Homework 2
22C:196 Computational Combinatorics
Due on October 16, 2001

Part I

For the following 3 problems, you do *not* have to write code.

1. Prove or disprove: every n-permutation is the product of two n-derangements.
   (**Hint:** I sketched a proof of this claim in class, first showing how it works for transpositions and then extending it to involutions and finally to arbitrary permutations.)

2. In class we studied a recursive algorithm for generating k-subsets on $[n]$ in Gray code (minimum change) order. In this order, each k-subset can be obtained from the previous by one insertion and one deletion. Describe an algorithm that takes a k-subset and returns the next k-subset in this order.

3. How many ways are there to pay 50 cents? More specifically, suppose you have an unlimited supply of pennies (1 cent), nickels (5 cents), dimes (10 cents), and quarters (25 cents). In how many ways can you use these coins to make 50 cents?

   Let us use the generating function approach to solve this problem. Let P_n, N_n, D_n, and Q_n be the number of ways to pay n cents when we are allowed to use coins that are worth at most 1, 5, 10, and 25 cents respectively. So Q_{25} is the answer to the problem. So $P_n = 1$ for all n. Let $P(z)$, $N(z)$, $D(z)$, and $Q(z)$ be the generating functions for the sequences P_n, N_n, D_n, and Q_n respectively. First derive closed forms for $P(z)$, $N(z)$, $D(z)$, and $Q(z)$.

   Then using these generating functions and equating coefficients of like terms derive recurrence relations for P_n, N_n, D_n, and Q_n. Use these recurrences to compute Q_{25}.

4. Here is the *Mathematica* implementation of a function called `NewNumberOfPartitions` that computes $p(n,k)$ using the recurrence $p(n,k) = p(n-k,k) + p(n,k-1)$.

```
NewNumberOfPartitions[n_Integer?Positive, k_Integer?Positive] :=
  NewNumberOfPartitions[n, n] /; (k > n)

NewNumberOfPartitions[n_Integer?Positive, 0] := 0
NewNumberOfPartitions[0, k_Integer] := 1 /; (k >= 0)

NewNumberOfPartitions[n_Integer?Positive, k_Integer?Positive] :=
  NewNumberOfPartitions[n - k, k] + NewNumberOfPartitions[n, k - 1]
```

In this problem we are interested in the efficiency of this and related functions. Time the above function for values $n = 20, 25, 30, 35, 40$, with $k = n$ and report your answers. Then change the recursive part of the function to

```
NewNumberOfPartitions[n_Integer?Positive, k_Integer?Positive] :=
  NewNumberOfPartitions[n, k] = NewNumberOfPartitions[n - k, k] +
                                NewNumberOfPartitions[n, k - 1]
```

Time the function again for the same values of n and k and report your answers. What difference do you notice? How do you explain the difference? For $k = n$, what is the running time of `NewNumberOfPartitions` expressed in asymptotic notation, as a function of n? Show your calculations (not just the answer).

Part II

For the following 2 problems you have to write *Mathematica* code or perform experiments with *Combinatorica* functions. Submit a *Mathematica* notebook containing solutions to these problems.

1. Based on Problem 1 in Part I, implement a function called `DecomposeIntoDerangements` that takes an arbitrary n-permutation and returns a list of two n-permutations whose product is the given n-derangement.

2. Using the solution in Problem 2 in Part I, implement a function called `NextGrayCodeKSubset` that takes as input a k-subset of $[n]$ returns the next k-subset in Gray code order. Write a function called `NewGrayCodeKSubsets` that calls `NextGrayCodeKSubset` repeatedly to generate the list of all k-subsets of $[n]$ in Gray code order.
