C-1.4

The algorithm operates recursively. We maintain two lists, bag, which holds the characters that have not yet been put into the permutation and permutation, which holds the characters that are in the permutation. Initially the bag list has all the characters and the permutation list is empty. One by one, characters are copied from the bag to the permutation and back again. When all characters of the bag have been added to the permutation, we print the result. If characters remain in the bag, we process them as follows. We iterate through each character of the current bag, remove this character and add it to the rear of the permutation. Then we recursively repeat the process on the remaining bag. On returning to the recursive calls, we remove the last element from the permutation, and put it back in the bag. In this way, each element of the bag takes turns being the last element of the permutation. We have omitted the header material, which includes the files <cstdlib>, <list>, <string>, and <iostream>.

```
using namespace std;

ostream& operator<<(ostream& out, const list<char>& L) {
  list<char>::const_iterator p = L.begin();
  while (p != L.end()) { out << *p; p++; }
  return out;
}

void permute(list<char>& bag, list<char>& permutation) {
  if(bag.empty())                          // empty bad means we're done
    cout << permutation << endl;
  else {
    list<char>::iterator p = bag.begin();  // for each element left in bag
    while (p != bag.end()) {
      list<char>::iterator n = p; n++;     // save next position
      char c = *p;                         // remove next item from bag
      bag.erase(p);
      permutation.push_back(c);            // add to back of permutation
      permute(bag, permutation);           // recursively permute the rest
      permutation.pop_back();              // remove the last element
      bag.insert(n, c);                    // ... and restore to the bag
      p++;
    }
  }
}

void printPermutations(list<char>& elements) {
  list<char> bag = elements;
  list<char> permutation;
  permute(bag, permutation);
}

main()
{
  const char elts[] = { 'a', 'b', 'c', 'd', 'e', 'f' };
  list<char> elements;
  for (int i = 0; i < 6; i++) elements.push_back(elts[i]);
  printPermutations(elements);
  return EXIT_SUCCESS;
}
```