1.

Union

Assume there is a Turing Machine M_1 that accepts the language L_1 with the alphabet $\Sigma_1 = \{a_1, a_2 \dots a_n\}$, and some Turing Machine M_2 that accpets the language L_2 with the alphabet $\Sigma_2 = \{b_1, b_2 \dots b_m\}$. By the Turing thesis we know that both M_1 and M_2 can be represented by a single tape single head machine. We can then construct $M_1 \cup M_2$ by making a two tape two head machine M_3 where the first tape is M_1 and the second tape is M_2 and $\Sigma_3 = \Sigma_1 \cup \Sigma_2$. The set of states and the set oftransitions will also be the union of the states and transitions in M_1 and M_2. For any input w you can run the two heads on w and if either of them reach their halting state then w will be accepted. By the Turing thesis we know that this multi tape Turing Machine can be converted into a single tape single head Machine.

I would build a machine that would copy the input w and split into two tapes with one head each. one head would use the transitions from M_1 and one head would use the transitions from M_2, if either of the heads halt then w is accecpted.

Intersection

since Turing acceptable languages are proven to be closed under union, demorgans law proves that they must also be closed under intersection $\overline{\overline{M_1} \cup \overline{M_2}} = M_1 \cap M_2$

In order to create this Machine I would use two tapes each with a copy of input w, this could be done by starting with rewriting w. Then each head would use the transitions from one of the two languages, The Machine then only accepts if both heads halt.

Reversal

Give a Turing Machine M that can recognize the language L a Machine M_r can be constructed that will recongnize $L^{Я}$. Given input $w \in L^{Я}$ on the tape M_r could just move to the other side of w which would essentially create $w^{Я}$ then from there M_r can follow the same process as M to recognize the language since $w^{Я} \in L$.

2.

if you have a Turing Machine M_L that can accept a language L and a Turing Machine M_K that can accept a language K then you can construct a machine M_{LK} that accpets

K concatenated onto L. This Turing machine will have some input w that can be split into two partitions xy where $x \in L, y \in K$, this can be done non determinantly so that a copy of the TM can be made that partitions the input at every symbol and if any of them halt then the input is an instance of LK

3.

a)

transition function:

$\delta : K \times \Sigma \times \Gamma_1 \times \Gamma_2 \to (K \cup \{h\}) \times (\Sigma \cup \{L, R\})$

configuration:

$(Q \cup h) \times \Sigma^* \times \Sigma \times (\Sigma^*(\Sigma \backslash \{\#\}) \cup \epsilon) \times \Gamma_1 \times \Gamma_2 : (q, x, a, y, u, w)$ meaning the machine is in state q with the head on a everything to the right of head being x, everything to the left of the head being y, u on the top of Γ_1 and w on the top of Γ_2

yields in one step:

$(q_1, x_1\underline{a_1}y_1, u\gamma, w\gamma) \vdash (q_2, x_2\underline{a_2}y_2, \Gamma_1 * \gamma, \Gamma_2 * \gamma)$ where the difference between the x, a, y are only their positions on the string.

language accepted:

M accepts L iff $\forall w \in L, (s, \#w\#, \gamma, \gamma) \vdash^* (h, \#w\#, \Gamma_1 *, \Gamma_2 *)$ basically all the strings must halt.

b)

its the same as a turing maching but i dont know how to prove that formally.