1) L_1 is recursive, because there is a program/total TM that can check for all natural numbers $2 \leqslant p \leqslant \frac{n}{2}$ whether p divides the number $n-1$, that is whether n is a prime number plus one. Because L_1 is recursive, it also is recursively enumerable.

   L_2 is recursive, since there is a program/total TM that can check whether $\ulcorner M \urcorner$ is (the code of) a TM (just like a C compiler can check whether a text is C-program) and $\ulcorner x \urcorner$ is (the code of) an input for this TM.

   L_3 is not recursive, because simulating M by means of a universal TM need not halt.

   L_4 is recursive as string search is easily programmable (even part of the standard library in most programming languages).[1]

2) a) Let K^{1000} use the universal TM U to run Turing Machine M on input x, for 1000 steps. If a halting (accept or reject) state is reached by then, then accept, otherwise reject.

   b) L_3 is r.e. since we can, for every n, simulate the TM M for n steps on all input words of length up to n and accept if any of these (finitely many) words is accepted (in n steps). Starting with $n = 0$ and successively incrementing n, if any word is accepted by M at all, it will be accepted for some n in this simulation, and then we accept M (since the language it accepts is non-empty). (Note that this does not contradict L_3 not being recursive, since for TMs M whose language *is* empty, the simulation will loop/run forever as for no n and no x, x is accepted by M within n steps.)

3) (open question so no sample solution)

4*) Let $f(M\#x) = M_1\#M_2$, where M_1 is a TM, that accepts all (well-formed) inputs, and M_2 is a TM, that simulates M on x and then accepts.

$$\begin{aligned} M\#x \in \text{HP} &\implies M \text{ halts on } x \implies M_2 \text{ halts on all inputs} \implies L(M_1) = L(M_2) \\ &\implies M_1\#M_2 \in L \implies f(M\#x) \in L \\ M\#x \notin \text{HP} &\implies M \text{ does not halt on x} \implies M_2 \text{ does not halt on any input} \\ &\implies L(M_1) \neq L(M_2) \implies M_1\#M_2 \notin L \implies f(M\#x) \notin L \end{aligned}$$

   Thus, we reduced HP to L, i.e. HP $\leq L$. Therefore, L is not recursive either.

[1] L_4 is even accepted by a so-called *finite automaton*, a machine-model that is far weaker than TMs; because of that the language L_4 is called a *regular* language. Alternatively, this follows from that the language is denoted by a so-called *regular expression*: $L_4 = \mathsf{L}((\mathbf{0{+}1})^*\mathbf{0011}(\mathbf{0{+}1})^*)$, where the regular expression $(\mathbf{0{+}1})^*\mathbf{0011}(\mathbf{0{+}1})^*$ denotes all strings that consists of an arbitrary repetition (*) of zeros or ($+$) ones, followed by zero zero one one, followed by another repetition of zeros or ones.