Homework Assignment #4
Due: February 9, 2018 at 2:30 p.m.

1. Mr Montmort is throwing a book exchange party. Each of n guests brings one book to the party and puts it on a table. (The guests have very extensive libraries, so no two guests bring the same book.) At the end of the night, Mr Montmort will hand the books out to his guests so that no guest gets his or her own book back. Each guest gets to take this book home. (Mr Montmort himself does not participate in the book exchange.)

Mr Montmort is a very thoughtful host. So, he wants to ensure that his guests are happy with the books they receive. During the party, he asks each guest to assign a value between 0 and 100 to all of the books on the table. A value of 0 means the guest doesn't like the book at all, and 100 means the guest would love to take the book home.

Suppose the guests are numbered $1..n$ and the books they bring are also numbered $1..n$ (where book number i was brought by guest number i). Let $V[i,k]$ be the value that guest i gives to book k. Let $B[i]$ be the book assigned to guest i. Mr Montmort wants to choose $B[1..n]$ to maximize $\sum_{i=1}^{n} V[i,B[i]]$ subject to the following constraints.

$$\begin{array}{lcll} B[i] & \neq & B[j] \text{ if } i \neq j & \text{(No two people take the same book)} \\ B[i] & \neq & i & \text{(Nobody gets the book he or she brought)} \end{array}$$

First, Mr Montmort calculates the number of different ways the books can be distributed to satisfy the constraints above as follows. Let $D(n)$ be the number of ways the books can be distributed. It is fairly easy to see that $D(1) = 0$ and $D(2) = 1$.

Now suppose $n > 2$. Guest number n gets some book $i < n$. We consider two cases.

Case 1: Guest i gets book n. Then, the other $n-2$ books (all books except i and n) must be distributed to the other $n-2$ guests (all guests except i and n) so that no guest gets his or her own book. There are $D(n-2)$ ways to do this.

Case 2: Guest i does not get book n. Such an assignment of books can be done in two phases. In phase 1, guests i and n swap books. In phase 2, all the guests except n do a book exchange so that nobody gets the book he or she had at the start of phase 2. (This 2-phase plan ensures that guest n gets book i, guest i gets neither book n nor book i and everybody else gets some book other than their own, as required.) The number of ways to do this is $D(n-1)$, since that is the number of ways that Phase 2 can be carried out.

Thus, the total number of ways (over both cases) is $D(n-2) + D(n-1)$. But this total counted the number of ways to do an exchange when guest n got book i. There are $n-1$ possible choices for i, so the total number of ways to do the whole book exchange is

$$D(n) = (n-1) \cdot (D(n-2) + D(n-1)).$$

Mr Montmort wants to use the argument above to write some Java code to generate a list of all of the possible arrays B that satisfy the constraints, so that he can then evaluate the total value of each one to pick the best one. (In fact he finds it more convenient to use a Java vectors instead of arrays.) He starts by writing the following pseudocode.

```
1  ASSIGNMENTS(A)
2      % Precondition: A is a non-empty vector of distinct elements
3      n ← length of A
4      if n = 1 return empty list
5      else if n = 2 return list containing one vector ⟨A[2], A[1]⟩
6      else
7          result ← empty list
8          for i ← 1..n − 1
9              % Case 1:
10             L1 ← ASSIGNMENTS(⟨A[1], A[2], ..., A[i − 1], A[i + 1], ..., A[n − 1]⟩)
11             for each array B in list L1
12                 insert A[n] after the first i − 1 elements of B
13                 append A[i] to the end of B
14             end for
15             % Case 2:
16             L2 ← ASSIGNMENTS(⟨A[1], A[2], ..., A[i − 1], A[n], A[i + 1], A[i + 2], ..., A[n − 1]⟩)
17             for each array B in list L2
18                 append A[i] to the end of B
19             end for
20             result ← result · L1 · L2 % concatenation of 3 lists
21         end for
22         return result
23     end if
24     % Postcondition: L″ is a list of D(n) vectors containing all possible rearrangements
25     %                B of A such that B[i] ≠ A[i] for all i
26 end ASSIGNMENTS
```

Mr Montmort's Java implementation of this pseudocode available from the course web page. He soon realizes that the number of possible vectors B to consider is very large, and he doesn't have enough computing power to check them all himself. So, he decides to distribute the task by writing an app that all of his guests can download onto their smart phones and run in parallel during the book exchange party. To do this, he needs a routine SELECTASSIGNMENT that outputs the kth vector B, where k is a given value between 1 and $D(n)$. For example, SELECTASSIGNMENT$(\langle 1, 2, 3, 4, 5\rangle, 9)$ should return $\langle 2, 5, 4, 3, 1\rangle$. This will allow Mr Montmort to partition the list of possible vectors among his guests and get each guest to check (using their smartphones) one section of the list. For example, guest number 2 might be assigned the range $k = 10000000, ..., 20000000$ so that guest needs to be able to compute all the vectors in that range of indices of the list produced by the ASSIGNMENTS routine *without* computing the entire list. The doorbell rings just as Mr Montmort is about to write the necessary code, so he has to go attend to his guests, and leaves the job of filling in the SELECTASSIGNMENT function to you. Make the routine as efficient as you can.

Fill in the SELECTASSIGNMENT function in the Java code provided on the course web page and submit your solution using the `submit` command. Instructions for doing this will be posted on the course web page.