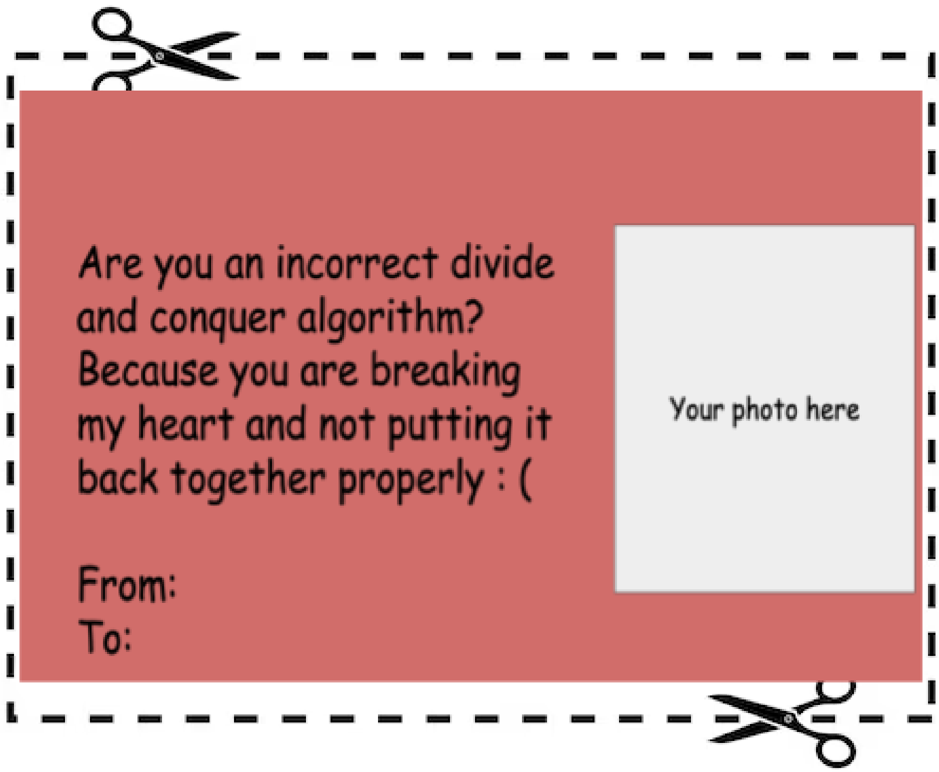
**Problem** 7.1.1**.** Let $G = (V, E)$ be an unweighted directed graph with $n$ vertices and $m$ edges. For two vertices $s$ and $t$, compute the length fo the shortest walk from $s$ to $t$ with an even number of edges. **Give an algorithm and analyze it.**

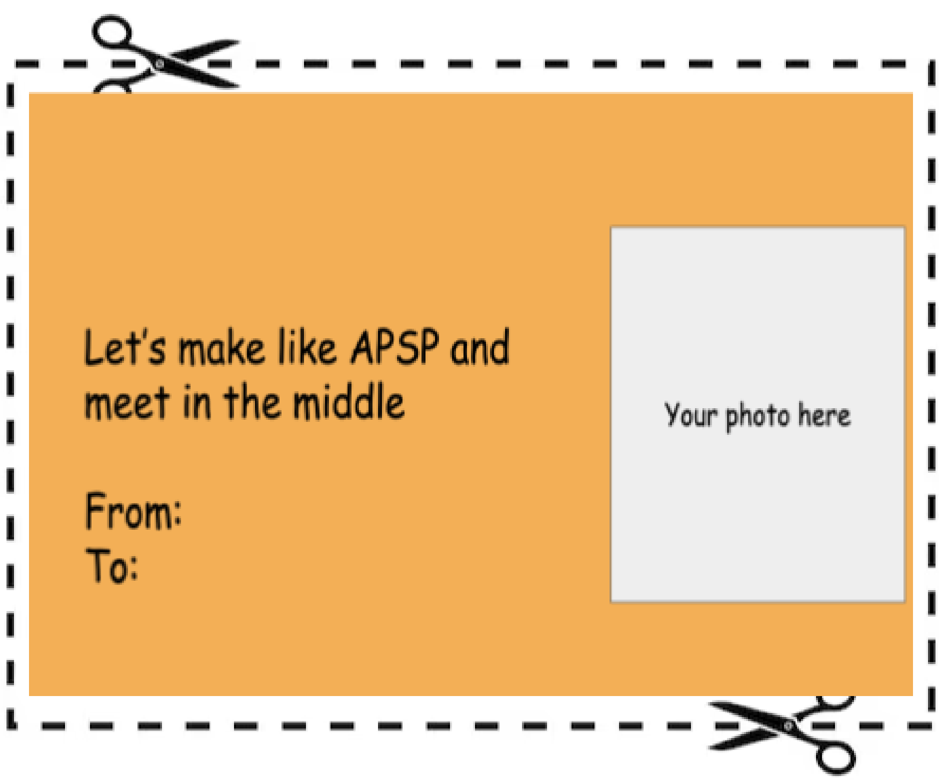
**Let's get some practice on how you would answer this on the exam.**

**Your Solution Here:**

**Problem** 7.1.1**.** Let $G = (V, E)$ be an unweighted directed graph with $n$ vertices and $m$ edges. For two vertices $s$ and $t$, compute the length fo the shortest walk from $s$ to $t$ with an even number of edges. **Give an algorithm and analyze it.**

**Now, rewrite your answer as if you were submitting it as an assignment. What details would you add? How can you make things clearer?**


Are you an incorrect divide and conquer algorithm? Because you are breaking my heart and not putting it back together properly : (
From:
To:
Your photo here


Let's make like APSP and meet in the middle
From:
To:
Your photo here

**Exercise 7.6.** Suppose you are a city planner, and you are worried that it takes too long for an ambulance to get from the university to the hospital. You've decided to build one new street to improve this particular route. You and your crack team have researched and gathered a list of potential streets that could be built; the task now is to choose one street that will improve this route the most.

To model this formally, let $G = (V, E)$ be a directed graph with positive edge lengths $\ell : E \to \mathbb{R}_{>0}$, and let $s, t \in V$ be two vertices. Let $E$ be partitioned into two nonempty disjoint sets $E_0$ and $E_1$. $E_0$ represents the edges/streets that are "already built"; $E_1$ represents the edges/streets that you can add to the graph. Let $G_0 = (V, E_0)$ denote the graph of already built edges.

Consider the problem of identifying the single edge $e \in E_1$ so that, when you add $e$ to $G_0$, the distance from $s$ to $t$ in the augmented graph is as short as possible. Design and analyze an algorithm for this problem.

## PICK A PROBLEM

**Exercise 9.8.** Let $k \geq 3$. Recall that the $k$-SAT problem is the special case of SAT where $f(x_1, \ldots, x_n)$ is a CNF with exactly $k$ distinct variables per clause.

1. Show that a polynomial time algorithm for $k$-SAT implies a polynomial time algorithm for $(k+1)$-SAT.
2. Show that a polynomial time algorithm for $(k+1)$-SAT implies a polynomial time algorithm for $k$-sat.