1. **Combinational Circuits**

```
(a) decimal  x2 x1 x0  |X|>2
       0      0  0  0    0
       1      0  0  1    0
       2      0  1  0    0
       3      0  1  1    1
      -4      1  0  0    1
      -3      1  0  1    1
      -2      1  1  0    0
      -1      1  1  1    0
```

```
(b)  x2'x1x0 + x2x1'x0' + x2x1'x0

(c)  The circuit does not match the function in parts (a) and (b).

     The following input to the circuit will incorrectly result in
       a 0 output: 100.

     The following inputs will incorrectly result in a 1 output:
       001 and 010.
```

2. **Regular Expressions, Deterministic Finite State Automata**

   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**

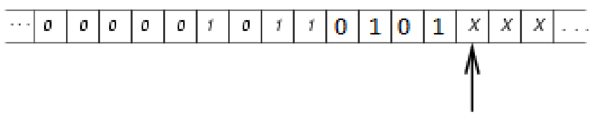
   (a)
   - i) returns true
   - ii) returns true
   - iii) returns false
   - iv) returns false

   (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**

a) 

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

5. **Data Structures**

(a) Symbol Table
(b) Binary Search Tree
(c) Stack

6. **True or False** 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.

7. **Number Systems**

(a) 012C
(b) FFC7
(c) 163

8. **TOY Programming**

(a) 0000

The program repeatedly reads integers from standard input and XORs them together. It terminates when it reads the value 0000. Recall that XORing a bit with itself always yields 0. Thus, $a \wedge a = 0$ for any integer a.

(b) ACDC

Observe that the XOR of a sequence of integers is independent of the order in which you do it. That is $a \wedge b \wedge a \wedge b = a \wedge a \wedge b \wedge b = 0$. Thus, all the integers cancel each other out except ACDC.