```cpp
#include <iostream>
#include <string>

using namespace std;

class SzovegMuveletek {
private:
    string szoveg1;
    string szoveg2;

public:
    SzovegMuveletek(string s1, string s2) : szoveg1(s1), szoveg2(s2) {}


    string leghosszabbAzonosResz() {
        int hossz1 = szoveg1.length();
        int hossz2 = szoveg2.length();

        // Táblázat létrehozása a közös részek hosszának tárolásához
        int** dp = new int* [hossz1 + 1];
        for (int i = 0; i <= hossz1; ++i) {
            dp[i] = new int[hossz2 + 1];
        }

        // Töltse fel a táblázatot nullákkal
        for (int i = 0; i <= hossz1; ++i) {
            for (int j = 0; j <= hossz2; ++j) {
                dp[i][j] = 0;
            }
        }

        // Dinamikus programozás: megtaláljuk a leghosszabb közös részt
        int maxHossz = 0;
        int maxHosszIndex = 0;
        for (int i = 1; i <= hossz1; ++i) {
            for (int j = 1; j <= hossz2; ++j) {
                if (szoveg1[i - 1] == szoveg2[j - 1]) {
                    dp[i][j] = dp[i - 1][j - 1] + 1;

                    if (dp[i][j] > maxHossz) {
                        maxHossz = dp[i][j];
                        maxHosszIndex = i;
                    }
                }
            }
        }

        // Felszabadítjuk a memóriát
        for (int i = 0; i <= hossz1; ++i) {
            delete[] dp[i];
        }
        delete[] dp;

```

```
54          // Visszatérés a leghosszabb közös részszel
55          return szoveg1.substr(maxHosszIndex - maxHossz, maxHossz);
56      }
57
58      string tomorites() {
59          string tomoritettSzoveg;
60          size_t hossz = szoveg1.length();
61
62          for (size_t i = 0; i < hossz; i++) {
63              char aktChar = szoveg1[i];
64              size_t count = 1;
65
66              // Számoljuk az egymást követő előfordulásokat
67              while (i < hossz - 1 && szoveg1[i] == szoveg1[i + 1]) {
68                  count++;
69                  i++;
70              }
71
72              // Ha egymást követő előfordulások több mint egy, akkor
                  adjuk hozzá a tömörített szöveghez a számot is
73              if (count > 1) {
74                  tomoritettSzoveg += aktChar;
75                  tomoritettSzoveg += to_string(count);
76              }
77              else {
78                  tomoritettSzoveg += aktChar;
79              }
80          }
81
82          return tomoritettSzoveg;
83      }
84  };
85
86  int main() {
87      SzovegMuveletek sm("alma", "almafa");
88      cout << "Leghosszabb azonos resz: " << sm.leghosszabbAzonosResz() <<
            endl;
89
90      SzovegMuveletek ha("malom", "halom");
91      cout << "Leghosszabb azonos resz: " << ha.leghosszabbAzonosResz() <<
            endl;
92
93      SzovegMuveletek sm2("maaaalommmmm", "bbba");  //csak az első szót
          tömöríti
94      cout << "Tomoritett szoveg: " << sm2.tomorites() << endl;
95
96      return 0;
97  }
98
```