Basic:

1. Build a Turing machine that accepts the language {$a^n b^{n+1}$}.

Solution: anbn1.tm

2. Build a Turing machine that accepts all string(a* or b*) with more a's than b's.

Solution: moreAthanB.tm

3. Build a Turing machine that flips the middle bit of a binary string of odd length, and leaves a string of even length unchanged.

Solution: flipMiddleOdd.tm

Project:

4. (a) Build a Turing machine which takes, as input, a binary number and does integer division by 2. Assume that the binary number is represented using alphabet {0,1} with least significant bit in tape cell 1. (So, if the tape is shown as being infinite to the right, then the order in which the bits are written on the tape is the reverse of the usual.) For example, if the input number is 13, then its usual binary representation is 1101, but we store this in reverse on the tape as 1011. Integer division of 13 by 2 gives 6, which in binary is 110, which when represented in reverse on the tape is 011.

Solution: halve.tm

(b) Build a Turing machine which takes, as input, a binary number and multiplies it by 3. Use the same convention for representing binary numbers on Turing machine tape that you used in part (i).

Solution: triple.tm

(c) Build a Turing Machine which takes, as input, a positive binary number (represented as above) and computes the following function.

$$f(n) = \begin{array}{l} 3n + 1, \text{ n is odd;} \\ n/2, \text{ n is even.} \end{array}$$

Solution: function.tm

5. Background:

Legal positions in 1D-Go can be scored as follows. A gap is a sequence of uncoloured vertices bounded on each side by a coloured stone or an end of the path graph. Note that a gap may have two, one, or no bounding stones, depending on where it appears in the position. The only way it can have no bounding stones is if it spans the entire path graph, so that every vertex in the path graph is uncoloured. If the gap has just one bounding stone, then it must sit at one end of the path graph. A gap is owned by Black if all of its bounding stones are Black. Similarly, a gap is owned by White if all of its bounding stones are White. A gap is neutral if it is neither owned by Black nor owned by White. So, if a gap has a Black stone on one side and a White stone on the other, then it is neutral. The only other kind of neutral gap is when the entire path of n vertices is uncoloured, so that the entire path is a neutral gap.

Build a Turing machine which takes, as input, a legal 1D-Go position (represented as a string over the alphabet {b,w,u} in the usual way) and computes the number of neutral points in that position. The output must consist solely of a string of n xs, i.e., n = xxx · · · x, where n is the number of neutral points; as usual, the rest of the tape must be blank. If the input string has no neutral points, then the output is the empty string. When the computation finishes, the tape should be entirely blank. You do not need to check that the input position is legal; just assume that it is.

Example:

position	# neutral points	Turing machine output string
uubbubwwwuu	0	{empty string}
uubuwuuuuuw	1	x
uubuwuuwuub	3	xxx
uuuuuuuuuub	0	{empty string}
uuuuuuuuuuu	11	xxxxxxxxxxx

Solution: 1dGo.tm