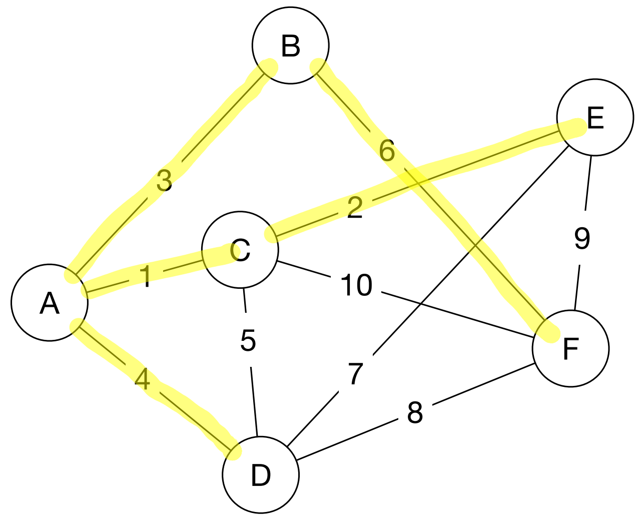
Name & Student Number:

1. Find a Minimum Spanning Tree (MST) in the following graph. (Highlight edges in your MST.)



2. Answer each of the following questions as True or False, and in one line justify your answer.
   (a) A MST contains a cycle.

   False. MSTs are acyclic graphs

   (b) If we remove an edge from a MST, the resulting subgraph is still a MST.

   False. Resulting graph will not be connected.

   (c) If we add an edge to a MST, the resulting subgraph is still a MST.

   False. Resulting graph will contain a cycle.

   (d) If there are V vertices in a given graph, a MST of that graph contains $|V|$ edges.

   False. Answer is $|V| - 1$ edges.

3. Following is the pseudocode for Prim's algorithm to find a MST.

```
function PrimMST(Graph G)
    initialize distances to ∞
    mark source as distance 0
    mark all vertices unprocessed
    for each edge (source, v) do
        v.dist = w(source,v)
    end for
    while there are unprocessed vertices do
        let u be the closest unprocessed vertex
        add u.bestEdge to spanning tree
        for each edge (u,v) leaving u do
            if w(u,v) < v.dist then
                v.dist = w(u,v)
                v.bestEdge = (u,v)
            end if
        end for
        mark u as processed
    end while
end function
```

$O(|V|)$

at most $|V|$ times

same as in Dijkstra's $O(\log|V|)$ if using minheap

at most $|E|$ times across all $|V|$ iterations of the while loop

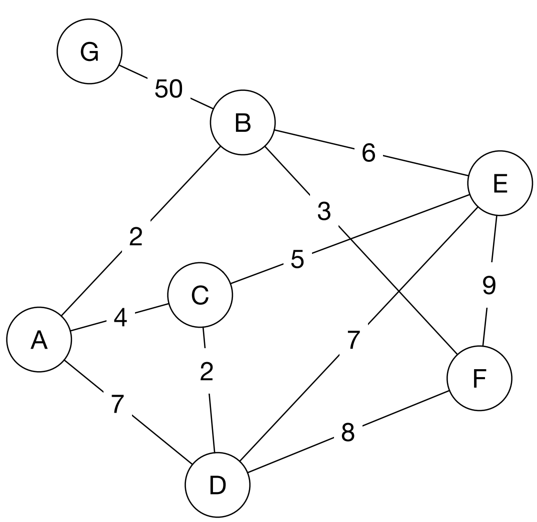
$O(\log|V|)$ to update and percolateUp in a heap

$O(1)$

Total : $O(|V|) + O(|V|\log|V| + |E|\log|V|)$

$= O(|V|\log|V| + |E|\log|V|)$

(a) Find a MST in the following graph by running the above Prim's algorithm, starting at vertex A. Fill the table.



Vertex	Distance	Best Edge	Processed	Order of Processing
A	0	-	✓	1
B	~~∞~~ 2	A	✓	2
C	~~∞~~ 4	A	✓	4
D	~~∞~~ ~~7~~ 2	~~A~~ C	✓	5
E	~~∞~~ ~~6~~ 5	~~B~~ C	✓	6
F	~~∞~~ 3	B	✓	3
G	~~∞~~ 50	B	✓	7

(b) Do a runtime analysis of Prim's MST algorithm. (Annotate the above pseudocode with your runtimes.)

Analysis of Prim's algorithm is very similar to Dijkstra's algorithm and results in the same runtime

$O(|V|\log|V| + |E|\log|V|)$