```cpp
#include <iostream>
#include <string.h>
#include <vector>
#include <sstream>
#include <iterator>
#include <algorithm>
#include "A9.h"


//////////////////////////////////////////////////////
// Assignment 9.1
// ///////////////////////////////////////////////////

// Consider the separate case when the vector contains
// only 1 element to avoid printing an empty space at
// at the end (which caused a fail in the automatic
// testing used in the course).

void reverse(std::vector<int> & v) {
    if (v.empty())
        return;
    else if (v.size() == 1) {
        std::cout << v.back();
        v.pop_back();
        return;
    }
    else {
        std::cout << v.back() << " ";
        v.pop_back();
        reverse(v);
    }
}



///////////////////////////////////////////////////////////////
// Assignment 9.2
///////////////////////////////////////////////////////////////

void fib(unsigned N, std::vector<int> & F) {
    if (N == 2)
        F.push_back(1);
    else {
        fib(N - 1 , F);
        F.push_back(F[N - 1] + F[N - 2]);
    }
}
```

```cpp

///////////////////////////////////////////////////////////////////
// Assignment 9.3
///////////////////////////////////////////////////////////////////

// PROBLEM: It scans linearly through the entire vector.
// With recursion there's really no way out of just going through
// the vector element by element, but the ideal would be to stop in
// the middle... With iterators that's not easy...
// For this task iterators are probably not the best option but I'm
// leaving this to show the reverse iterators and how to simplify a
// long declaration with 'using'.

// It is not necessary to have these here because they're
// in the header file already, but for clarity I repeat them

using IT = std::vector<int>::iterator;
using RIT = std::vector<int>::reverse_iterator;

bool palindrome(const std::vector<int> & V, IT it, RIT rit) {
    if (it != V.end() && rit != V.rend())
        if (*it == *rit)
            palindrome(V , ++it , ++rit);
        else
            return false;
    return true;
}




/////////////////////////////////////////////////////////////////////////
// Assignment 9.4
/////////////////////////////////////////////////////////////////////////

using SIT = std::string::iterator;

unsigned lev(SIT b1, SIT e1, SIT b2, SIT e2) {
    std::vector<unsigned> d{ 0 , 0 , 0 };

    if (e1 - b1 == 0)
        return (unsigned)(e2 - b2);
    else if (e2 - b2 == 0)
        return (unsigned)(e1 - b1);
    else {
        d[0] = lev(b1 + 1 , e1 , b2 , e2) + 1;
        d[1] = lev(b1 , e1 , b2 + 1 , e2) + 1;
        d[2] = lev(b1 + 1, e1 , b2 + 1 , e2) + (((*b1) == (*b2)) ? 0 : 1);

        return *min_element(d.begin() , d.end());
    }
}
```