```cpp
1  #include <iostream>
2  #include <string.h>
3  #include <vector>
4  #include <sstream>
5  #include <iterator>
6  #include <algorithm>
7  #include "A9.h"
8
9
10
11 // Assignment 9.1
12 // Consider the separate case when the vector contains
13 // only 1 element to avoid printing an empty space at
14 // at the end (which caused a fail in the automatic
15 // testing used in the course).
16
17 void reverse(std::vector<int> & v) {
18     if (v.empty())
19         return;
20     else if (v.size() == 1) {
21         std::cout << v.back();
22         v.pop_back();
23         return;
24     }
25     else {
26         std::cout << v.back() << " ";
27         v.pop_back();
28         reverse(v);
29     }
30 }
31
32
33
34 /////////////////////////////////////////////////////////////
35
36 // Assignment 9.2
37
38 void fib(unsigned N, std::vector<int> & F) {
39     if (N == 2)
40         F.push_back(1);
41     else {
42         fib(N - 1 , F);
43         F.push_back(F[N - 1] + F[N - 2]);
44     }
45 }
46
47
48
49
50
51
52
53
54
55
56
57
58
```

```cpp
59  ///////////////////////////////////////////////////////////////
60
61  // Assignment 9.3
62
63  // PROBLEM: It scans linearly through the entire vector.
64  // With recursion there's really no way out of just going through
65  // the vector element by element, but the ideal would be to stop in
66  // the middle... With iterators that's not easy...
67  // For this task iterators are probably not the best option but I'm
68  // leaving this to show the reverse iterators and how to simplify a
69  // long declaration with 'using'.
70
71  // It is not necessary to have these here because they're
72  // in the header file already, but for clarity I repeat them
73
74  using IT = std::vector<int>::iterator;
75  using RIT = std::vector<int>::reverse_iterator;
76
77  bool palindrome(const std::vector<int> & V, IT it, RIT rit) {
78      if (it != V.end() && rit != V.rend())
79          if (*it == *rit)
80              palindrome(V , ++it , ++rit);
81          else
82              return false;
83      return true;
84  }
85
86
87  ////////////////////////////////////////////////////////////////////
88
89
90  // Assignment 9.4
91
92  using SIT = std::string::iterator;
93
94  unsigned lev(SIT b1, SIT e1, SIT b2, SIT e2) {
95      std::vector<unsigned> d{ 0 , 0 , 0 };
96
97      if (e1 - b1 == 0)
98          return (unsigned)(e2 - b2);
99      else if (e2 - b2 == 0)
100         return (unsigned)(e1 - b1);
101     else {
102         d[0] = lev(b1 + 1 , e1 , b2 , e2) + 1;
103         d[1] = lev(b1 , e1 , b2 + 1 , e2) + 1;
104         d[2] = lev(b1 + 1, e1 , b2 + 1 , e2) + (((*b1) == (*b2)) ? 0 : 1);
105
106         return *min_element(d.begin() , d.end());
107     }
108 }
```