```
void kruskal(BasicGraph& graph, BasicGraph &mst) {
    // first, copy the graph and remove the edges
    // This is so we can populate it later with the mst edges
    mst = graph;
    mst.clearEdges();
    // put each vertex into a 'cluster', initially containing only itself
    Map<Vertex*, Set<Vertex*>* > clusters;
    Set<Vertex*> allVertices = graph.getVertexSet();
    Vector<Set<Vertex*>* > allSets;   // for freeing later
    for (Vertex* v : allVertices) {
        Set<Vertex*>* set = new Set<Vertex*>();
        set->add(v);
        clusters[v] = set;
        allSets.add(set);
    }

    // put all edges into a priority queue, sorted by weight
    PriorityQueue<Edge*> pq;
    Set<Edge*> allEdges = graph.getEdgeSet();
    for (Edge* edge : allEdges) {
        pq.enqueue(edge, edge->cost);
    }

    // repeatedly pull min-weight edge out of PQ and add it to MST if its
    // endpoints are not already connected
    Set<Edge*> mstEdges;
    while (!pq.isEmpty()) {
        Edge* e = pq.dequeue();
        Set<Vertex*>* set1 = clusters[e->start];
        Set<Vertex*>* set2 = clusters[e->finish];
        if (set1 != set2) {
            mstEdges.add(e);

            // merge the two sets
            set1->addAll(*set2);
            for (Vertex* v : *set1) {
                Set<Vertex*>* setv = clusters[v];
                if (setv != set1) {
                    clusters[v] = set1;
                }
            }
        }
    }

    for (Set<Vertex*>* set : allSets) {
        delete set;
    }

    // populate the graph with the edges
    // We can't add the edge pointers directly
    // because that would cause trouble freeing later
    for (Edge *edge : mstEdges) {
        mst.addEdge(edge->start->name,edge->end->name,edge->cost,false);
    }
}
```