**EXPERIMENT N0-10**

**AIM:** **Implementation of Longest Common Subsequence Algorithm.**

**CODE:**

#include<stdio.h>

#include<string.h>

int max(int a, int b);

void findLCS(char \*X, char \*Y, int XLen, int YLen);

int max(int a, int b) {

return (a > b)? a : b;

}

void findLCS(char \*X, char \*Y, int XLen, int YLen) {

int L[XLen + 1][YLen + 1];

int r, c, i;

for(r = 0; r <= XLen; r++) {

for(c = 0; c <= YLen; c++) {

if(r == 0 || c == 0) {

L[r][c] = 0;

} else if(X[r - 1] == Y[c - 1]) {

L[r][c] = L[r - 1][c - 1] + 1;

} else {

L[r][c] = max(L[r - 1][c], L[r][c - 1]);

}

}

}

r = XLen;

c = YLen;

i = L[r][c];

char LCS[i+1];

LCS[i] = '\0';

while(r > 0 && c > 0) {

if(X[r - 1] == Y[c - 1]) {

LCS[i - 1] = X[r - 1];

i--;

r--;

c--;

} else if(L[r - 1][c] > L[r][c - 1]) {

r--;

} else {

c--;

}

}

printf("Length of the LCS: %d\n", L[XLen][YLen]);

printf("LCS: %s\n", LCS);

}

int main(void) {

char A[20],B[20];

printf("Enter String A:--> ");

scanf("%s",A);

printf("Enter String B:--> ");

scanf("%s",B);

int XLen = strlen(A);

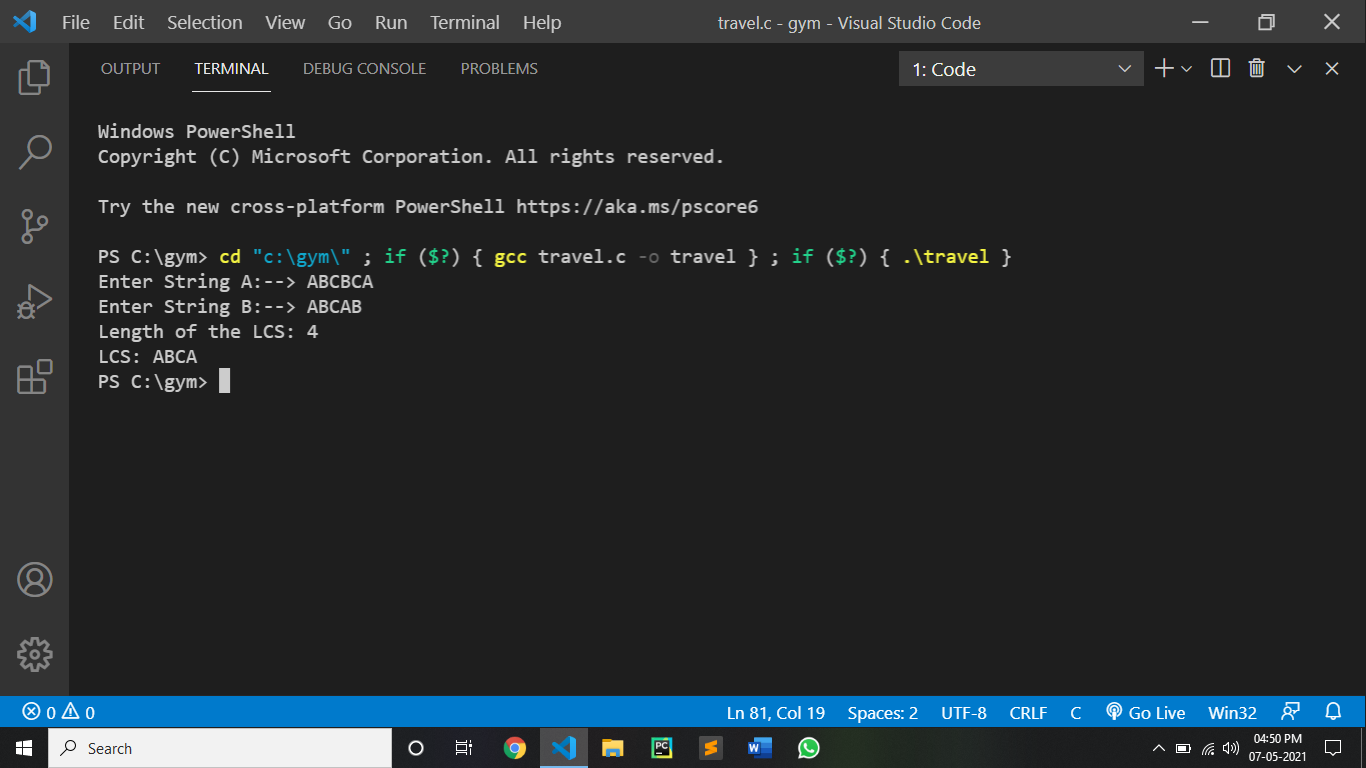
int YLen = strlen(B);

findLCS(A,B, XLen, YLen);

return 0;

}

**OUTPUT:**



CONCLUSION:

By performing the above algorithm we can conclude that:

•Checking membership of one subsequance of P[1…m] into Q[1…n] takes O(n) time. subsequance are possible for string P of length m.

•So worst case running time of brute force approach would be O(n.)

•In dynamic programming, the only table of size m\*n is filled up using two nested for loops.

•So running time of dynamic programming approach would take O(mn)

•Same thing would consider for space complexity.