1) A)G1:

b)

reached = {a}

pop a off the queue

distance[a] = 0

add unreached neighbors of a, i.e. {b, d}, to reached

reached = {b, d}

pop b off the queue

distance[b] = 1

add unreached neighbors of b, i.e. {e} to reached

reached = {d, e}

pop d off the queue

distance[d] = 1

add unreached neighbors of d, i.e. {c} to reached

reached = {e, c}

pop e off the queue

distance[e] = 2

e has no unreached neighbors

reached = {c}

pop c off the queue

distance[c] = 2

exit returning 2 as distance to c from a

c) 2 edges is the shortest path from a to c (i.e. a, ad, d, dc, c)

There are 8 different paths in this graph with 2 edges. They are as follows:

1. a, ad, d, d, dc, c

2. a, ad, d, db, b

3. a, ab, b, bd, d

4. a, ab, b, be, e

5. b, ba, a, ad, d

6. b, be, e, ec, c

7. e, eb, b, bd, d

8. e, ec, c, cd, d

2) An edge with one vertex is a loop. This is a simple graph for which the associated relation is reflexive. G5 = (V5, E5) where V5 = {a} and E5 = {{a}}

G5:

3)a) G2:

b) yes. All vertices are connected if we did away with the edges directions.

c) No. d can’t reach any of the other vertices. Subgraph of G2 that only includes V = {a,b,e} and E = {(a, b), (b, e), (e, a)} could be strongly connected because there are directed paths from every vertex to each other.

4)a) G3:

b) spanning tree of G3: (found by simply inspecting the graph)

5)a) G4:

b) Yes. c, cb, b, bd, d, da, a, ab, b, bf, f, fa, a, ae, e, ec, c, ed, d, de, e, ef, f.

c) No. Because f and c have an odd number of edge connections you have to start at one and end at the other. You would need an even number of edge connections at all vertices to get a Eulerian tour.

6) I enjoyed working with and learning the pseudo code and algorithm dealing with first breadth searching graphs. I disliked the repetitiveness in going through every line for first breadth search then spanning trees.