EECE 7205-Assignment 3

Sreejith Sreekumar: 001277209

October 29, 2018

1 Qn2. Longest Common Subsequence

1.1 Code

```
#include <iostream>
#include <vector>
#include <algorithm>
#include <time.h>
#include <iomanip>
#include <math.h>
#include <string>

using namespace std;

void printLCS(vector<vector<char>> _direction, string _seq1, int i, int j){

  if(i == 0 || j == 0){
    return;
  }

  if(_direction[i][j] == '@'){        //diagonal arrow, match sequence
    printLCS(_direction, _seq1, i - 1, j - 1);
    cout << _seq1[i];
  }

  else if(_direction[i][j] == '^'){  //up arrow
    printLCS(_direction, _seq1, i - 1, j);
  }

  else{         //left arrow
    printLCS(_direction, _seq1, i, j - 1);
  }

}


int longest_common_subsequence(string seq1, string seq2){
```

```
  int m = seq1.length();
  int n = seq2.length();

  //B matrix - modeled as vector of vector B[m+1][n+1];
  vector<vector<char>> direction(m+1, vector<char>(n+1, 0));

  //C matrix - modeled as vector of vector  C[m+1][n+1];
  vector<vector<char>> magnitude(m+1, vector<char>(n+1, 0));

  for(int i = 0; i <= m; i++){

    for(int j = 0; j <= n; j++){

      if(i == 0 || j == 0) {        //default fill
        magnitude[i][j] = 0;
        direction[i][j] = '/';
      }

      else if(seq1[i] == seq2[j]){
        magnitude[i][j] = magnitude[i-1][j-1] + 1;
        direction[i][j] = '@'; //using @ to represent diagonal(up/left) arrow.
      }

      else if(magnitude[i-1][j] >= magnitude[i][j-1]){
        magnitude[i][j] = magnitude[i-1][j];
        direction[i][j] = '^'; //using # to represent the up arrow.
      }

      else{
        magnitude[i][j] = magnitude[i][j-1];
        direction[i][j] = '<'; //using ! to represent the left arrow.
      }
    }
  }

  printLCS(direction, seq1, seq1.length(), seq2.length());

  return magnitude[m][n];
}



int main(){

  string sequence1;
  string sequence2;
```

```
  cout << "Enter sequence 1: ";
  cin >> sequence1;

  cout << "Enter sequence 2: ";
  cin >> sequence2;
  // sequence1 = "abcdeeffdd";
  // sequence2 = "ded";

  string space = " ";
  sequence1.insert(0, space);
  sequence2.insert(0, space);

  int sequenceLength = longest_common_subsequence(sequence1, sequence2);
  cout << endl << sequenceLength - 1 << endl;

}
```

1.2 Output

```
Enter sequence 1: aafeghedi
Enter sequence 2: eefdj
eed
```