Problem 1

Let X denote the random variable that describes the current state, and Y denote the random variable that describes the next state (i.e. $p(Y = w|X = w) = 0.9$, $p(Y = w|X = b) = 0.2$, etc.).

Then the conditional entropy $H(Y|X)$ is

$$H(Y|X) = p(X = w)H(Y|X = w) + p(X = b)H(Y|X = b).$$

We also have

$$\begin{aligned} H(Y|X = w) &= -p(Y = w|X = w)\log(p(Y = w|X = w)) - p(Y = b|X = w)\log(p(Y = b|X = w)) \\ &= 0.46 \end{aligned}$$

and similarly, $H(Y|X = b) = 0.72$.

The question does not specify the unconditional probabilities for b and w. However, if we assume that these probabilities are independent from the absolute position in the stream of b's and w's (i.e. we look for the stationary probability distribution of the Markov process), we can deduce that

$$p(X = w) = 2/3, \quad p(X = b) = 1/3$$

by using the equations

$$P(Y = w) = p(Y = w|X = b)p(X = b) + p(Y = w|X = w)p(X = w),$$

and

$$P(X = w) = P(Y = w), \qquad P(X = w) = 1 - P(X = b).$$

It follows that $H(Y|X) = 0.55$, while the unconditional entropy is

$$H(X) = H(Y) = \frac{2}{3}\log\frac{3}{2} + \frac{1}{3}\log 3 = 0.91.$$

Therefore, knowing the previous character reduces entropy by a factor of $H(X)/H(Y|X) = 1.66$.

Problem 2

The unary code ($C(i) = 10^i$) is uniquely decipherable but not prefix-free. In particular, for any $i < j$, $C(i)$ is a prefix of $C(j)$. However, we can easily decipher the code by counting the number of zeros between two consecutive ones.

Problem 3

We have

$$0.01001110110_{(2)} = 0.3076171875_{(10)}.$$

We decode the sequence using the following procedure:

```
x=0.3076171875; // the message in decimal format\
low=0; // initial lower interval bound
high=1; // initial upper interval bound
n=4; // the number of characters in the code
while (n>0) do  // repeatedly narrow the interval around x
{
  determine which of the following intervals contains x:
    (1): [low, 9/10*low+1/10*high),
    (2): [9/10*low+1/10*high, 7/10*low+3/10*high),
    (3): [7/10*low+3/10*high, high) ;
  output a,b,c for intervals (1),(2),(3) respectively;
  low=low boundary of the interval containing x;
  high=high boundary of the interval containing x;
  n--;
}
```

We implement this idea in Mathematica:

```
Clear[Code] (* executes one step of the coding *)
Code[low_, high_, x_] := {a, low, 9/10*low + 1/10*high} /;
    x < 9/10*low + 1/10*high
Code[low_, high_, x_] := {b, 9/10*low + 1/10*high, 7/10*low + 3/10*high} /;
    And[x >= 9/10*low + 1/10*high,  x < 7/10*low + 3/10*high]
Code[low_, high_, x_] := {c, 7/10*low + 3/10*high, high} /;
    x >= 7/10*low + 3/10*high
Clear[CodeF] (* executes decoding using Code *)
CodeF[low_, high_, x_, 0] := {}
CodeF[low_, high_, x_, n_] :=
  Module[{aux1 = Code[low, high, x]},
      Join[{aux1[[1]]}, CodeF[aux1[[2]], aux1[[3]], x, n - 1]]
      ] /; n > 0
```

Finally, we find that the code that corresponds to $x = 0.01001110110_{(2)}$ is **caba** :

```
In[57]:=
CodeF[0, 1, 0.3076171875, 4]
Out[57]=
{c, a, b, a}
```

The entropy for the given model is $H = 1.15678$, so one would expect an average 4-bit message to take $4 * H + 2 = 6.62712$ bits of information. However, our message contains two a's and one b who all have low probability and therefore a large self-information. Hence, the sum of self-informations is relatively high for our specific message caba. By Theorem 3.3.1. from the handouts, it is the sum of self-informations that determines the length of the code: the length of the code is bounded by $2 + \sum_{i=1}^{n} s_i = 11.480357$. To sum up, our code is relatively long (11 bits) because the corresponding message caba contains a relatively high amount of self-information.

Problem 4(a)

Define $n := \max_{w \in C} l(w)$.

Let us construct a full binary tree of depth n (i.e. leaves are at depth n, whereas root is at depth 0) and let us mark left-ward edges with 0 and right-ward edges with 1, exactly as we did in the construction of Huffman code trees. Then, exactly as with Huffman codes, to each node (except the to root) corresponds a certain code. For each $w \in C$ let us define $\mathcal{L}(w)$ to be the set of all leaves that are descendants of the node that corresponds to the code w. Then for $w \neq v$, the sets $\mathcal{L}(w)$ and $\mathcal{L}(v)$ must be disjoint. In the opposite case, a common leaf x would be a descendant of both w and v and therefore one of w, v would be a descendant of the other, contradicting the fact that C is a prefix code. The total number of leaves is 2^n, so we have:

$$2^n \leq \sum_{w \in C} |\mathcal{L}(w)| = \sum_{w \in C} 2^{n-l(w)}.$$

By cancelling 2^n, we obtain the Kraft-McMillan inequality.

Problem 4(b)

We will prove this by contradiction. Assume that a prefix-free code C has 2^k code words–one shorter than k bits, at most one longer than k bits, and the rest at most k bits long. Let the one longer than k bits be called w.

Then,

$$\sum_{(s,w') \in C} 2^{-l(w)} \geq 2^{1-k} + 2^{-l(w)} + (2^k - 2) \times 2^{-k} = 2^{1-k} + 2^{-l(w)} + 1 - 2^{1-k} = 1 + 2^{-l(w)} > 1$$

This contradicts the Kraft-McMillan inequality and thus such a code cannot exist.