1. For each element $x$ of $A$ there are $n$ choices of possible outputs $f(x)$ from $B$ in building a function $f : A \to B$. Since there are $m$ elements of $A$ and these choices are made independently, there are 

$$n \times n \times \ldots \times n = n^m$$

possible functions $f : A \to B$. The instructions for a Turing machine $M$, with a finite alphabet $\Sigma$ and finite state set $Q$, are 5-tuples 

$$(q, s, t, M, r)$$

that arise from a function 

$$f : Q \setminus \{q_0\} \times \Sigma \to \Sigma \times \{L, R, C\} \times Q$$

where $(q, s) \in Q \setminus \{q_0\} \times \Sigma$ and $(t, M, r) = f(q, s) \in \Sigma \times \{L, R, C\} \times Q$. Machines are determined by their sets of instructions, so the total number of machines for $\Sigma$ and $Q$ is the total number of such functions, which is, by the earlier calculation applied in this case,

$$(3|Q||\Sigma|^{(|Q|-1)|\Sigma|}.$$ 

(a) In the case of one non-halting state and one non-blank letter, $|Q| = |\Sigma| = 2$ and this formula becomes $(3(2)(2))^2 = 144$ Turing machines.

(b) In the case of one non-halting state and two non-blank letters, $|Q| = 2$ and $|\Sigma| = 3$ and this formula becomes $(3(2)(3))^3 = 5,832$ machines.

(c) In the case of two non-halting states and one non-blank letter, $|Q| = 3$ and $|\Sigma| = 2$ and this formula becomes $(3(3)(2))^4 = 104,976$ machines.

(d) In the case of two non-halting states and two non-blank letters, $|Q| = 3$ and $|\Sigma| = 3$ and this formula becomes $(3(3)(3))^6 = 387,420,489$ machines.

2. (a) The instructions yield the following state diagram:

The machine halts and outputs a blank tape when the number of 1’s is even, and a tape that is all blank except for a single square with 1 when the number of 1’s is odd.
(b) The instructions yield the following state diagram:

![State diagram](image)

The machine halts and outputs a blank tape in all cases.

*(c) The instructions yield the following state diagram:

![State diagram](image)

The machine halts and outputs a blank tape when the number of 1’s is even. When the number of 1’s is odd, the machine first erases all of the 1’s as it moves to the right, then prints a 1 on the next blank square to the right, reverses direction, and keeps moving to the left forever, without halting, printing a 1 on each square as it goes, filling up the tape with consecutive 1’s to the left.

*3.
4. A Turing machine instruction of the form

\[(q, s, t, C, r)\]

may be simulated, without \(C\), by adding a new state \(q'\) and replacing the instruction by

\[(q, s, t, L, q')\],

and adding instructions

\[(q', s', s', R, r)\]
a for all \(s'\) in the alphabet. If we do this wherever an instruction of \(M\) uses the move symbol \(C\), we will create a new Turing machine \(M'\) whose instructions only employ move symbols \(L\) and \(R\). The halting, input and output behaviour of \(M'\) is identical to those of \(M\), because the instruction \((q, s, t, C, r)\) will be performed in \(M\) if and only if, at the corresponding point of the computation, the pair of instructions \((q, s, t, L, q')\) and \((q', t, t, L, r)\) will be performed in \(M'\), with identical effects on symbols on the tape.

5. Let \(a\) and \(b\) be positive integers. The following machine takes an input of the form

\[1 \ldots 1 * 1 \ldots 1\]
of \(a\) consecutive 1’s, followed by an asterisk, followed by \(b\) consecutive 1’s, scanning the left-most 1, and outputs a string of \(a + b\) consecutive 1’s, scanning the left-most 1 when it halts.

```
START
1 --> R * 1 --> L b --> R
```

HALT

This has the effect of adding the positive integers \(a\) and \(b\). The asterisk is a device for inputting an ordered pair of positive integers that have been converted into strings of 1’s in the usual way.

6. Let \(m\) and \(n\) be positive integers. We design a machine that takes an input of the form

\[1 \ldots 1 * 1 \ldots 1\]
of \(m\) consecutive 1’s, followed by an asterisk, followed by \(n\) consecutive 1’s, scanning the left-most 1, and outputs a string of \(mn\) consecutive 1’s, scanning the left-most 1 when it halts. The idea is to first replace all of the left 1’s with a new symbol \(A\), the right 1’s with a new symbol \(B\) to get

\[A \ldots A * B \ldots B\],

and move back to scan the right-most \(A\). The machine then converts the \(A\)’s to \(C\)’s, one by one, each time propagating a loop that converts all of the \(B\)’s to \(C\)’s...
whilst duplicating \( n \) 1’s to the far right, with each conversion of a \( B \) into a \( C \). When it has run out of \( B \)’s the machine then converts the \( C \)’s back to \( B \)’s, to the right of the asterisk, and then moves to left-side of the asterisk to convert another \( A \) into a \( C \), propagating another loop to the right. By the time all of the \( A \)’s have been converted to \( C \)’s, there will be \( mn \) 1’s to the right. It then remains to remove any symbols from the tape that are not 1’s, finishing scanning the left-most 1.