Problem 1: String alignment
This problem relies heavily on material presented in class. Write a program to ask the user to input two strings. Use the C++ string class. Here’s a minimal example:

```cpp
#include <string>

using std::string;

int main() {
    string s;
    cin>>s;
}
```

After that, you should compute the edit distance of the two strings and display their alignment. The alignment is done by backtracking as shown in class. Use dynamic memory allocation to create the two dimensional array needed for computing the edit distance.

Problem 2: Pairs
Consider the following template class:

```cpp
template<class T, class S>
class Pair {
    T a;
    S b;

    public:
    Pair() {a=T(); b=S();}
    Pair(T x, S y) {a=x; b=y;}
    T left() {return a;}
    S right() {return b;}
};
```

Modify the backtracking in Problem 1 by using an auxiliary two dimensional array of pairs. If the original two dimensional array is called $a$ and the auxiliary is called $b$, then $b[i][j]$ contains the pair $(k, l)$ such that $a[i][j]$ was computed
from $a[k][l]$.

**Problem 3: Backtracking with pointers**

Consider the following class:

```cpp
class Entry {
    public:
        int i;
        int j;
        int value;
        Entry * back;
};
```

Repeat the work of Problem 1 with a two dimensional arrays of Entries (instead of a two dimensional array of ints). Each entry in the array stores its indices, the integer value of the edit distance, and a pointer used for backtracking for obtaining the actual alignment.