5.3

$$\frac{ab}{abab}\quad \frac{ab}{abab}\quad \frac{aba}{b}\quad \frac{b}{a}\quad \frac{b}{a}\quad \frac{aa}{a}\quad \frac{aa}{a}$$

5.9:

Let T = { <M> | M is a TM that accepts w^r whenever it accepts w }.

Assume T is decidable and let decider R decide T. Reduce from Atm by constructing a TM S as follows:

```
S: on input <M,w>
        1. create a TM Q as follows:
                On input x:
                        1. if x does not have the form 01 or 10 reject.
                        2. if x has the form 01, then accept.
                        3. else (x has the form 10), Run M on w and accept if M accepts w.
        2. Run R on <Q>
        3. Accept if R accepts, reject if R rejects.
```

Because S decides Atm, which is known to be undecidable, we then know that T is not decidable.

5.12:

Let L = { <M> | M is a TM and M writes a blank symbol over a nonblank symbol during the course of it's computation on any string }

Assume L is decidable and R be a decider for L. Reduce from Atm by constructing a TM S as follows:

```
S: on input <M,w>
        1. create a TM Q based on M as follows:
                On input w:
                        1. Replace all transitions in M that read or write a blank symbol to
                        read or write a new tape symbol #
                        2. Replace each accept transition with three operations to write a
                        nonblank symbol, move the head back to this position, write a
                        blank position over the same space, and then go to the accept state.
                        3. Simulate the above modified M on w
        2. Run R on <Q>
        3. Accept if R accepts, reject if R rejects.
```

Note that Q simulates M and will only write a blank space if and only if M accepts w because of our new tape symbol #. Because S decides Atm, which is known to be undecidable, we then know that L is not decidable.

5.15:

Let L = { <M,w> | M is a TM that attempts to move its head left at any point during the computation on w. }

To show that this is decidable, construct a TM S that will decide it.

```
S: on input <M,w>
        1. compute the number of states of M, denoted n
        2. Simulate M on input w for at most length(w) + n + 1 steps.
                If M goes left during the simulation, accept
        3. If M did not go left during the simulation, reject
```

S decides L because it is bounded by a finite computation length. Note that the above length is sufficient, for after passing through length(w) states and moving, in the worst case, to the right each time, then all further computation will be spent reading blank symbols (having passed the input w length). Now, because the input symbol will be constant hereafter during the simulation while reading to the right, we can consider what will happen over the following n+1 states. These n+1 states incur the longest possible transition sequence the machine can traverse and still go to unique states. Thus, if the machine does not move left during this number of transitions, it will then loop (or may already be looping) through the states on the same input, and will loop forever moving right.

5.17

PCP over a unary alphabet is decidable as follows:

Because the tiles are all of the same alphabet, we can just represent each tile as (m,n) where m is the number of, say, 1's in on the top sequence and n is the number of 1's on the bottom sequence. Let S be a TM.

```
S: on input {$tile_1$, $tile_2$, $tile_3$, $tile_4$,…, $tile_k$}
        If there exists $tile_i$ such that $m_i \geq n_i$ and $tile_j$ such that $m_j \leq n_j$ then accept
        Otherwise, reject
```

S will always terminate, thus it is a decider. To show that it decides PCP over unary symbols, note that the above set of tiles will not have a satisfying correspondence if all of the mi's are greater than all of the ni's, or vice-versa. In effect, this means that there will always be too many elements on the top sequence (or bottom) and no possible way to even out the strings. If there is an element of with at least as many on top as on the bottom, and on with at least as many on bottom as on top, then using those two tiles alone will provide a satisfying correspondence. In particular, given tilei and tilej, then

m_i-n_i tile_j and n_j-m_j tile_i is a solution.

Show that the language REINCF = {(G, RE)| G is a CFG and RE is a regular expression, and L(RE) is contained in L(G)} is undecidable.

To show that REINCF is undecidable, let R be a decider for REINCF. We will reduce from ALL_{CFG} = { <G> | G is CFG and L(G) = Σ* }, which is undecidable, constructing a decider S as follows:

```
S: on input <G>
        1. Run R on ( <G>, Σ* )
        2. Accept if R accepts, reject if R rejects
```

Note that S will always halt because R is a decider. Also, S decides ALLCFG because if all binary strings are contained in L(G), then L(G) = Σ* because L(G) is then maximal.