PROBLEM 1.

(a) We have $\rho(X_1^\infty) = 0$. We show this by showing that $\rho(X_1^\infty) \leq \delta$ for any $\delta > 0$. To see the last statement, build an invertible FSM which "recognizes" a string of type "ab...ab" for a particular even length, call it L, and outputs lets say "0" at the end of this string and returns to the starting state. Hence this machine will output an infinite string of "0" when the input is X_1^∞. From each state (including the starting state) of the chain which recognizes the special string make an edge back to the starting state in the case the next input is not the correct one. The output for each such edge is $1 + \lceil \log L \rceil$ bits long, the first bit is 1 to indicate that it is not the special path and on the next $\lceil \log L \rceil$ bits we give the index of the state (in binary representation) from which the return edge is drawn. This machine is clearly lossless and has a compressibility of $1/L$ for the desired sequence.

(b) A machine as described above will have $\rho_M(X_1^\infty) = 1/4$. In fact, one cannot do better than this. Consider a cycle, when from a given state we get back to the same state. During such a cycle we have to output at least one symbol, because the machine has to be information lossless. In an L state machine we eventually create such a cycle within at most L steps. This means that we output at least one symbol for every L input symbols, so $\rho_M(X_1^\infty) \geq 1/L$.

(c) We have $\rho_{\text{LZ}} = 0$ since compressibility is non-negative and we know that the compressibility of LZ is at least as good as that of any FSM, i.e., we know that $\rho_{\text{LZ}}(X_1^\infty) \leq \rho(X_1^\infty)$.

(d) The dictionary increases by 1 every time and has size 2 in the beginning. Hence, if we look at lets say c steps of the algorithm then we need in total

$$\sum_{i=1}^{c} \lceil \log(1+i) \rceil \leq c \log(2(c+1))$$

bits to describe the output.

What are the words which we are using. Note that the parsing is a, b, ab, aba, ba, bab,... Note that in average at most every second step the length of the used dictionary word increases by 1, i.e., we have a linear increase in the used dictionary words. Therefore, if we compute the total length which we have parsed after c steps, this length increases like the square of c.

It follows that the ratio of the total number of bits used divided by the total length described behaves like $1/c$, i.e., it tends to 0.

PROBLEM 2. Let $s(m) = 0 + 1 + \cdots + (m-1) = m(m-1)/2$. Suppose we have a string of length $n = s(m)$. Then, we can certainly parse it into m words of lengths 0, 1, ..., $(m-1)$, and since these words have different lengths, we are guaranteed to have a distinct

parsing. Since a parsing with the maximal number of distinct words will have at least as many words as this particular parsing, we conclude that whenever $n = m(m-1)/2$, $c \geq m$.

Now, given n, we can find m such that $s(m-1) \leq n < s(m)$. A string with n letters can be parsed into $m-1$ distinct words by parsing its initial segment of $s(m-1)$ letters with the above procedure, and concatenating the leftover letters to the last word. Thus, if a string can be parsed into $m-1$ distinct words, then $n < s(m)$, and in particular, $n < s(c+1) = c(c+1)/2$.

From above, it is clear that no sequence will meet the bound with equality. On the other hand, an all zero string of length $s(m)$ can be parsed into at most m words: in this case distinct words must have distinct lengths.