Three algorithms for generating one random permutation

First algorithm:
Keep picking numbers at random, throwing out any that have already been picked.
This method is very inefficient because you have to check whether a number has already been picked, and more importantly, because it will take longer and longer to find a random number that hasn't been picked.

Second algorithm:
Instead of picking a random number 1..n, choose a random number 1..n-k, where k is the number that have already been picked. Keep a list of numbers that haven't been picked. When the random number generator chooses r, add the rth member of this list to the permutation.
Advantage: no random numbers are thrown away.
Disadvantage: Determining the rth element in the list of unchosen numbers is lots of work.

Third algorithm
\[
\begin{align*}
p[1]:=1, \ldots, p[n]:=n & \quad ; \text{initialize permutation} \\
\text{for } i = 1..n & \\
\quad r = \text{rand}(i,n) \\
\quad p[i] \leftrightarrow p[r] \\
\text{endfor} \\
\text{output: } p \\
\end{align*}
\]

algorithm for generating all permutations
found on page 351

homework: p. 354: 3,4,10,20,26-33
3,10,26,27,28 were done in class.
solutions: RAND(i,j) produces a random integer between i and j, inclusive

26. input n
   let i=rand(1,n)
   let j=rand(1,n)
   output (i,j)

27. input n
   for i=1 to n
   for j=1 to n
   output (i,j)
   endfor
   endfor

28. This algorithm does not choose all pairs with equal probability. For example, it chooses (n,n) with probability 1/n, but it chooses (1,1) with probability 1/n^2.

29. input n
   Repeat
   let i=rand(1,n)
   let j=rand(1,n)
   EndRepeat when i < j
   output (i,j)

30. input n
   Repeat
let i=\text{rand}(1,n)
let j=\text{rand}(i,n)
\text{EndRepeat when } i \leq j
\text{output } (i,j)

31. first solution: Use algorithm PERMUTE-1 on p.346, but change the for loop so that it runs from 1 to k instead of from 1 to n.
second solution: Use algorithm PERMUTE-2 on p.348, but change the for loop so that it runs from 1 to k instead of from 1 to n.
third solution: Modify algorithm PERMUTE-3 on p.349 as follows:
input n and k
for i=1 to n
    let perm[i]=i
endfor
for i=1 to k
    let r = \text{rand}(i,n)
    exchange perm[i] and perm[r]
endfor
output the first k entries of perm

32. Use the same algorithm as appears on p.352, but instead of outputting each permutation, only output the first k digits each time one of the first k digits changes.

33. Use any of the algorithms described in #31. Use the permutation produced by the algorithm as a combination. For example, if n=5 and k=3 and the algorithm produces the permutation 413, then use the combination \{1,3,4\}. 