Midterm 2

This midterm is due **November 14** at **11:59PM**. The suggested lengths of explanations are exactly that: suggestions. You do not need the exact specified number of sentences. These suggestions exist solely to provide an idea for the expected level of detail.

Problem 2-1. [20 points] **Dependencies**

CSUCI has contracted you to create a utility which:

- takes as input a list of classes
- has access to a list of prerequisites for each class
- outputs an order in which the classes can be taken such no class is taken until after its prerequisites

How should this utility be implemented?

Problem 2-2. [40 points] **Shortest Paths**

For each of the following situations, describe a process (or name an algorithm) that could be used to efficiently find a shortest path from a node to a target. Briefly justify your answer.

(a) [10 points] Each vertex is a location, and each edge is a road. The weight of each edge is the length of the stretch of road connecting the two locations corresponding to the edge's vertices.

(b) [10 points] Each vertex is a level in the game Star Fox 64. The graph is directed and acyclic; the weight of each edge is a non-negative estimation of the difficulty of the level represented by the vertex that the edge leads to.

(c) [10 points] Each vertex represents a location, and each edge represents a path between two locations. The edge weights denote the approximate amount of food expended during the journey from one location to another. These edge weights account for the estimated availability of food along the paths; in bountiful areas, edge weights can be negative.

(d) [10 points] You want to find a shortest path containing at most twelve edges.

Problem 2-3. [40 points] **Dynamic Programming**

$C(n, k)$, often expressed as "n choose k", is the number of unique subsets of size k of a set of size n (for $0 \leq k \leq n$). $C(n, k)$ can be defined as follows:

$$C(n,k) = \begin{cases} C(n-1,k-1) + C(n-1,k) & \text{if } n > k \\ 1 & \text{if } n = k \text{ or } k = 0 \end{cases}$$

$C(n, k)$ can therefore be calculated with a naive recursive algorithm as follows:

```
C(n, k)
   IN : n, k ∈ 0, 1, 2, ..., n ≥ k
1  if k == 0 or n == k
2  |  return 1
3  else
4  |  return C(n − 1, k − 1) + C(n − 1, k)
   OUT: true if x = 2^n for some integer n (initially); false otherwise.
```

(a) [20 points] How could you improve this recursive algorithm? (Improve the provided algorithm; do not list the closed-form formula for $C(n, k)$).

(b) [20 points] What are the subproblems? What is their topological order?