```
// file1.cpp
//
// Marius Rejdak
// Informatyka, mgr, OS1

/*

Running:
#!/bin/fish
g++ -Wall -std=c++11 -o3 -o file1 file1.cpp; and ./file1

Output:
qsort(std::vector) time: 0.047021s
std::sort(std::vector) time: 0.062702s
std::stable_sort(std::vector) time: 0.077105s
std::list::sort() time: 0.07842s
std::sort(std::vector) for reversed strings time: 2.1847s

*/

#include <iostream>     // std::cout
#include <fstream>            // std::ifstream
#include <functional> // std::function
#include <iterator>           // std::istream_iterator
#include <algorithm>    // std::sort, std::stable_sort, std::random_shuffle
#include <vector>       // std::vector
#include <string>             // std::string
#include <ctime>        // time, clock, clock_t, CLOCKS_PER_SEC
#include <cstdlib>      // srand, qsort
#include <list>                       // std::list

//using namespace std; // http://stackoverflow.com/questions/1452721/why-is-using-namespace-std-
considered-bad-practice

template <class T>
float get_time(T& container, std::function<void(T&)> sorting_function) {
	clock_t clockStart = clock();
	sorting_function(container);
	return ((float)clock()-clockStart)/CLOCKS_PER_SEC;
}

int main(int argc, char *argv[]) {
	std::ifstream in("lab1.dic", std::ifstream::in);
	std::istream_iterator<std::string> in_iterator(in), eos;
	std::vector<std::string> lines(in_iterator, eos); // all lines from file

	srand(unsigned(time(0))); // for std::random_shuffle

	std::random_shuffle(lines.begin(), lines.end());
	std::list<std::string> lines_list(lines.begin(), lines.end()); // randomised list

	//
	// Zad 1
	//

	//
	// qsort(std::vector)
	//
	//std::random_shuffle(lines.begin(), lines.end()); // is already random
	std::cout << "qsort(std::vector) time: " << get_time<std::vector<std::string> >(lines
		, [](std::vector<std::string> &x) -> void
		{
			qsort(&x[0]
				, x.size()
				, sizeof(std::string)
				, [](const void *x, const void *y) -> int
					{
						return ((std::string*)x)->compare(*(std::string*)y);
					});
		}) << "s\n";
	//
	// std::sort(std::vector)
	//
	std::random_shuffle(lines.begin(), lines.end());
	std::cout << "std::sort(std::vector) time: " << get_time<std::vector<std::string> >(lines
		, [](std::vector<std::string> &x) -> void
		{
			std::sort(x.begin(), x.end());
		}) << "s\n";

	//
	// std::stable_sort
	//
	std::random_shuffle(lines.begin(), lines.end());
	std::cout << "std::stable_sort(std::vector) time: " << get_time<std::vector<std::string> >(lines
		, [](std::vector<std::string> &x) -> void
		{
			std::stable_sort(x.begin(), x.end());
		}) << "s\n";

	//
	// std::list::sort()
	//
	//std::random_shuffle(lines_list.begin(), lines_list.end()); // cannot apply to list, is already
random
	std::cout << "std::list::sort() time: " << get_time<std::list<std::string> >(lines_list
		, [](std::list<std::string> &x) -> void
		{
			x.sort();
		}) << "s\n";

	//
	// Zad 2
	//

	//
	// std::sort(std::vector) for reversed strings
	//
	std::random_shuffle(lines.begin(), lines.end());
	std::cout << "std::sort(std::vector) for reversed strings time: " <<
get_time<std::vector<std::string> >(lines
		, [](std::vector<std::string> &x) -> void
		{
			// Reverse string sorting function
			std::function<bool(std::string, std::string)> reverse_sorting = [](std::string i,
std::string j) -> bool
			{
				return std::string(i.rbegin(), i.rend()) < std::string(j.rbegin(),
j.rend());
			};

			std::sort(x.begin(), x.end(), reverse_sorting);
		}) << "s\n";

	// For testing
	//for(std::string s : lines)
	//{
	//	std::cout << s << std::endl;
	//}

	return 0; //Huge success!
}
```

```
// file2.cpp
//
// Marius Rejdak
// Informatyka, mgr, OS1

/*

Running:
#!/bin/fish
g++ -Wall -std=c++11 -o3 -o file2 file2.cpp; and ./file2

*/

#include <iostream>           // std::cout, std::endl
#include <vector>             // std::vector

//
// Zad 3
//
template <class BidirectionalIterator>
bool next_combination(BidirectionalIterator first1, BidirectionalIterator last1, BidirectionalIterator
first2, BidirectionalIterator last2)
{
	bool b = false;
	BidirectionalIterator vi_it = last1-1, vc_it;

	for(auto it1 = last2-1; it1 != first2 || it1 == first2; --it1, --vi_it)
	{
		if(*it1 == *vi_it)
		{
			if(it1 != first2)
			{
				b = true;
				vc_it = it1-1;
				continue;
			}
			else
			{
				return false;
			}
		}
		else
		{
			if(b)
			{
				auto tmp = it1;
				for(auto it2 = first1; it2 != last1; ++it2)
				{
					if(*vc_it == *it2)
					{
						tmp = it2+1;
						break;
					}
				}

				for(auto it3 = vc_it; it3 != last2; ++it3, ++tmp)
				{
					*it3 = *tmp;
				}

				return true;
			}
			for(auto it2 = first1; it2 != last1; ++it2)
			{
				if(*it1 == *it2)
				{
					*it1 = *(++it2);
					return true;
				}
			}
		}
	}
	return true;
}
```

```cpp
template <class T>
void show_collection(T &c)
{
	for(auto i : c)
	{
		std::cout << i << " ";
	}
	std::cout << std::endl;
}

int main(int argc, char *argv[]) {

	char tab [10] = {'A', 'B', 'C', 'E', 'G', 'I', 'M', 'O', 'P', 'Y'};
	std::vector<char> vi(tab, tab+10);
	std::vector<char> vc(tab, tab+4);

	do
	{
		show_collection(vc);
	}
	while(next_combination(vi.begin(), vi.end(), vc.begin(), vc.end()));

	return 0; //Huge success!
}
```