```
 1: #include "Editdistance.hpp"
 2: #include <SFML/System.hpp>
 3:
 4: int main(int argc, char* argv[])
 5: {
 6:   sf::Clock clock;
 7:   sf::Time t;
 8:   std::string test_1, test_2, align;
 9:
10:   std::cin >> test_1;
11:   std::cin >> test_2;
12:
13:   EditDistance ED(test_1, test_2);
14:
15:   int opt = ED.OptDistance();
16:
17:   std::cout << "Edit distance = " << opt << std::endl;
18:
19:   align = ED.Alignment();
20:
21:   std::cout << align;
22:
23:   t = clock.getElapsedTime();
24:
25:   std::cout << "Execution time is: " << t.asSeconds() << std::endl;
26:
27:   return 0;
28: }
```

```
 1: #ifndef EDITDISTANCE_HPP
 2: #define EDITDISTANCE_HPP
 3:
 4: #include <string>
 5: #include <vector>
 6: #include <iostream>
 7:
 8: class EditDistance
 9: {
10:  public:
11:   EditDistance(std::string string_1, std::string string_2);
12:   ~EditDistance();
13:   int penalty(char a, char b);
14:   int min(int a, int b, int c);
15:   int OptDistance();
16:   std::string Alignment();
17:
18:  private:
19:   std::vector< std::vector< int > > data;
20:   std::string x, y;
21: };
22: #endif
```

```
 1: #include "Editdistance.hpp"
 2:
 3: EditDistance::EditDistance(std::string string_1, std::string string_2)
 4: {
 5:   x = string_1;
 6:   y = string_2;
 7:   x.append("-");
 8:   y.append("-");
 9:
10:   std::vector< std::vector< int > > my_data(string_2.size() + 1,
11:                  std::vector<int> (string_1.size() + 1, -1));
12:   data = my_data;
13: }
14:
15: EditDistance::~EditDistance()
16: {
17:
18: }
19:
20: int EditDistance::penalty(char a, char b)
21: {
22:   if(a == b)
23:     {
24:       return 0;
25:     }
26:   else if ((a != b) && (a == '-' || b == '-'))
27:     {
28:       return 2;
29:     }
30:   else
31:     {
32:       return 1;
33:     }
34: }
35:
36: int EditDistance::min(int a, int b, int c)
37: {
38:   if(a <= b)
39:     {
40:       if(a <= c)
41:         {
42:           return a;
43:         }
44:       else
45:         {
46:           return c;
47:         }
48:     }
49:   else
50:     {
51:       if(b <= c)
52:         {
53:           return b;
54:         }
55:       else
56:         {
57:           return c;
58:         }
59:     }
60: }
61:
```

```
   62: int EditDistance::OptDistance()
   63: {
   64:   // This handles the outsides
   65:   int count = 0;
   66:   for(int i = x.size() - 1; i >= 0; i--)
   67:     {
   68:
   69:       data.at(y.size() - 1).at(i) = count;
   70:       count += 2;
   71:     }
   72:   count = 0;
   73:   for(int i = y.size() - 1; i >= 0; i--)
   74:     {
   75:       data.at(i).at(x.size() - 1) = count;
   76:       count += 2;
   77:     }
   78:
   79:   //this handles the inside
   80:   for(int i = x.size() - 2; i >= 0; i--)
   81:     {
   82:
   83:       for(int j = y.size() - 2; j >= 0; j--)
   84:         {
   85:           data.at(j).at(i) = min(
   86:               data.at(j+1).at(i+1) + penalty(x.at(i), y.at(j)),
   87:               data.at(j).at(i+1) + 2,
   88:               data.at(j+1).at(i) + 2);
   89:         }
   90:     }
   91:
   92:
   93:   return data.at(0).at(0);
   94: }
   95:
   96: std::string EditDistance::Alignment()
   97: {
   98:   std::string z;
   99:   unsigned int i = 0;
  100:   unsigned int j = 0;
  101:
  102:   while(i < y.size() -1 && j < x.size() - 1)
  103:   {
  104:     if(data.at(i).at(j) == data.at(i+1).at(j+1) + penalty(x.at(j),y.at(i)))
  105:     {
  106:       z.push_back(x.at(j));
  107:       z.push_back(' ');
  108:       z.push_back(y.at(i));
  109:       z.push_back(' ');
  110:       z.push_back(penalty(x.at(j),y.at(i)) + '0');
  111:       z.push_back('\n');
  112:       i++;
  113:       j++;
  114:     }
  115:     else if(data.at(i).at(j) ==  data.at(i).at(j+1) +2)
  116:     {
  117:       z.push_back(x.at(j));
  118:       z.push_back(' ');
  119:       z.push_back('-');
  120:       z.push_back(' ');
  121:       z.push_back('2');
  122:       z.push_back('\n');
```

```
123:       j++;
124:     }
125:     else if(data.at(i).at(j) ==  data.at(i+1).at(j) +2)
126:     {
127:       z.push_back('-');
128:       z.push_back(' ');
129:       z.push_back(y.at(i));
130:       z.push_back(' ');
131:       z.push_back('2');
132:       z.push_back('\n');
133:       i++;
134:     }
135:   }
136:
137:   if(j <x.size())
138:   {
139:       z.push_back(x.at(j));
140:       z.push_back(' ');
141:       z.push_back('-');
142:       z.push_back(' ');
143:       z.push_back('2');
144:       z.push_back('\n');
145:       j++;
146:   }
147:   else if(i < y.size())
148:   {
149:       z.push_back('-');
150:       z.push_back(' ');
151:       z.push_back(y.at(i));
152:       z.push_back(' ');
153:       z.push_back('2');
154:       z.push_back('\n');
155:       i++;
156:    }
157:
158:   return z;
159: }
```

```
 1: C=g++ -g -Wall --std=c++98 -Werror
 2: E=.cpp
 3: O=main.o Editdistance.o
 4: P=ED
 5: SFML= -lsfml-system
 6: all: $(P)
 7: $(P):$(O)
 8:         $(C) -o $(P) $(O) $(SFML)
 9:
10: $(E).o:
11:         $(C) -c $<  -o $@
12:
13: clean:
14:         rm $(O) $(P)
```