Collaboration is encouraged, but students MUST write up solutions independently and list those with whom they worked. No materials from prior years’ classes or the Internet can be consulted.

1. (15 pts: 5,5,5) Consider inserting the keys 10, 22, 31, 4, 15, 28, 17, 88, 59 into a hash table of length \( m = 11 \) using open addressing with the auxiliary hash function \( h'(k) = k \mod m \). Illustrate the final result of inserting these keys using the following methods:
   
   (a) Linear Probing
   (b) Quadratic Probing with \( c_1 = 1 \) and \( c_2 = 3 \)
   (c) Double Hashing with \( h_2(k) = 1 + (k \mod (m - 1)) \)

2. (30 pts: 10,10,10) Given two text sequences \( X \) and \( Y \), let \( C(X,Y) \) denote the number of times that \( X \) appears as a subsequence of \( Y \). For example, the sequence ‘ab’ appears 4 times as a subsequence of ‘adabcb’. Let \( X_i \) denote the first \( i \) characters of the string \( X \) and let \( X[i] \) denote the \( i \)th character (similarly for \( Y \)).
   
   (a) Write a recurrence formula (not relation) for \( C(X_i,Y_j) \). For an example of a recurrence formula, see formula 15.14 on page 352 of CLRS.
   
   (b) Say \( X \) has length \( m \) and \( Y \) has length \( n \). Let \( C[i,j] \) be a 2-dimensional \( m \times n \) matrix initialized to all zeros. Describe briefly (or write pseudocode) how to convert your solution to (a) into a dynamic programming algorithm to compute \( C(X,Y) \). You may use either a bottom-up or a top-down approach.
   
   (c) What is the running time of your algorithm as a function of \( m \) and \( n \)?

3. (25 pts: 10,10,5) Consider the problem of making change for \( n \) cents using the fewest number of coins. Assume that each coin’s value is an integer.
   
   (a) Describe a greedy algorithm to make change consisting of quarters, dimes, nickels, and pennies. Prove that your algorithm yields an optimal solution.
   
   (b) Suppose that the available coins are in the denominations that are powers of \( d \) (i.e. \( d^0, d^1, d^2, \) etc. for some integer \( d > 1 \). Prove that the greedy algorithm always yields an optimal solution.)
   
   (c) Give a set of coin denominations for which the greedy algorithm does not yield an optimal solution. Your set should include a penny so that there is a solution for every \( n \).

4. (30 pts: 10,10,10) Suppose that we want to implement a stack using a dynamic array, but instead of doubling the size when the array is full, we decide to multiply the size of the array by \( 3/2 \). Assume that increasing the size of the array from \( n \) to \( 3n/2 \) costs \( n \). Thus, the actual cost of a push is 1 if the array was not full, and \( (1 + n) \) if it was full.
   
   (a) What is the amortized cost of a push?
   
   (b) Suppose that we want to make sure that the array never becomes less than \( 2/3 \) full. One way to do this is to shrink the array from size \( n \) to \( 2n/3 \) whenever we pop from an array that is \( 2/3 \) full. Does this yield an amortized cost of \( O(1) \) per operation?
   
   (c) Another approach is to reduce the size from \( n \) to \( 5n/6 \) whenever we pop from an array that is \( 2/3 \)-full. Does this yield an amortized cost of \( O(1) \)?