Il primo argomento indica il numero massimo di bit da utilizzare per salvare le posizioni e quindi anche la dimensione massima del dizionario (da 1 a 31). Se il dizionario raggiunge quella dimensione deve essere svuotato e si ricomincia a comprimere da dove si è arrivati, con un dizionario vuoto.
Gli altri due argomenti sono rispettivamente il nome del file da comprimere e quello da produrre in output. Tutti gli argomenti sono obbligatori.

Il file di output è codificato con un magic number "LZ78" (4 byte), poi un valore intero a 5 bit senza segno che indica il massimo numero di bit utilizzati per il dizionario durante la codifica e poi il file codificato. Ad esempio sia dato il seguente file:

```
aabaacabcabcbaa
```

Se lo codificassimo con `<max bits>=4` otterremmo:

```
4C 5A 37 38   "LZ78"
00100         4 (max bits)
              0 (codificato con 0 bit) ← Notate che il primo zero non viene neppure trasmesso
01100001      'a'
1             1 (codificato con 1 bit)
01100010      'b'
01            1 (codificato con 2 bit)
01100001      'a'
00            0 (codificato con 2 bit)
01100011      'c'
010           2 (codificato con 3 bit)
01100011      'c'
101           5 (codificato con 3 bit)
01100010      'b'
001           1 (codificato con 3 bit)
01100001      'a'
00000         padding per completare l'ultimo byte

Scrivo i valori di seguito:
4C 5A 37 38 00100 01100001 1 01100010 01 01100001 00 01100011 010 01100011 101 01100010 001
01100001 00000

Li raggruppo in ottetti:
4C 5A 37 38 00100011 00001101 10001001 01100001 00011000 11010011 00011101 01100010 00101100
00100000

Tutto in esadecimale:
4C 5A 37 38 23 0D 89 61 18 D3 1D 62 2C 20
```

Se invece lo codificassimo con <max bits>=2 otterremmo:

```
4C 5A 37 38   “LZ78”
00010         2 (max bits)
              0 (codificato con 0 bit)
01100001      ‘a’
1             1 (codificato con 1 bit)
01100010      ‘b’
01            1 (codificato con 2 bit)
01100001      ‘a’
00            0 (codificato con 2 bit)
01100011      ‘c’ ← a questo punto il dizionario viene svuotato
              0 (codificato con 0 bit)
01100001      ‘a’
0             0 (codificato con 1 bit)
01100010      ‘b’
00            0 (codificato con 2 bit)
01100011      ‘c’
01            1 (codificato con 2 bit)
01100010      ‘b’ ← a questo punto il dizionario viene svuotato
              0 (codificato con 0 bit)
01100011      ‘c’
0             0 (codificato con 1 bit)
01100010      ‘b’
00            0 (codificato con 2 bit)
01100001      ‘a’
00            0 (codificato con 2 bit)
01100001      ‘a’
0000          padding per completare l’ultimo byte

Scrivo i valori di seguito:
4C 5A 37 38 00010 01100001 1 01100010 01 01100001 00 01100011 01100001 0 01100010 00 01100011 01
01100010 01100011 0 01100010 00 01100001 00 01100001

Li raggruppo in ottetti:
4C 5A 37 38 00010011 00001101 10001001 01100001 00011000 11011000 01001100 01000011 00011010
11000100 11000110 01100010 00011000 01000110 00010000

Tutto in esadecimale:
4C 5A 37 38 13 0D 89 61 18 D8 4C 43 1A C4 C6 62 18 46 10
```