Data Structures 2018
Exercise 11, solutions (Week 47)

1. See Graphs.java file (file does not convert).
2. a) The rows are for starting vertices, columns for end vertices, i.e. the edges on the first row are edges that go out of A.

	A	B	C	D	E	F	G	H	I	J	K	L
A		AB		AD								
B			BC		BE							
C	CA											
D						DF				DJ		
E										EJ		
F							FG					
G				GD								
H						HF	HG					
I									IH		IK	
J										JI		
K											KK	
L												

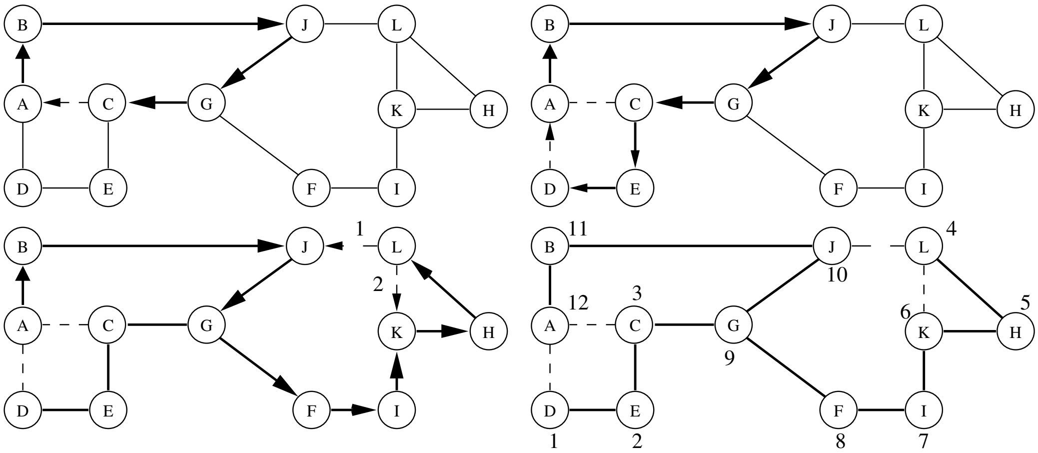
   b) In the edge list AB, AD, BC, BE, CA, DF, DJ, EJ, GD, HF, HG, IH, IK, JI, KK and in each edge a reference to the vertices that the edge connects. The vertex list includes the vertices A,...,L.

   c) In addition to an edge list, each vertex has a reference to the outgoing and incoming edges.

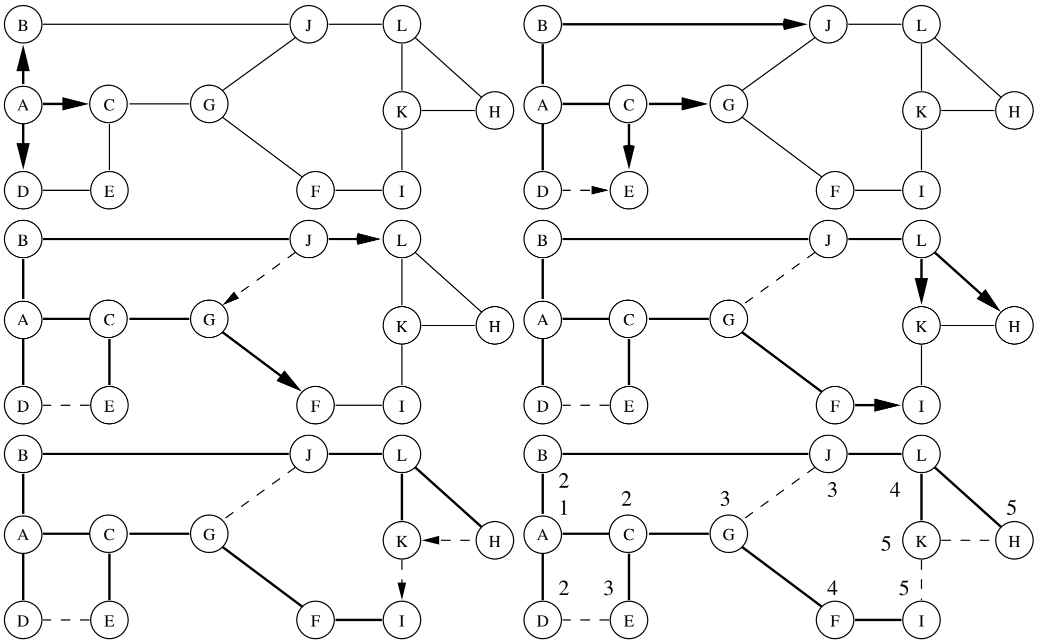
vertex	IN	OUT	vertex	IN	OUT
A	CA	AB, AD	G	FG, HG	GD
B	AB	BC, BE	H	IH	HF, HG
C	BC	CA	I	JI	IH, IK
D	AD, GD	DF, DJ	J	DJ, EJ	JI
E	BE	EJ	K	KK, IK	KK
F	DF, HF	FG	L		

3. a) No, L is not connected.
   b) No, the graph contains cycle
   c) C,e,A,e,B,e,E,e,J,e,I,e,K and C,e,A,e,D,e,J,e,I,e,K
4. a) e.g. $V' = \{A, B, C\}, E' = \{AB, BC\}$
   b) The graph has two components: $\{L\}$ and $V - \{L\}$ (the rest of the vertices).
   c) ABC, DFG, DJIHG, DJIHFG, KK.

5. a) See picture below. The discovery edges are boldened and solid. The back edges are dashed. In the last picture, the vertices are numbered according to the order in which the DFS terminates (vertex 1 was the first vertex from which we could not move on).



b) See picture below. The discovery edges are boldened and solid. The cross edges are dashed. The numbers in the last picture indicate the BFS round (or level) in which the vertex was discovered.



6. a) e.g. $V' = \{B, A, D\}, E' = \{AB, AD\}$ (must be connected and must not have cycles)

   b) e.g. $V' = \{B, A, D, J, L, K\}, E' = \{AB, AD, BJ, LK\}$ (must not have cycles, can be disconnected)

   c) For example, remove edges AD, AC, GJ and HL.

7. Algorithm IsCyclic is based on depth-first search and finds out if there are cycles in the given directed graph. The second part of the algorithm CycleDFS finds out if there are cycles in the graph on the paths starting from the node v. IsCyclic repeats the algorithm CycleDFS for all unvisited nodes until the whole graph is searched.
   The basic idea in the algorithm is to mark the nodes that have been visited and using depth-first search go through all the nodes that can be reached from the current node using edges. If we can somehow get back to the same node, there is a cycle in the graph. This can be tested by marking the nodes on the current search path. If the search finds a node which is in a cycle, the node won't be marked finished until we have found the cycle. Note that it is not possible that a node which is in a cycle would be marked finished if the cycle is not found.
   IsCyclic (G)
   precondition: $G = (V, E)$ is a graph, which has nodes $V = \{v_1, v_2, \ldots, v_n\}$ and edges $E = \{e_1, e_2, \ldots, e_m\}$.
   postcondition: Returns true, if there is a cycle in the graph, otherwise returns false.

```
mark the graph nodes V unvisited
cyclic ←false
i ← 1
while i ≤ n and not cyclic do /*Going through the components of the graph*/
   if v_i is unvisited then
      cyclic ←CycleDFS(v_i)
   end if
   i ← i + 1
end while
return cyclic
```

   CycleDFS (v)

```
mark node v unfinished
let k_1, k_2, ..., k_j be all the edges starting from node v
cyclic ←false
i ← 1
while i ≤ j and not cyclic do
   if the destination w of edge k_i is unvisited then
      cyclic ←CycleDFS(w)
   else if the destination w of edge k_i is unfinished then
      cyclic ←true
   end if
```

  $i \leftarrow i + 1$
end while
mark node v visited
return cyclic