Lab Exercises 12/04/2021: Huffman

Esercizio 1

Write a command line program in C++ with this syntax:

```
huffman1 [c|d] <input file> <output file>
```

When the "c" option is specified, the program opens the specified file (the file must be treated as a binary file, that is, it can contain any value from 0 to 255 in each byte), calculates the frequencies and generates the corresponding Huffman codes. Then it produces a file with the following format:

Field	**Size**	**Description**
MagicNumber	8 byte	“HUFFMAN1”
TableEntries	8-bit unsigned integer	Number of items in the following Huffman table **(0 means 256 symbols)**.
HuffmanTable	TableEntries **triplets** (sym = 8 bit, len = 5 bit, code = len bit)	Triplet table with *symbol, length and Huffman code*: the length and the Huffman code is specified for each symbol. The code is written with as many bits as indicated in the triplet len field.
NumSymbols	32 bit unsigned integer stored in **big endian**	Number of symbols encoded in the file.
Data	NumSymbols Huffman codes	Values encoded with Huffman codes, according to the previous table.

When the "d" option is specified, the program decompresses the contents of the input file (check that it’s stored in the previous format) and saves it in the output file.

Esercizio 2

Variation on the previous one:

```
huffman2 [c|d] <input file> <output file>
```

with this format:

Field	Size	Description
MagicNumber	8 byte	“HUFFMAN2”
TableEntries	8-bit unsigned integer	Number of items in the following Huffman table **(0 means 256 symbols)**.
HuffmanTable	TableEntries **couples** (sym = 8 bit, len = 5 bit)	Table of *symbol, length* couples. This table **must** be sorted on len and then on sym.
NumSymbols	32 bit unsigned integer stored in **big endian**	Number of symbols encoded in the file.
Data	NumSymbols Huffman codes	Values encoded with **canonical** Huffman codes, according to the previous table.

In this case the Huffman codes are the canonical codes with the most likely one starting from 0.