CPSC 490: ASSIGNMENT 1

*To get full marks on this assignment, you must solve **four** problems in total out of the two problems below and Problems A, C, E, F on the online judge. Each problem solved contributes 25 marks to your grade. Any extra problems you solve will be counted as bonus towards future assignments. Feel free to discuss the problems with your classmates and with me, but writing up the assignment must be done alone.*

(1) Let $G = (V, E)$ be a directed graph with **non-negative** weighted edges.

(a) (10 marks) In class we discussed a trick for computing single-source shortest paths in a graph with $\{0, 1\}$ weight edges in $\mathcal{O}(|V| + |E|)$ time. How can you apply the same trick to a graph with $\{0, 1, 2\}$ weight edges to compute single source shortest paths, again in $\mathcal{O}(|V| + |E|)$ time?

(b) (15 marks) Solve Problem B on the online judge, in any way that you want (as long as it runs in time!).

(c) (10 marks) **Bonus:** Let s be a vertex of our graph, and suppose that the distance from s to every other node in G is $\mathcal{O}(|E|)$. Can you come up with an algorithm to compute the single source shortest path from s in $\mathcal{O}(|V| + |E|)$ time?

(2) While preparing his lecture notes, Paul recently learned about the wonders of Dijkstra's algorithm. He decided that he should prepare a problem on this algorithm for all the keen students of CPSC 490. He hurried home, and was eager to code Dijkstra's algorithm to make sure he understood it (*"I just have to stay one lesson ahead of the students!"*, he thought to himself). Unfortunately, his memory was not very good, and he ended up coding this:

```
initialize dist[v] = Inf for all vertices v in V
initialize Q to be an empty queue
dist[s] = 0
Q.push(s)
while Q is not empty:
        u = Q.dequeue()
        for each vertex v adjacent to u:
                if dist(u) + wt(u, v) < dist(v) then:
                        dist(v) = dist(u) + wt(u, v)
                        if v is not in Q then:
                                Q.push(v)
```

He quickly realized his mistake, but for whatever reason, the code actually worked on all the graphs he tried! Not only that, the code also worked for graphs with both positive and negative weight edges (but without negative weight cycles). Can you help Paul figure out why?

(a) (10 marks) Prove the correctness of Paul's code for graphs with arbitrary – i.e. possibly negative – weight edges (but without negative weight cycles).

(b) (10 marks) Solve Problem D on the online judge, in any way that you want (as long as it runs in time!).

(c) (5 marks) What is the asymptotic running time of Paul's algorithm?