```
#include <iostream>
#include <cstring>
#include <vector>
#include <sstream>

#include "EDistance.hpp"

using namespace std;

EDistance::EDistance(string _stringA, string _stringB) : stringA(_stringA
), stringB(_stringB) {

  vector<int> temp;

  // Populate the matrix with 0's to start
  for(unsigned i = 0; i < stringB.length() + 1; i++)
    temp.push_back(0);
  for(unsigned i = 0; i < stringA.length() + 1; i++)
    opt.push_back(temp);

  editDistance = optDistance();
  editString = alignment();
}

int EDistance::penalty(char a, char b) {

  if(a == b)
    return 0;
  else
    return 1;
}
int EDistance::min(int a, int b, int c) {

  if(a <= b && a <= c)
    return a;
  else if(b <= c)
    return b;
  else
    return c;
}

int EDistance::optDistance() {

  // Fill in the matrix with the EditDistances
  for(int i = opt.size() - 1; i >= 0; i--)
    for(int j = opt[i].size() - 1; j >= 0; j--) {
      if((i == opt.size() - 1) && (j == opt[i].size() - 1))
        opt[i][j] = 0;
      else if(i == opt.size() - 1)
        opt[i][j] = opt[i][j + 1] + 2;
      else if(j == opt[i].size() - 1)
        opt[i][j] = opt[i + 1][j] + 2;
      else
        opt[i][j] = min(opt[i+1][j+1] + penalty(stringA[i], stringB[j]),
                        opt[i+1][j] + 2,
                        opt[i][j+1] + 2);
    }

  return opt[0][0];
}
string EDistance::alignment() const {

  stringstream ss;

  unsigned i = 0, j = 0;
```

```cpp

  while(i < opt.size() - 1 || j < opt[0].size() - 1) {
    if((i < opt.size() - 1)
       && (j < opt[0].size() - 1)
       && (opt[i+1][j+1] <= opt[i+1][j] + 1)
       && (opt[i+1][j+1] <= opt[i][j+1] + 1)) {
      ss << stringA[i] << " " << stringB[j] << " " << opt[i][j] - opt[i+1][j+1] << '\n';
      i++;
      j++;
    }
    else if(((i < opt.size() - 1) && (opt[i+1][j] <= opt[i][j+1]))
            || (j == opt[0].size() - 1)) {
      ss << stringA[i] << " " << "-" << " " << opt[i][j] - opt[i+1][j] << '\n';
      i++;
    }
    else {
      ss << "-" << " " << stringB[j] << " " << opt[i][j] - opt[i][j+1] << '\n';
      j++;
    }
  }

  return ss.str();
}

void EDistance::printOpt() const {

  // Print the Matrix
  for(unsigned i = 0; i < opt.size(); i++) {
    for(unsigned j = 0; j < opt[i].size(); j++) {
      cout.width(4);
      cout << opt[i][j];
    }
    cout << endl;
  }
}

int EDistance::getEditDistance() const {

  return editDistance;
}

string EDistance::getEditString() const {

  return editString;
}
```