1

This should be your own intuition on the Recursion Theorem. Mine (Andy's) uses the idea that we can use the lambda calculus to describe Turing machines which input and output the encodings of other Turing machines. That is, we can write AB to mean "the Turing machine which A outputs the description of when run on input $\langle B\rangle$" and we can write $\lambda X.Y$ to mean "the Turing machine which outputs $\langle Y\rangle$ when run on input $\langle X\rangle$". Now we can describe the "fixed-point combinator"

$$Y = \lambda f.(\lambda x.f(xx))(\lambda x.f(xx))$$

On input g, this will give

$$Yg = (\lambda x.g(xx))(\lambda x.g(xx)) = g((\lambda x.g(xx))(\lambda x.g(xx))) = g(Yg)$$

so Y takes the encoding of g to some encoding of a Turing machine which is a fixed point of g.

Now, for any computable function $t : (\Sigma^*)^2 \to \Sigma^*$, there is a Turing machine T which computes it. Now consider the Turing machine A_x which, on input w, writes $\langle x, w\rangle$ onto the tape and then simulates T. Finally, consider the machine B which, on input z, outputs $\langle A_z\rangle$.

Running Y on $\langle B\rangle$ gives a fixed-point F of B, that is a Turing machine so that B outputs $\langle F\rangle$ when run on input $\langle F\rangle$. But B outputs $\langle A_{\langle F\rangle}\rangle$ when run on $\langle F\rangle$, so we have $F = A_{\langle F\rangle}$. Then on input x, F writes $\langle\langle F\rangle, x\rangle$ onto the tape and simulates T. Thus F outputs $t(\langle F\rangle, x)$ on input x.

2

(a) This fails the first condition of Rice's Theorem. Assume some Turing machine M deciding the language $\{0\}$. Consider the machine N which is M with a useless state added. Then M and N are equivalent, but cannot both have even or odd numbers of states, so they cannot both be in the language.

(b) This also fails the first condition. Consider the Turing machine M which obtains its own description and accepts iff the input is equal to $\langle M\rangle$, so $L(M) = \langle M\rangle$. Now consider the Turing machine N which is M with a useless state added. Then $M \equiv N$, but $\langle M\rangle \in L(M)$ while $\langle N\rangle \notin L(M) = L(N)$.

(c) Rice's Theorem is applicable. The first condition is met since if $L(M) = L(N)$, then $L(M)$ contains ww iff $L(N)$ does. The second condition is met since Turing machines which accept everything and which reject everything are in and out of the language, respectively.

(d) This fails the second condition of Rice's Theorem, since Rice's Theorem gives that the language of part (c) is undecidable, so no Turing machine is in the language.

3

We define P_x to be the machine, as in lecture, which on any input, returns x; moreover, we define it to be such a machine where $|\langle P_x\rangle| > |x|$ (this is computable by adding unreachable states until it is large enough). M is the Turing machine which performs the following steps on any input:

1. Obtain its own encoding $\langle M\rangle$.

2. Return $\langle P_{\langle M \rangle} \rangle$.

Then $N = P_{\langle M \rangle}$ outputs $\langle M \rangle$ on any input, and its encoding is output by M on any input. Moreover, $|\langle N \rangle| > |\langle M \rangle|$, so they cannot be equal.

4

Consider a Turing machine M which never accepts or rejects any input, say by looping on a single state. Swapping the accept or reject states, it still never enters either, and loops on all input. Thus M is a fixed point of t.