Exercise 1

To Show: Elias-Gamma and Elias-Delta are prefix free.

Proof:
Elias-Gamma Encoding
By definition, Elias-Gamma is constructed through placing $\lfloor \log_2 n \rfloor$ zeroes infront of the binary represantion of the number n. Therefore we have a code whereas the first **x** bits are zeros and the next x+1 bits are the binary represantion of the number.

Obvious, this is prefix free because we can construct a decoder which reads and counts the first bits as long as they are zeros (= x). When he reaches the first '1' he knows, that the following x+1 bits are the binary code for that particular number.
We can construct a deterministic pushdown automaton who accepts our code, and therefor we have a context-free language (which is prefix free).

Elias-Delta Encoding
We can do the same with the Elias-Delta code, but the decoder has to read the first n zero bits and the following n+1 bits to get the number (here: the prefix) which is decoded by Elias-Gamma. With those n and n+1 bits he computes the number j and knows, that the following $\lfloor \log_2 j \rfloor + 1$ bits is the binary represantion of the decoded number.

Also, with Krafts inequality, we can see (ex.2) that $\sum_{i=1}^{\infty} 2^{-(2\cdot\lfloor log_2 n \rfloor+1)} \leq 1$, and therefor our encoding is prefix free.

Exercise 2

As the Elias-Gamma code is constructed through the binary representation of a number n (which takes $\lfloor log_2 n \rfloor + 1$ bits) and the prepanded zeros (which take $\lfloor log_2 n \rfloor$ bits), we get:

$$l_i = 2 \cdot \lfloor log_2 n \rfloor + 1$$

$$\sum_{i=1}^{\infty} 2^{-(2\cdot\lfloor log_2 n\rfloor+1)} = \sum_{i=1}^{\infty} \frac{1}{2^{2\cdot\lfloor log_2 n\rfloor+1}} \tag{1}$$

$$= \frac{1}{2}\cdot\sum_{i=1}^{\infty} \frac{1}{2^{2\cdot\lfloor log_2 n\rfloor}} \tag{2}$$

$$= \frac{1}{2}\cdot\left(\frac{1}{2^{2\cdot 0}} + \frac{1}{2^{2\cdot 1}} + \frac{1}{2^{2\cdot 1}} + \frac{1}{2^{2\cdot 2}} + \frac{1}{2^{2\cdot 2}} + \frac{1}{2^{2\cdot 2}} + \frac{1}{2^{2\cdot 2}} + \frac{1}{2^{2\cdot 3}} + \cdots\right) \tag{3}$$

$$= \frac{1}{2}\cdot\left(1 + 2\cdot\frac{1}{2^{2\cdot 1}} + 4\cdot\frac{1}{2^{2\cdot 2}} + 8\cdot\frac{1}{2^{2\cdot 3}} + \cdots\right) \tag{4}$$

$$= \frac{1}{2}\cdot\left(1 + \frac{2^1}{2^{2\cdot 1}} + \frac{2^2}{2^{2\cdot 2}} + \frac{2^3}{2^{2\cdot 3}} + \cdots\right) \tag{5}$$

$$\tag{6}$$

$$= \frac{1}{2} \cdot (1 + \frac{1}{2} + \frac{1}{4} + \frac{1}{8} + \frac{1}{16} + \cdots) \tag{7}$$
$$= \frac{1}{2} \cdot (1 + 1) \tag{8}$$
$$= 1 \tag{9}$$

Exercise 3

To Show: $\nexists$ universal encoding scheme with $l_1 = 1$ to the number 1 and $l_i \leq 2 \cdot \log_2 i$ $\forall$ integers $i \geq 2$

If we try to build a code with $l_i \leq 2 \cdot \log_2 i$ and $l_i = 1$ we have to take '1' for the number 1 (wlog). Therefor all numbers ≥ 2 will have a series of '0' as a prefix.
If we want to get a code below $2 \cdot \log_2 i$, we have to build a tree like in the lecture on slide 16. With that method we have to code our numbers with a scheme like to stay prefix free:

$$n \cdot \text{"0"} + ((n-1) \cdot \text{"0"} + 1)$$

With this method, we'll be $\geq 2 \cdot \log_2 i$, and we can't do better because i.e. to achive $l_5 \leq 4.64 \cdots$, we have to map 5 = "0000äs all other prefixes are taken. This would break the possible code for $l_6 \leq 5.17 \cdots$ as all prefixes are taken.
Krafts Inequality $\sum_{i=1}^{\infty} 2^{-(2 \cdot \log_2 n)} \approx 1.644 \geq 1$ shows us that this can't be prefix free.
$\Longrightarrow$ $\nexists$ universal encoding scheme $\cdots$

It doesn't contradict to the result of ex.2 because the sum of $2 \cdot \log_2 i$ is overall smaller than $2 \cdot \lfloor \log_2 i \rfloor + 1$.