1. (10 points) Suppose you are given a graph $G=(V,E)$ with edge weights $w(e)$ and a minimum spanning tree $T$ of $G$. Now, suppose a new edge $\{u,v\}$ is added to $G$. Describe (in words) a method for determining if $T$ is still a minimum spanning tree for $G$.

Explain how your method can be implemented to run in $O(n)$ time if both $G$ and $T$ are provided as instances of the $wgraph$ data structure.

Suppose that instead of a single edge, you are given a set of $k$ new edges to add to $G$. For small enough $k$ it makes sense to apply your algorithm repeatedly in order to update the MST, but if $k$ is “too large”, it’s more efficient to re-compute the MST from scratch. How big does $k$ have to be (as a function of $m$ and $n$) in order for this to be a better choice? Assume that the MST is computed using Prim’s algorithm with a $d$-heap, where $d=2$. 
2. (10 points) If Dijkstra’s algorithm is applied to a graph with negative length edges, some vertices may get scanned more than once. Suppose we apply Dijkstra’s algorithm to the graph shown below, starting from vertex \( a \).

List the first 7 vertices that are scanned.

In the diagram label each vertex with its tentative distance after the first 7 vertices are scanned and indicate the parent pointer of each vertex using an arrow pointing from each vertex to its parent.

List the next two vertices to be scanned.

List the next six.

What is the total number of scanning steps that Dijkstra’s algorithm will perform on this graph?

If you extended the graph by adding a fourth “diamond” to the left, with edge lengths of 8, 8, 32 and -24, how many steps would be Dijkstra’s algorithm use for this graph?
3. (10 points) Consider a Fibonacci heap containing an unmarked node $x$ for which $p(x), p^2(x), \ldots, p^9(x)$ are all marked, but $p^{10}(x)$ is not (where $p(x)$ is the parent of $x$, $p^2(x)$ the grandparent, and so forth). Suppose that a \textit{reducekey} operation is performed at $x$ that makes the key of $x$ smaller than the key of $p(x)$. Do any previously unmarked nodes become marked as a result of this operation? If so, which ones? Assume that $p^{10}(x)$ is not a tree root.

If $k$ is the number of credits needed to maintain the credit invariant, in the amortized analysis before the \textit{reducekey} operation, how many credits are needed after the operation.

Recall that during a \textit{deletemin}, the basic step involves inserting a tree root into an array, at a position determined by its rank. In some steps, the current tree root “collides” with a tree root that was inserted earlier. In other steps, there is no collision. Suppose that a \textit{deletemin} is done on this heap and that during the \textit{deletemin}, there are 20 steps during which no collision occurs. Give an expression (in terms of $\phi=(1+5^{1/2})/2$) that represents a lower bound on the number of nodes in the heap. Justify your answer.
4. (15 points) The residual graph shown below is for some flow $f$ on a flow graph $G$.

![Residual Graph](image)

What is the capacity of the edge connecting $e$ and $c$ in $G$ (note: this is asking about $G$ not $R$)? Justify your answer.

Is the edge connecting $c$ and $e$ in $G$ directed from $c$ to $e$ or from $e$ to $c$? Justify your answer. (Hint: consider the total incoming flow at $e$.) You may assume that there is no more than one edge joining any two vertices, but you should not assume anything about the direction of the edges at $s$ and $t$ (that is, $G$ may have edges entering $s$ or leaving $t$).

Find a shortest augmenting path relative to $f$ and list the vertices on the path. Draw the residual graph that results from adding as much flow as possible to this path.
5. (10 points) Let $R_f$ be a residual graph for a min-cost flow $f$, let $p$ be a source-sink path in $R_f$ with cost $c$ and let $q$ be a source-sink path in $R_f$ with cost $d$. Let $f^+$ be the flow we get when we add enough flow to $p$ to saturate it and let $R_{f^+}$ be the resulting flow graph.

Does $R_f$ contain a negative cycle? Explain your answer.

If $R_{f^+}$ has no negative cycle, what does that tell us about $p$? Explain.

If $d < c$ does $R_{f^+}$ contain a negative cycle. Explain.

If $(u, v)$ is in $R_{f^+}$ but not in $R_f$, what does that tell us about $(u, v)$? Explain.

If $(u, v)$ is in $R_f$ but not in $R_{f^+}$, what does that tell us about $\text{dist}_f(u)$ and $\text{dist}_f(v)$ (where $\text{dist}_f(x)$ is the length of the shortest path from the source $s$ to $x$ in $R_f$)?
6. (15 points) The graph at left below has negative length edges. In the diagram at right, give transformed edge lengths that preserve the relative lengths of shortest paths while eliminating negative edge lengths.

Let $G$ be an arbitrary graph with negative length edges and let $\text{length}'(u,v)$ be the transformed edge length for $(u,v)$. Suppose $p$ is a path from $x$ to $y$ with $\text{length}(p) = -8$ and $\text{length}'(p) = 5$. If $q$ is another path from $x$ to $y$ with $\text{length}(q) = 7$, what is $\text{length}'(q)$?

In the min-cost augmenting path algorithm (for the min-cost flow problem) using transformed edge costs, a new shortest path tree is computed during each step, and the shortest path distances are used to modify the edge costs. What is the total cost of the edges in the shortest path tree, using the newly modified costs? Explain.
7. (15 points) In the degree-constrained subgraph problem, we are given a graph $G=(V,E)$ and a degree bound $b(u)$ for every vertex $u$. The objective is to find a subgraph of $G$ in which every vertex $u$ has at most $b(u)$ incident edges. In the graph at right, find a degree-constrained subgraph with 6 edges. Indicate the edges in the subgraph by making them heavier weight.

In the weighted version of the problem, each edge $e$ has a weight $w(e)$ and we are interested in the degree-constrained subgraph of maximum weight. Describe (in words) an algorithm to solve this problem when the graph is bipartite. Use the graph shown at right to illustrate your solution. Note that the shapes of the vertices define the division of the vertices into subsets.