1. - The property defined by M is that the input bit-string x is such that $(x)_2 \equiv 0 \pmod 3$, i.e. x represents a natural number divisible by 3. (The Turing machine being in state i, for $i \in \{0,1,2\}$, at a given moment corresponds to $(y)_2 \equiv i \pmod 3$ for y the part of the input read until that moment.)
   - M is total as can be seen from that the head moves to the right in every transition (and from that there's a transition for every state (0, 1, 2) and any symbol (0, 1, and blank). Since there exists a total Turing machine, namely M itself, accepting the language $L(M)$, the language is recursive. The corresponding property P, defined by $P(x)$ if $x \in L(M)$, is therefore decidable, so certainly semi-decidable.
   - For a total TM M, the language $\sim L(M)$ can be obtained by swapping its accepting and rejecting states, in this case by swapping `halt-accept` and `halt-reject`.

2. The language L comprises bit-strings consisting of 0s only. To show that the complement $\sim L$ is recursive, we exhibit a total TM K' such that $\sim L = L(K')$:

```
0 0 * r 0
0 1 * * halt-accept
0 _ * l halt-reject
```

   Note that K itself is not a total TM, so that $L = L(K)$ does not entail that L is recursive (although it is), and that simply swapping the accept and reject states of K would not yield a TM accepting the complement of L (it would rather yield a TM that accepts $\emptyset$). However, observing that K loops as soon as a 1 is detected, it suffices to let K' accept in such cases for it to accept $\sim L$. (Alternatively, one can proceed by first 'making' K total as K'':

```
0 0 * r 0
0 1 * * halt-reject
0 _ * l halt-accept
```

   observing $L(K) = L(K'')$ and then swap the reject and accept states; cf. the previous exercise.)

3. a) As a language over $\{0,1,2\}$, L_1 is recursively enumerable, as can be seen by extending M_1 with transitions rejecting in case a 2 is read.
   b) $L_1 \cup L_2$ is recursively enumerable, as can be seen by constructing a TM that simulates running both M_1 and M_2 on a given input (in parallel) and accepting as soon as one of them accepts.
   c) $L_1 \cap L_2$ is recursively enumberable, as can be seen by constructing a TM as in the previous item and accepting as soon as both accept. (Here one could opt to simulate one after the other, instead of running both in parallel as in the previous item.)
   d) $L_1 - L_2$ is in general not recursively enumerable. For instance, letting L_1 be the (recursively enumerable) set of all strings, then $L_1 - L_2 = \sim L_2$, with the latter being recursively enumerable iff L_2 is recursive (as L_2 was assumed recursively enumerable).

4$*$ M_ϵ halts on ϵ, so K accepts $M_\epsilon \# \epsilon$, and CD loops on ϵ, and indeed $cd(\epsilon) = \circlearrowleft$. Proceeding in the same way, we obtain $M_0(0) = \circlearrowleft \neq\ ! = cd(0)$, $M_1(1) = \circlearrowleft \neq\ ! = cd(1)$, $M_{00}(00) =\ ! \neq \circlearrowleft = cd(00)$, $M_{10}(10) = \circlearrowleft \neq\ ! = cd(10)$, $M_{11}(11) = \ ! \neq \circlearrowleft = cd(11)$, $M_{000}(000) = \circlearrowleft \neq\ ! = cd(000)$,

- Changing the top–right from accepting into rejecting makes no difference, since it is only relevant that the machine *halts* on x in case K *loops* on x, and both CD and this modification do so.
- Changing the bottom–right from looping into rejecting breaks the proof. E.g. the so modified version of CD would reject 00, meaning that its behaviour is to *halt*, which is the *same* as the behaviour of K on 00, so distinct from $cd(00)$.

5∗ First observe that if $(M_x)_{x \in \{0,1\}^*}$ is a family comprising *all* Turing machines, then $cd = \{x \in \{0,1\}^* \mid x \notin L(M_x)\}$ (i.e. cd is the complement of the diagonal language $d = \{x \in \{0,1\}^* \mid x \in L(M_x)\}$) is distinct from each $L(M_x)$, so cd is not a language accepted by a TM.

Now suppose the membership problem were decidable, i.e. there would exist a total TM K such that $L(K) = \{M\#x \mid x \in L(M)\}$. Then we could construct a TM CD[1] that accepts/rejects input x if K rejects/accepts $M_x\#x$. Then $x \in L(CD)$ iff K rejects $M_x\#x$ iff $x \notin L(M_x)$ iff $x \in cd$, hence $L(CD) = cd$. But, by the observation, cd is *not* a language accepted by any TM, so in particular CD cannot be a TM, and this can only be because our supposition that K be a TM was false.

[1]As follows: on input x we first compute the Turing machine M_x from its code x, then swap its accept and reject states, and finally using a universal Turing machine simulate running the resulting Turing machine on input x.