Dijkstra developed an efficient algorithm to find shortest paths on a directed graph from a designated source vertex to all other vertices, but only on graphs with non-negative edge weights.

(a) Give a clear and complete explanation of the algorithm. Be sure to cover its use of relaxation and to explain what happens if some vertices are not reachable from the source. [5 marks]

(b) Give a correctness proof for the algorithm. You may use the convergence lemma without having to prove it. [5 marks]

[Hint: here is the convergence lemma. If \( s \leadsto u \rightarrow v \) is a shortest path from \( s \) to \( v \), and at some time \( d[u] = \delta(s, u) \), and at some time after that the edge \( (u, v) \) is relaxed, then, from then on, \( d[v] = \delta(s, v) \).]

Additional hint on notation: \( s \leadsto u = \) path from \( s \) to \( u \) consisting of 0 or more edges (0 when \( s \equiv u \)); \( u \rightarrow v = \) path from \( u \) to \( v \) consisting of precisely one edge; \( d[u] = \) weight of the shortest path found so far from source \( s \) to vertex \( u \); \( \delta(s, v) = \) weight of shortest existing path from \( s \) to \( v \).]

(c) Why does the algorithm require non-negative edge weights? [2 marks]

(d) Would the algorithm work if the only negative weights were on edges leaving the source? Justify your answer with a proof or counterexample. [5 marks]

(e) Consider the following approach for finding shortest paths in the presence of negative edges. “Make all the edge weights positive by adding a sufficiently large biasing constant to each; then find the shortest paths using Dijkstra’s algorithm and recompute their weights on the original graph.” Will this work? Justify your answer with a proof or counterexample. [3 marks]