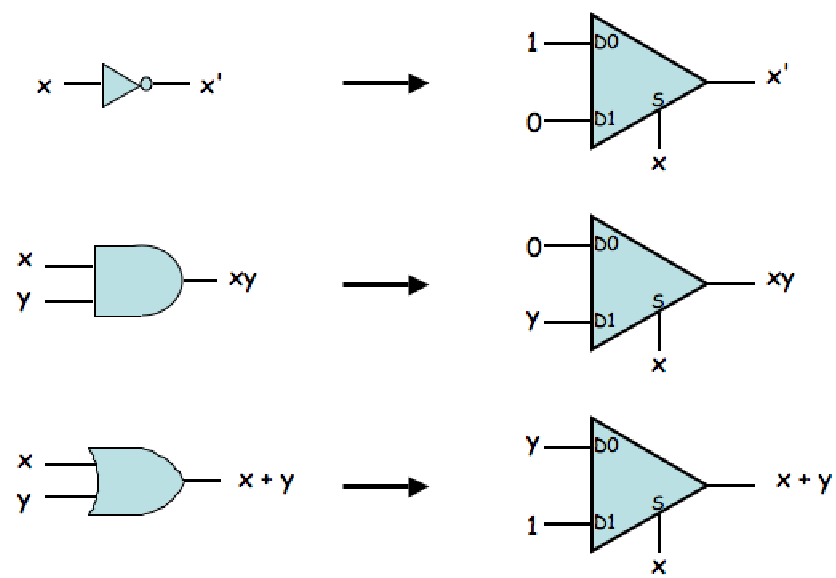
1. **Combinational Circuits**

(a) Truth Table

S	D0	D1	output
0	0	0	0
0	0	1	0
0	1	0	1
0	1	1	1
1	0	0	0
1	0	1	1
1	1	0	0
1	1	1	1

(b)



2. **Regular Expressions, Deterministic Finite State Automata (6 points)**

   a) The answer, iii) generates all desired strings and only desired strings.

      i) can generate a string that starts with b.

      ii) cannot generate a single a.

      iv) can generate a string that starts with b.

      v) cannot generate a single a.

   b) The answer, i) accepts all desired strings and only desired strings.

      ii) accepts the empty string.

      iii) accepts strings that start with b.

3. **Linked Lists (6 points)**

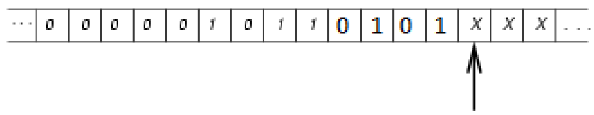
   (a)
   - i) returns true
   - ii) returns true
   - iii) returns false
   - iv) returns false

   (b) `linky_dink` returns true for a null-terminated linked list. It returns false for a circular linked list, even if the circular part is preceded by a straight path.

   (c) N

   For a null terminated linked list, `b` will traverse each node once before the method returns true. For a circular linked list, `b` which is traveling twice as quickly as `a`, will catch up to `a` in a constant number of circuits of the length N list.

4. **Turing Machine (4 points)**

a) 

b) The Turing Machine subtracts 1 from the binary number on the tape.

5. **Data Structures (3 points)**

   (a) Symbol Table

   (b) Binary Search Tree

   (c) Stack

6. **Theory True or False (6 points)** Circle your answer.

T (a) P is the set of search problems solvable in Polynomial time by a deterministic Turing Machine.

F (b) NP is the set of search problems not solvable in Polynomial time by a deterministic Turing Machine.

F (c) For proper encapsulation, instance variables should always be declared public.

F (d) Because the Halting Problem is unsolvable, it is impossible to tell if *your* TSP program for Assignment 6 has an infinite loop.

T (e) A Universal Turing Machine can compute anything that any other Turing Machine could possibly compute.

T (g) If P equals NP, then the Traveling Salesperson Problem can be solved in polynomial time by a deterministic Turing Machine.

F (h) If P does not equal NP, then there is no case of the Traveling Salesperson Problem for which you can find the optimal tour in polynomial time.

F (j) Factoring is known to be in NP but has not been proven to be NP-complete, so the discovery of a polynomial-time algorithm for factoring would mean that P equals NP.

F (k) Factoring is known to be in NP but has not been proven to be NP-complete, so no polynomial-time algorithm for factoring is possible.