Answers to Assignment 5: Due Tuesday March 24, 2020

1. Work Problem 4.1 on page 120 of your textbook.

Each entry shows the current best distance from A of that node at that step. The colors of the table encode the data structure. A red numeral indicates that the corresponding node is in the priority queue at that step. Numbers may be changed. A black or blue numeral indicates the minimum distance from A to that node. A blue numeral indicates that the corresponding node is deleted from the priority queue at that step, and its subsequents are visited and their The shortest path tree is shown in red.

2. Work Problem 4.2 on page 120 of your textbook.

The shortest path tree is shown in red.
There are two choices for the shortest path tree.
3. You are working on computer which lacks multiplication and addition. However, it can add or subtract 1 or 2. What does this function do? What is its loop invariant?

```c
int double(int n)
// input condition: n >= 0
{
    int p = n;
    int q = 0;
    while(p > 0)
    {
        p = p-1;
        q = q+2;
    }
    return q;
}
```

The function returns $2n$. The loop invariant is: $p \geq 0$ and $2p + q = 2n$. 
4. Walk through Kruskal’s algorithm on the graph shown below. Show the union/find data structure after each step.

We execute a step for each edge, in order of weight. The step for the edge \{X,Y\} is

```java
void Step(vertex X, vertex Y)
{
    vertex U = find(X);
    vertex U = find(Y);
    if(U != V)
        union(U, V);
}
```

Each set is represented as a tree rooted at its leader. Initially, each set has rank 0 and consists of a single vertex. When we execute `union(U, V)` where U and V are distinct leaders, we combine their trees by making V the parent of U if rank(U) ≥ rank(V), or if they have equal rank and V is after U in the order of vertices, alphabetic for this case. If V retains the same rank if rank(U) ≥ rank(V); if they have equal rank, rank(V) is incremented by 1.

(a) In the initial configuration, each vertex is a leader and has rank 0.
(b) Execute Step(A,B). B is now the parent of A. A acquires rank 1. In the figures, I delete rank labels for all vertices except leaders, to indicate that those values are now irrelevant.
(c) Execute Step(C,D). D becomes the parent of C, and gets rank 1.
(d) Execute Step(B,D). D becomes the parent of B, and gets rank 2.
(e) Execute Step(A,C). Since `find(A) = find(C)`, there is no union. However, `find(A)` triggers path compression, and the parent of A becomes D.
(f) Execute Step(E,F). The parent of E becomes F.
(g) Execute Step(E,G). The parent of G becomes F, since `find(E) = F`.
(h) Execute Step(A,G). `Find(A) = D` and `find(G) = F`. Parent(F) becomes D.
(i) Execute Step(G,H). `Find(G) = D`, causing parent(G) = D by path compression, and parent(H) becomes D. There is no union. All vertices now belong to the same set, but we may continue.
(j) Execute Step(C,E). There is no union, but `find(E)` triggers path compression.
(k) Step(F,H) does not change the data structure.
(l) An edge \{X,Y\} becomes part of the spanning tree if there is a union during the execution of Step(X,Y). The resulting spanning tree consists of the emboldened edges in the last figure.