Generating Permutations and Combinations

Generating Permutations

Generating the permutations of the \( n \) smallest positive integers and then replacing those integers with any set of \( n \) elements will create the set of permutations for that set. Often, the permutation of a set will be given in lexicographic ordering.

**Definition**

The **lexicographic ordering** for a set of permutations \( \{1,2,3,...,n-1,n\} \) has the permutation \( a_1a_2...a_n \) precede the permutation \( b_1b_2...b_n \) when, for some \( k, 1 \leq k \leq n \), \( a_1 = b_1, a_2 = b_2, ..., a_{k-1} = b_{k-1}, \) and \( a_k < b_k \).

A procedure for generating the next permutation in lexicographic order can be developed for a given \( a_1a_2...a_n \). If \( a_{n-1} < a_n \), swap the two to get the next largest permutation (..56 to ...65). If \( a_{n-1} > a_n \), then a larger permutation cannot be made from the two integers. In that case, look at the final three integers. If \( a_{n-2} < a_{n-1} \), then put the smaller of the two integers \( a_{n-1} \) and \( a_n \) in the \( a_{n-2} \) position. Fill the remaining positions in lexicographic order to complete the permutation (..165 to ...516). This procedure can be generalized to produce the next largest permutation for any \( a_1a_2...a_n \).

**Algorithm**

Generating the Next Largest Permutation in Lexicographic Order

```c
NextPermutation(a_1a_2...a_n; permutation of \{1, 2,..., n\} != to n, (n-1),...2, 1) {
    j = n -1;
    while (a[j] > a [j+1])
        {j = j - 1;}
    k = n;
    while (a[j] > a[k])
        {k = k -1;}
    swap a[j] and a[k];
    r = n;
    s = j + 1;
    while (r > s)
        {
            swap a[r] and a[s];
            r = r - 1;
            s = s + 1;
        }
}
```
Generating Combinations

For any \( r \)-combination, a procedure for creating the next largest combination can be developed. A combination \( a_1a_2...a_r \) in lexicographic order is given for a set \{1,2,3,...,n\}. In the set \{1, 2, 3, 4, 5, 6\}, a 4-combination could be \{1, 2, 5, 6\}. To obtain the next largest combination, find the last \( a_i \) in the combination so that \( a_i \neq n - r + i \). (The last element in the combination with \( a_i \neq n - r + i \) is 2.) Replace \( a_i \) with \( a_i + 1 \), and \( a_j \) with \( a_j + j - i + 1 \), for \( j = i + 1, i + 2, ..., r \). (This creates the next largest combination, \{1, 3, 4, 5\}. This algorithm is also used in the applet below.

Algorithm

<table>
<thead>
<tr>
<th>Algorithm</th>
</tr>
</thead>
<tbody>
<tr>
<td>Generating the Next Largest ( r )-Combination in Lexicographic Order</td>
</tr>
<tr>
<td>NextCombination((a_1a_2...a_r); \text{subset of } {1, 2,..., n} \neq {((n-r+1),..., n}) with ( a_1 &lt; a_2 &lt; ... &lt; a_r )</td>
</tr>
<tr>
<td>{</td>
</tr>
<tr>
<td>( i = r; )</td>
</tr>
<tr>
<td>\textbf{while} ( (a[i] &gt; n - r + i) )</td>
</tr>
<tr>
<td>{ ( i = i - 1; ) }</td>
</tr>
<tr>
<td>( a[i] = a[i] + 1; )</td>
</tr>
<tr>
<td>\textbf{for} ( (j = i + 1; j &lt;= r; j++) )</td>
</tr>
<tr>
<td>{ ( a[j] = a[i] + j - i; ) }</td>
</tr>
<tr>
<td>}</td>
</tr>
</tbody>
</table>

Your Programming Assignment:

Use the algorithms above to write your \textit{nextPermutation} and \textit{nextCombination} functions. Use these functions to write another function called \textit{allPermutations(int n, int r)}that will return all permutations of size \( r \) from the input set of size \( n \). Both \( n \) and \( r \) are input parameters entered by the user. The \( n \) elements of the input set will also be entered by the user as comma separated strings. Run your program with the following three sets of inputs.

Input set 1:
\( n = 5, r = 3 \)
\text{elements} = a, b, c, d, e

Input set 2:
\( N = 4, r = 2 \)
\text{Elements} = red, green, blue, orange, violet

Input set 3:
\( N = 3, r = 3 \)
\text{Elements} = Jim, Mary, Bill
Output format

Each valid permutation should be printed on the screen within braces and separated by commas. For example, output for Input set 1 should look as following:

\{c, d, e\}
\{d, c, e\}
\{d, e, c\}
...

Submission:

As before save your screen captures in a word document. Turn in printed output and source code in class. Turn in your soft copies using blackboard.