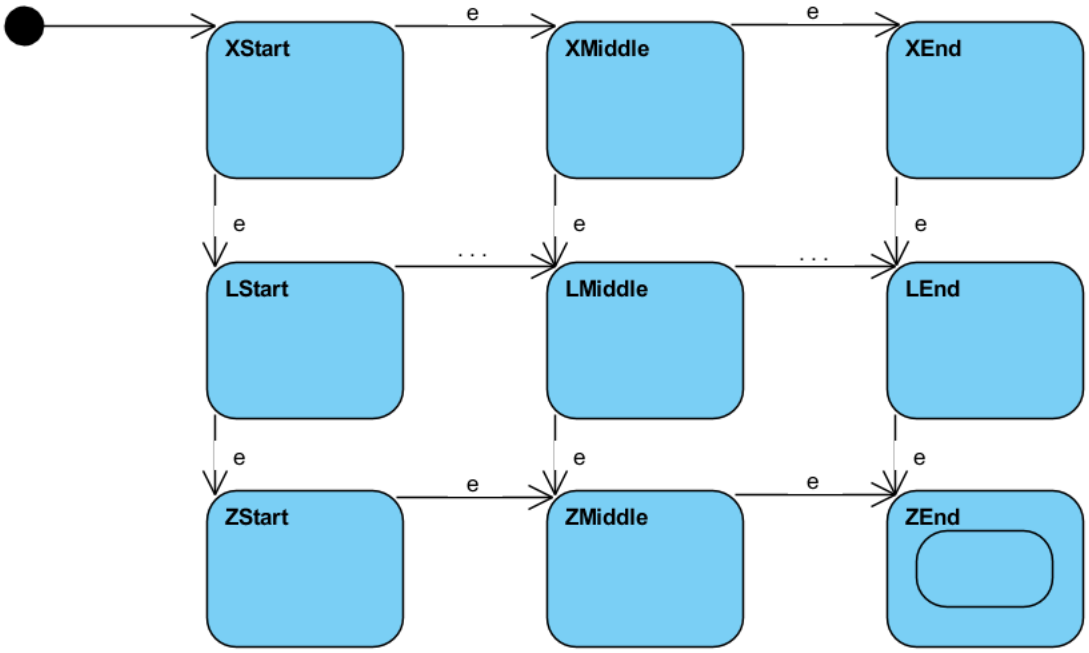
Michael Wang
HW 8

1a)
Yes, the language is ambiguous since S_1 can go to both (S) and () while S can go to ε, so to possible productions of () are $S_1 \rightarrow (S) \rightarrow ()$ and $S_1 \rightarrow ()$

1b)
$\underline{()}()() \rightarrow \underline{S_1}()() \rightarrow S\underline{()}() \rightarrow \underline{SS_1}() \rightarrow S\underline{()} \rightarrow \underline{SS_1} \rightarrow S$

2)



The DFA for L is represented by the states LStart, LMiddle, and Lend. For each state in the DFA for L, there are two new states in the NFA labeled X- and Z-. The NFA starts in the first state of X-, proceeds through the X states skipping symbols, jumps to the the appropriate L-state when the time is right somewhere in the middle of the string, then jumps to the Z states when it's ready to start skipping symbols again. This NFA ensures that only symbols at the beginning and end of the string are skipped while keeping intact those in the middle.

3)
The language P appears to be recursive, i.e. we will always be able to accept or reject an input string. This can be done by searching the list of prime numbers generated by M_{PE} until we find a number which matches the input, in which case we accept, or we encounter a number larger than the input in which case we reject. Since M_{PE} writes numbers in strictly increasing order, we will always encounter a larger number after a finite number of searches so our knew Turing Machine will never hang.