```
 1: all: ED
 2:
 3: ED: main.o ED.o
 4:         g++ -g -o ED main.o ED.o -lsfml-system
 5:
 6: main.o: main.cpp ED.hpp
 7:         g++ -g -c main.cpp -Wall -ansi -pedantic -std=c++0x
 8:
 9: ED.o: ED.cpp ED.hpp
10:         g++ -g -c ED.cpp -Wall -ansi -pedantic -std=c++0x
11:
12: clean:
13:         rm *.o ED
```

```
    1: //
    2: //  main.cpp
    3: //  ps4
    4: //
    5: //  Created by Jingxian Shi on 3/2/18.
    6: //  Copyright Â© 2018 Jingxian Shi. All rights reserved.
    7: //
    8:
    9: #include <iostream>
   10: #include <SFML/System.hpp>
   11: #include "ED.hpp"
   12: using namespace std;
   13:
   14: int main(int argc, const char * argv[])
   15: {
   16:     string a, b;
   17:     cin >> a >> b;
   18:
   19:     sf::Clock clock;
   20:     sf::Time t;
   21:     int edit_distance;
   22:
   23:     ED Matrix(a, b);
   24:     edit_distance = Matrix.OptDistance();
   25:     cout << "Edit distance = " << edit_distance << endl << Matrix.Alignment() <<
endl;
   26:
   27:     t = clock.getElapsedTime();
   28:     cout << "Execution time is " << t.asSeconds() << " seconds \n" << endl;
   29:
   30:     return 0;
   31: }
```

```
    1: //
    2: //  ED.cpp
    3: //  ps4
    4: //
    5: //  Created by Jingxian Shi on 3/8/18.
    6: //  Copyright Â© 2018 Jingxian Shi. All rights reserved.
    7: //
    8:
    9: #include "ED.hpp"
   10: #include <iostream>
   11: #include <sstream>
   12: #include <string>
   13: #include <vector>
   14: #include <iomanip>
   15:
   16: ED::ED(std::string seq1, std::string seq2)
   17: {
   18:     _seq1 = seq1;
   19:     _seq2 = seq2;
   20:
   21:     matrix.resize(_seq1.length() + 1);
   22:     for(int i = 0; i < matrix.size(); i++)
   23:     {
   24:         matrix[i].resize(_seq2.length() + 1);
   25:     }
   26: }
   27:
   28: ED::~ED()
   29: {
   30:
   31: }
   32:
   33: int ED::penalty(char a, char b)
   34: {
   35:     return a != b;
   36: }
   37:
   38: int ED::min(int a, int b, int c)
   39: {
   40:     int min = a;
   41:     if(b < min)
   42:     {
   43:         min = b;
   44:     }
   45:     if(c < min)
   46:     {
   47:         min = c;
   48:     }
   49:     return min;
   50: }
   51:
   52: int ED::OptDistance()
   53: {
   54:     for(int i = _seq1.length(); i >= 0; i--)
   55:     {
   56:         matrix[i][_seq2.length()] = 2 * (_seq1.length() - i);
   57:     }
   58:     for(int j = _seq2.length(); j >= 0; j--)
   59:     {
   60:         matrix[_seq1.length()][j] = 2 * (_seq2.length() - j);
   61:     }
   62:
   63:     for(int i = _seq1.length() - 1; i >= 0; i--)
   64:     {
   65:         for(int j = _seq2.length() - 1; j >= 0 ; j--)
```

```
   66:         {
   67:             matrix[i][j] = min(2 + matrix[i + 1][j],
   68:                                2 + matrix[i][j + 1],
   69:                                matrix[i + 1][j + 1] + penalty(_seq1[i], _seq2[j])
);
   70:         }
   71:     }
   72:
   73:     return matrix[0][0];
   74: }
   75:
   76: std::string ED::Alignment()
   77: {
   78:     int current = matrix[0][0];
   79:     int i = 0, j = 0;
   80:     std::string alignment = "";
   81:
   82:     while(!(i == _seq1.length() && j == _seq2.length()))
   83:     {
   84:         if (i == _seq1.length())
   85:         {
   86:             while (j < _seq2.length())
   87:             {
   88:                 alignment.push_back('-');
   89:                 alignment.push_back(' ');
   90:                 alignment.push_back(_seq2[j]);
   91:                 alignment.push_back(' ');
   92:                 alignment.push_back('2');
   93:                 alignment.push_back('\n');
   94:                 j++;
   95:             }
   96:             break;
   97:         }
   98:         if (j == _seq2.length())
   99:         {
  100:             while (i < _seq1.length())
  101:             {
  102:                 alignment.push_back(_seq1[i]);
  103:                 alignment.push_back(' ');
  104:                 alignment.push_back('-');
  105:                 alignment.push_back(' ');
  106:                 alignment.push_back('2');
  107:                 alignment.push_back('\n');
  108:                 i++;
  109:             }
  110:             break;
  111:         }
  112:
  113:         if(current == (matrix[i+1][j] + 2)) //below
  114:         {
  115:             alignment.push_back(_seq1[i]);
  116:             alignment.push_back(' ');
  117:             alignment.push_back('-');
  118:             alignment.push_back(' ');
  119:             alignment.push_back('2');
  120:             alignment.push_back('\n');
  121:             i++;
  122:             current = matrix[i][j];
  123:         }
  124:         else if(current == (matrix[i][j+1] + 2)) //right
  125:         {
  126:             alignment.push_back('-');
  127:             alignment.push_back(' ');
  128:             alignment.push_back(_seq2[j]);
  129:             alignment.push_back(' ');
```

```
  130:             alignment.push_back('2');
  131:             alignment.push_back('\n');
  132:             j++;
  133:             current = matrix[i][j];
  134:         }
  135:         else if(current == (matrix[i+1][j+1] + penalty(_seq1[i], _seq2[j]))) //di
agonal
  136:         {
  137:             alignment.push_back(_seq1[i]);
  138:             alignment.push_back(' ');
  139:             alignment.push_back(_seq2[j]);
  140:             alignment.push_back(' ');
  141:             alignment.push_back(penalty(_seq1[i], _seq2[j]) + '0');
  142:             alignment.push_back('\n');
  143:             i++;
  144:             j++;
  145:             current = matrix[i][j];
  146:         }
  147:     }
  148:
  149:     return alignment;
  150: }
```

```
 1: //
 2: //  ED.hpp
 3: //  ps4
 4: //
 5: //  Created by Jingxian Shi on 3/8/18.
 6: //  Copyright Â© 2018 Jingxian Shi. All rights reserved.
 7: //
 8:
 9: #ifndef ED_hpp
10: #define ED_hpp
11:
12: #include <stdio.h>
13: #include <string>
14: #include <vector>
15:
16: class ED
17: {
18: public:
19:     ED(std::string seq1, std::string seq2);
20:     ~ED();
21:     int penalty(char a, char b);
22:     int min(int a, int b, int c);
23:     int OptDistance();
24:     std::string Alignment();
25:
26: private:
27:     std::string _seq1, _seq2;
28:     std::vector<std::vector<int> > matrix;
29:
30: };
31:
32: #endif /* ED_hpp */
```