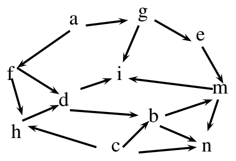
CSCD320 Homework6, Winter 2024, Eastern Washington University, Spokane, Washington.

Name: **EWU ID:**

Please follow these rules strictly:

1. Write your name and EWUID on **EVERY** page of your submission.
2. Verbal discussions with classmates are encouraged, but each student must independently write his/her own solutions, without referring to anybody else's solution.
3. The deadline is sharp. Late submissions will **NOT** be accepted (it is set on the Canvas system). Send in whatever you have by the deadline.
4. Submission must be computer typeset in the **PDF** format and sent to the Canvas system. I encourage you all to use the LaTeX system for the typesetting, as what I am doing for this homework as well as the class slides. LaTeX is a free software used by publishers for professional typesetting and are also used by nearly all the computer science and math professionals for paper writing.
5. Your submission PDF file must be named as: **firstname_lastname_EWUID_cscd320_hw6.pdf**
   (1) We use the underline '_' not the dash '-'.
   (2) All letters are in the lower case including your name and the filename's extend.
   (3) If you have middle name(s), you don't have to put them into the submission's filename.
6. Sharing any content of this homework and its keys in any way with anyone who is not in this class of this quarter is NOT permitted.

Problem 1 (20 points)**.** *Print the BFS and DFS (the sequences of letters being visited over the course of the search) that starts from the vertex* d *of the following graph. If a node has multiple next-hops, then search the next-hops in the order of their vertical coordinates from the lower ones to the higher ones. For example, node* b *has a lower vertical coordinate than the node* i*. (Note that the textbook's DFS algorithm tries all vertices as the starting node, but here you only need to show the print of the DFS that starts from the vertex d).*



Problem 2 (20 points)**.** *Given the adjacency list representation of an unweighted graph $G = (V, E)$, give your pseudocode that constructs the matrix representation of G. Describe the time complexity of your algorithm in the big-oh notation and make the bound as tight as possible.*

Problem 3 (20 points)**.** *The DFS algorithm that we discussed in class uses the adjacency list representation of a graph $G = (V, E)$ and its time cost is $O(|V| + |E|)$. Suppose you are only given the matrix representation of G, describe your pseudocode for DFS of G using the matrix representation. Give the time complexity of your algorithm in the big-oh notation and make the bound as tight as possible. Did you learn why we used the adjacency list representation for DFS in class ? Note: you can assume $matrix[i, j] = 1$ if $(i, j) \in E$; $matrix[i, j] = 0$, otherwise.*

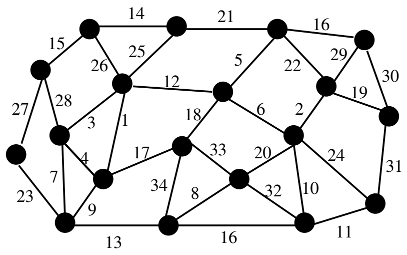
CSCD320 Homework6, Winter 2024, Eastern Washington University, Spokane, Washington.

Name: **EWU ID:**

Problem 4 (20 points). *In class, we have discussed how to use DFS for topological sorting of a DAG. Someone tries to come up with something new by trying to use BFS for topological sorting of a DAG. Below is his/her code. Does his/her code work ? If yes, explain why ? If no, give a counter example.*

```
time = 0; //global clock

//G: the graph. s: the node to start with
graph_BFS(G,s)
{
   time = time + 1;       //New operation
   s.start_time = time;  //New operation
   s.visited = true;
   FIFO.enqueue(s);

   while(FIFO.size > 0)
      u = FIFO.dequeue();
      print(u);
      time = time + 1;     //New operation
      u.end_time = time;  //New operation

      for every (u,v) \in E
         if v.visited == false
            v.visited = true;
            time = time + 1;       //New operation
            v.start_time = time;  //New operation
            FIFO.enqueue(v);
}

BFS(G) {
   for each vertex u \in G.V
      u.visited = false;

   for each u \in G.V
      if u.visited = false
         graph_BFS(G,u)
}

topological_sort(G)
{
   1. call BFS(G) to compute finishing time u.end_time for each vertex u
   2. as each vertex is finished, insert it onto the *END* of a linked list
   3. return the linked list of vertices
}
```

Problem 5 (20 points). *Show a minimum spanning tree (MST) of the following connected undirected graph by using any method. You don't have to trace the algorithm that you use, but instead you can just show the MST by making the tree edges in a different color.*


14
21
16
15
25
26
5
22
29
30
12
19
27
28
3
1
18
6
2
4
17
33
20
24
7
34
8
32
10
31
23
9
13
16
11