Esercizio 1

Sia dato il seguente formato per la codifica di immagini mediche:

Campo	**Dimensione**	**Descrizione**
Magic Number	6 byte	"OCTIMG"
Width	intero senza segno a 32 bit big endian	Larghezza in pixel
Height	intero senza segno a 32 bit big endian	Altezza in pixel
NumImages	intero senza segno a 10 bit	Numero di immagini nel file
NumElement	intero senza segno a 16 bit big endian	Numero di elementi nella tabella di Huffman seguente
Huffman Table	NumElement terne di - intero senza segno a 8 bit - intero senza segno a 8 bit - intero senza segno a 5 bit	Tabella delle terne (R, G/B, lunghezza dei codici di Huffman): per ogni possibile simbolo (R,G/B) viene specificata la lunghezza del codice di Huffman.
Dati	Variabile	Valori delle immagini, per righe, codificati con i codici canonici di Huffman (a partire da 0 per il più probabile), secondo la tabella precedente.

Le immagini sono a colori, ma i canali G e B sono sempre uguali.

Si scriva un programma, **a linea di comando**, che accetti la sintassi seguente:

```
octimgdec <file di input> <prefisso di output>
```

Il programma deve aprire il file di input e produrre in output *NumImages* immagini i cui nomi saranno dati dal prefisso a cui aggiungere 001, 002, …, 009, 010, 011, 012, …, 099, 100, 101, 102, … fino a *NumImages*. Per fare questo deve

1) leggere la tabella di Huffman
2) creare i codici canonici a partire da 0
3) leggere i codici di Huffman dal file e per ogni codice riempire il pixel corrente con il valore R e i valori G/B uguali.
4) salvare le immagini come *prefisso*001.ppm, *prefisso*002.ppm, ecc…

Le specifiche del formato PPM sono fornite nel file *PPM Format Specification.html*. Utilizzare la versione P6.

Consideriamo l'immagine *esempio.octimg* vista in un editor esadecimale:

```
Offset(h) 00 01 02 03 04 05 06 07 08 09 0A 0B 0C 0D 0E 0F
00000000  4F 43 54 49 4D 47 00 00 00 02 00 00 00 02 00 80  OCTIMG.........€
00000010  00 FF E6 43 FE 3C 2F FF F1 62 80                 .ÿæCþ</ÿñb€
```

Questo file è del formato indicato, contiene immagini 2×2 e i byte successivi sono 00 80 00 FF E6 43 FE 3C 2F FF F1 62, ovvero

0000.0000 1000.0000 0000.0000 1111.1111 1110.0110 0100.0011 1111.1110 0011.1100 0010.1111 1111.1111 1111.0001 0110.0010 1000.0000

In giallo vediamo NumImages, ovvero 2.
In verde vediamo NumElement, ovvero 3. Poi seguono tre terne (R, G/B, len):
0xFF, 0x99, 1
0xFF, 0x1E, 2
0xFF, 0xFF, 2

I codici canonici di Huffman saranno:
0xFF, 0x99, 0
0xFF, 0x1E, 10
0xFF, 0xFF, 11

Quindi decodificando i bit restanti troviamo: 110.0010 1000.0000
11 0
0 0

10 10
0 0

più un padding di 4 bit a 0 (non è un problema, dato che sappiamo che le immagini sono 2×2).

Le immagini risultanti saranno allora:
FF,FF,FF FF,99,99
FF,99,99 FF,99,99

FF,1E,1E FF,1E,1E
FF,99,99 FF,99,99

ovvero (con tanto zoom):