Feedback — Exercises: Directed Graphs

You submitted this quiz on **Sun 31 Mar 2013 11:44 PM PDT -0700**. You got a score of **3.00** out of **3.00**.

To specify an array or sequence of values in an answer, you must separate the values by a single space character (with no punctuation and with no leading or trailing whitespace). For example, if the question asks for the first ten powers of two (starting at 1), the only accepted answer is:

1 2 4 8 16 32 64 128 256 512

If you wish to discuss a particular question and answer in the forums, please post the entire question and answer, including the seed (which is used by the course staff to uniquely identify the question) and the explanation (which contains the correct answer).

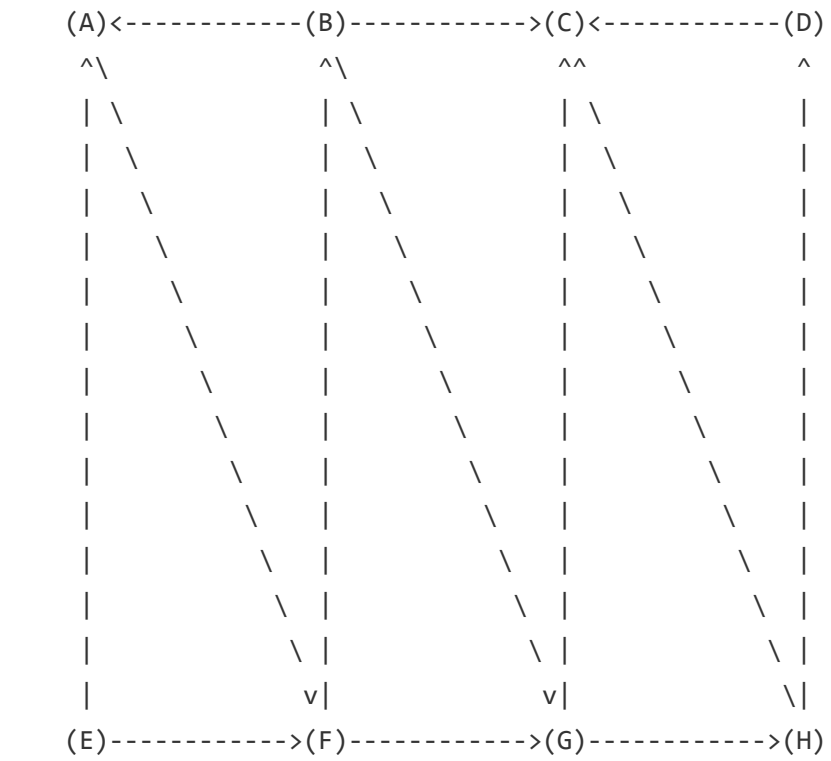
Question 1

(seed = 155645)
Consider the adjacency-lists representation of a digraph with 8 vertices and 13 edges:

```
A:  F
B:  C G A
C:
D:  C
E:  F A
```

```
F:  B G
G:  C H
H:  D C
```

Here is a graphical representation of the same digraph:



Run breath-first search (using the adjacency-lists representation), starting from vertex A.
Give the sequence in which the vertices are dequeued from the FIFO queue.

You entered:

A F B G C H D

Your Answer		Score	Explanation
A F B G C H D	✓	1.00	
Total		1.00 / 1.00	

Question Explanation

The correct answer is:

A F B G C H D

Here is a trace of the breadth-first search:

```
enqueue A
dequeue A
   enqueue F
dequeue F
   enqueue B
   enqueue G
dequeue B
   enqueue C
   check G
   check A
dequeue G
   check C
   enqueue H
dequeue C
dequeue H
```

```
   enqueue D
   check C
dequeue D
   check C


Here are the shortest paths and distances:

A to A (0):  A
A to B (2):  A->F->B
A to C (3):  A->F->B->C
A to D (4):  A->F->G->H->D
A to E (-):  not connected
A to F (1):  A->F
A to G (2):  A->F->G
A to H (3):  A->F->G->H
```

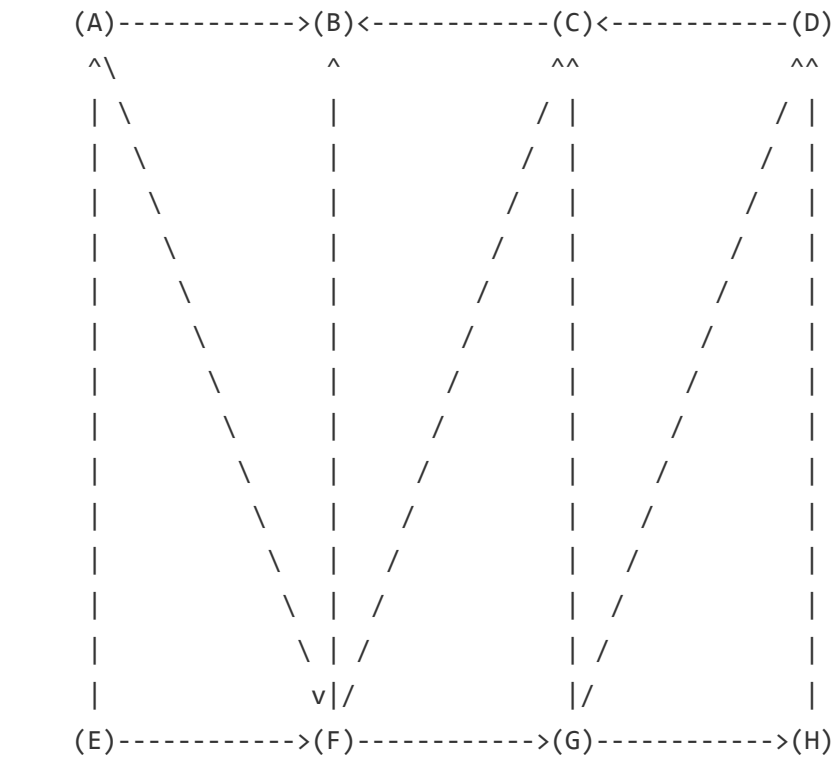
Question 2

```
(seed = 741738)
Consider the adjacency-lists representation of a DAG with 8 vertices and 13 edges:

   A:  B F
   B:
   C:  B
   D:  C
   E:  F A
   F:  B C G
```

G: D C H
H: D

Here is a graphical representation of the same DAG:



Give the topological order of the vertices that results from the DFS-based topological sort algorithm. As usual, perform the first DFS from vertex A.

You entered:

E A F G H D C B

Your Answer		Score	Explanation
E A F G H D C B	✓	1.00	
Total		1.00 / 1.00	

Question Explanation

The correct answer is:

```
E A F G H D C B
```

```
Here is a trace of the depth-first search:

dfs(A)
  dfs(B)
  B done
  dfs(F)
    check B
    dfs(C)
      check B
    C done
    dfs(G)
      dfs(D)
        check C
      D done
      check C
      dfs(H)
        check D
      H done
    G done
```

```
  F done
A done
check B
check C
check D
dfs(E)
  check F
  check A
E done
check F
check G
check H


The postorder is the order in which the vertices are done. The reverse postorder
provides a topological order.
```

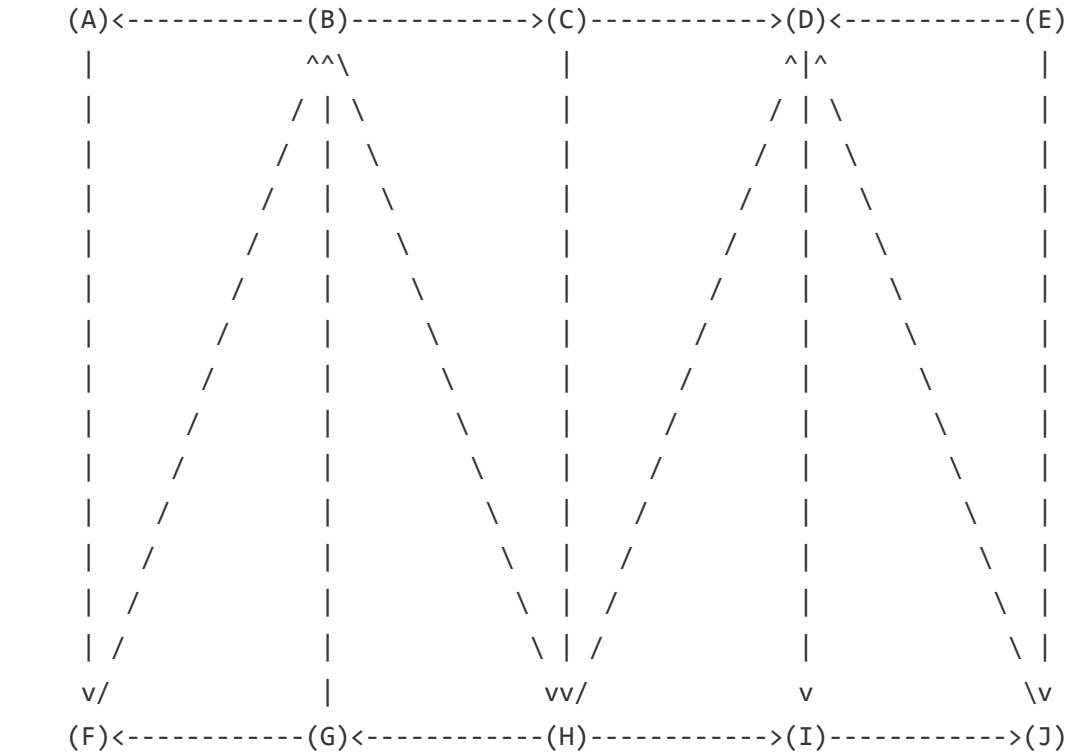
Question 3

```
(seed = 237868)
Consider the adjacency-lists representation of a digraph G with 10 vertices and 17 edges:

   A:  F
   B:  A H C
   C:  H D
   D:  I
   E:  J D
   F:  B
```

```
G:  B F
H:  I G D
I:  J
J:  D
```

Here is a graphical representation of the same digraph G:



Compute the strongly-connected components of the digraph using the Kosaraju-Sharir algorithm.
Assume that the first depth-first search of Kosaraju-Sharir computes the reverse postorder of G^R:

```
D J E I A B F G H C
```

Give the sequence of values in the id[] array for the vertices A through J.

```
   v    A  B  C  D  E  F  G  H  I  J
-----------------------------------
id[v]
```

You entered:

2 2 2 0 1 2 2 2 0 0

Your Answer		Score	Explanation
2 2 2 0 1 2 2 2 0 0	✓	1.00	
Total		1.00 / 1.00	

Question Explanation

The correct answer is:

2 2 2 0 1 2 2 2 0 0

```
   v    A  B  C  D  E  F  G  H  I  J
-----------------------------------
id[v]   2  2  2  0  1  2  2  2  0  0

The second depth-first search considers the vertices in the following order:

    D J E I A B F G H C
```

Here is a trace of the second depth-first search:

```
strong component 0
------------------
dfs(D)
  dfs(I)
    dfs(J)
      check D
    J done
  I done
D done
------------------

check J

strong component 1
------------------
dfs(E)
  check J
  check D
E done
------------------

check I

strong component 2
------------------
dfs(A)
  dfs(F)
    dfs(B)
      check A
      dfs(H)
```

```
        check I
        dfs(G)
          check B
          check F
        G done
        check D
      H done
      dfs(C)
        check H
        check D
      C done
    B done
  F done
A done
------------------

check B
check F
check G
check H
check C
```