```
    1: #include <string>
    2: #include "ED.hpp"
    3:
    4: //constructor
    5: ED::ED(std::string str1, std::string str2){
    6:   //basic assignment
    7:   _str1 = str1;
    8:   _str2 = str2;
    9:
   10:   _str1Len = str1.size();
   11:   _str2Len = str2.size();
   12:
   13:   //allocate for the first dimension
   14:   _array = new int*[_str1Len + 1];//+1 because we assume the first column/ro
w is empty
   15:
   16:   //need to allocate 2nd dimension of array, <= because _array is (str1Len +
 1
   17:
   18: }
   19:
   20: //get the penalty for comparing the args
   21: int ED::penalty(char a, char b){
   22:
   23:   //test = b - a;
   24:   if(a == b)//check if chars are the same
   25:     return 0;
   26:   return 1;
   27: }
   28:
   29: //returns minimum of the three args, basic comparison
   30: int ED::min(int a, int b, int c){
   31:   if(a < b && a < c)
   32:     return a;
   33:   if(b < c)
   34:     return b;
   35:   else
   36:     return c;
   37: }
   38:
   39: //traverses the 2d array, the meat of the program, have to move backwards (s
tart at end)
   40: /*int ED::optDistance(){
   41:   //traverse from the end of the y-axis to the front
   42:
   43: }
   44:
   45: //traces the matrix and returns a string
   46: std::string ED::alignment(){
   47:
   48:
   49: }*/
   50:
   51: int ED::getStr1Len(){
   52:   return _str1Len;
   53: }
   54:
   55: int ED::getStr2Len(){
   56:   return _str2Len;
   57: }
   58:
```

```
59: ED::~ED(){
60:   //need to delete the memory we used for _array, this is second level
61:   //need to delete the first level
62:   delete[] _array;
63: }
```