**Batch: B1 Roll No.: 1611077**

**Experiment No. \_\_\_9\_\_\_**

**Grade: AA / AB / BB / BC / CC / CD /DD**

**Signature of the Staff In-charge with date**

|  |
| --- |
| **Title: Implementation Of String Matching Algorithm** |

**Objective:** To compute longest common subsequence for the given two strings.

**CO to be achieved:**

|  |  |
| --- | --- |
| Sr. No | Objective |
| CO 1 | Compare and demonstrate the efficiency of algorithms using asymptotic complexity notations. |
| CO 2 | Analyze and solve problems for divide and conquer strategy, greedy method, dynamic programming approach and backtracking and branch & bound policies. |
| CO 3 | Analyze and solve problems for   different string matching algorithms. |

**Books/ Journals/ Websites referred:**

1. **Ellis horowitz, Sarataj Sahni, S.Rajsekaran,” Fundamentals of computer algorithm”, University Press**
2. **T.H.Cormen ,C.E.Leiserson,R.L.Rivest and C.Stein,” Introduction to algortihtms”,2nd Edition ,MIT press/McGraw Hill,2001**
3. [**http://en.wikipedia.org/wiki/Longest\_common\_subsequence\_problem**](http://en.wikipedia.org/wiki/Longest_common_subsequence_problem)
4. [**http://www.columbia.edu/~cs2035/courses/csor4231.F11/lcs.pdf**](http://www.columbia.edu/~cs2035/courses/csor4231.F11/lcs.pdf)
5. **http://www-igm.univ-mlv.fr/~lecroq/seqcomp/node4.html**

**Pre Lab/ Prior Concepts:**

Data structures, Concepts of algorithm analysis

**Historical Profile:**

Given 2 sequences, *X* = *x*1 *, ..., xm*  and *Y* = *y*1 *, ... , yn* , find a subsequence common to both whose length is longest. A subsequence doesn’t have to be consecutive, but it has to be in order.

**New Concepts to be learned:**

String matching algorithm, Dynamic programming approach for LCS, Applications of LCS

**Recursive Formulation:**

Define *c*[*i, j* ] = length of LCS of *Xi* and *Y j* .

Final answer will be computed with *c*[*m, n*].

c[i, j]= 0 if i=0 or j=0.

c[i, j]= c[i − 1, j − 1] + 1 if i,j>0 and xi=yj

c[i, j]= max(c[i − 1, j ], c[i, j − 1]) if i, j > 0 and xi <> y j

**Algorithm: Longest Common Subsequence**

**Compute length of optimal solution-**

**LCS-LENGTH** *( X , Y, m, n)*

**for** *i* ← 1 **to** *m*

**do** *c*[*i,* 0] ← 0

**for** *j* ← 0 **to** *n*

**do** *c*[0*, j* ] ← 0

**for** *i* ← 1 **to** *m*

**do for** *j* ← 1 **to** *n*

**do if** *xi* = *y j*

**then** *c*[*i, j* ] ← *c*[*i* − 1*, j* − 1] + 1

*b*[*i, j* ] ← “≈”

**else if** *c*[*i* − 1*, j* ] ≥ *c*[*i, j* − 1]

**then** *c*[*i, j* ] ← *c*[*i* − 1*, j* ]

*b*[*i, j* ] ← “↑”

**else** *c*[*i, j* ] ← *c*[*i, j* − 1]

*b*[*i, j* ] ← “←”

**return** *c* and *b*

**Print the solution-**

**PRINT-LCS*(b, X , i, j )***

**if** *i* = 0 or *j* = 0

**then return**

**if** *b*[*i, j* ] = “≈”

**then** PRINT-LCS*(b, X , i* − 1*, j* − 1*)*

print *xi*

**elseif** *b*[*i, j* ] = “↑”

**then** PRINT-LCS*(b, X , i* − 1*, j )*

**else** PRINT-LCS*(b, X , i, j* − 1*)*

• Initial call is PRINT-LCS*(b, X , m, n)*.

• *b*[*i, j* ] points to table entry whose sub problem we used in solving LCS of *Xi*

and *Y j* .

• When *b*[*i, j* ] = ≈, we have extended LCS by one character. So longest com- mon subsequence = entries with ≈ in them.

**Analysis of Algorithm:**

* Time Complexity of the above implemented algorithm is O(mn); where m is length of Sting1 and n is length of String2.
* This is much better than the worst case time complexity of Naive Recursive implementation.

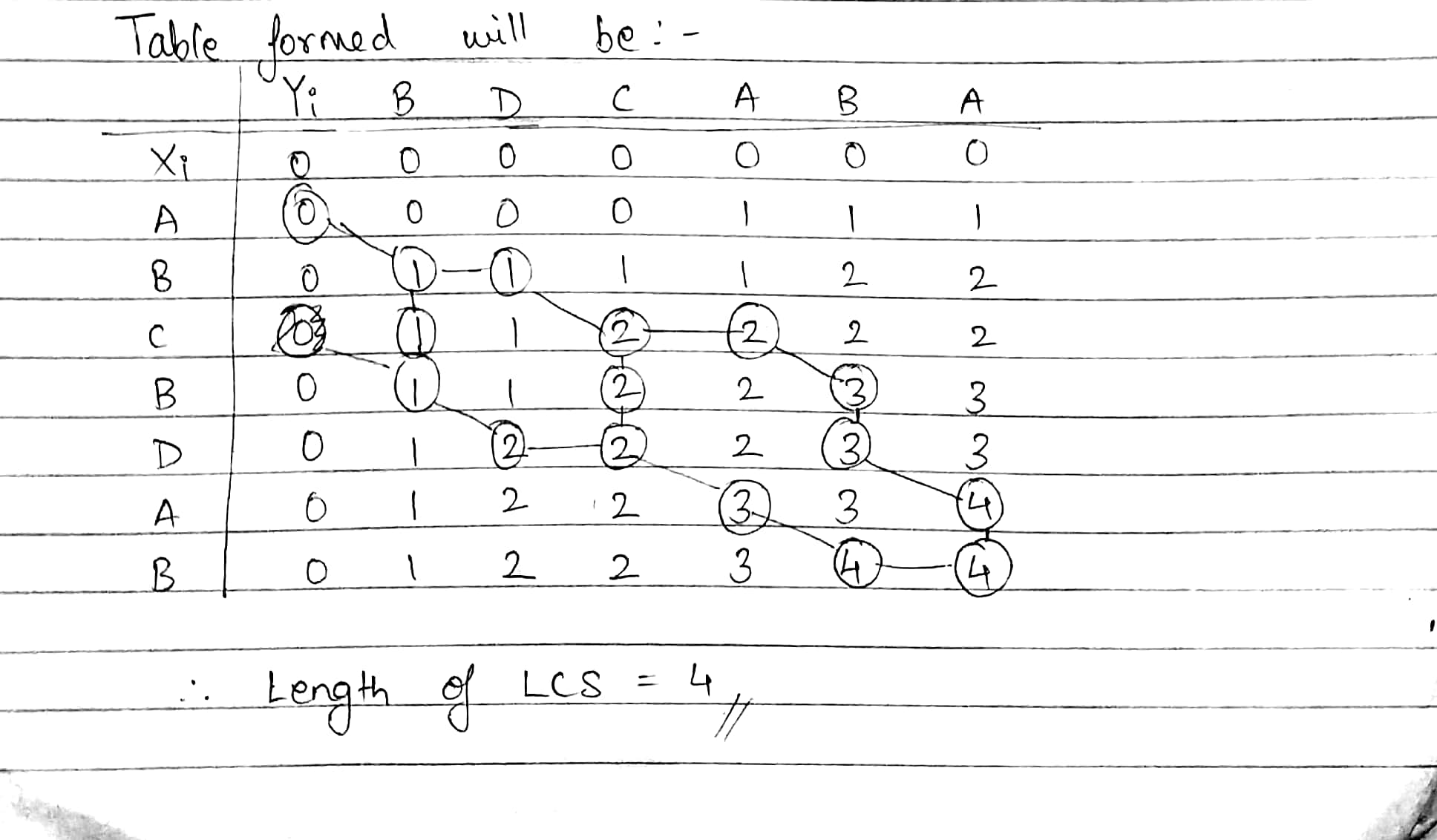
**Example:**

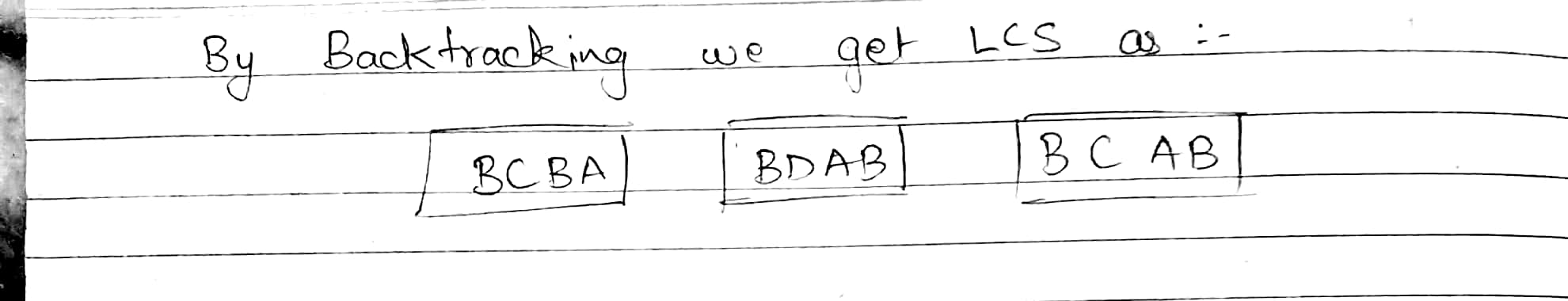
**Let the two strings be**

**X=**ABCBDAB

**Y=**BDCABA

**LCS computation**

****

****

**IMPLEMENTATION:**

import java.util.\*;

class Lcs

{

static int q=0;

public static void main(String[] args)

{

Scanner sc=new Scanner(System.in);

System.out.println("Enter the string: ");

String sa=sc.nextLine();

System.out.println("Enter test string: ");

String sb=sc.next();

String s1=sa.toLowerCase();

String s2=sb.toLowerCase();

int m=s1.length();

int n=s2.length();

char a[]=s1.toCharArray();

char b[]=s2.toCharArray();

int x[][]=new int[m+1][n+1];

int i,j;

for(i=0;i<n;i++)

{

x[0][i]=0;

}

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

{

x[i][0]=0;

}

for(i=1;i<=m;i++){

for(j=1;j<=n;j++){

if(a[i-1]==b[j-1])

{

x[i][j]=x[i-1][j-1]+1;

}

else

{

x[i][j]=Math.max(x[i-1][j],x[i][j-1]);

}

}

}

System.out.print("\n\n");

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

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

System.out.print(x[i][j]+" ");

}

System.out.print("\n");

}

char lcs[][]=new char[100][Math.min(m,n)];

System.out.print("\n\nLCS is: \n");

i=m;

j=n;

int k=0;

int l=0;

again(x,a,i,j,k,l,lcs);

i=0;

while(lcs[0][i]!='\u0000')

{

i++;

}

while(i>=1)

{

System.out.print(lcs[0][i-1]);

i--;

}

for(i=1;i<=q;i++)

{

System.out.print("\n");

j=0;

while(lcs[i][j]=='\u0000')

{

j++;

}

System.out.print(lcs[i][j]);

while(j>=0)

{

if(j!=0)

{

if(lcs[i][j-1]=='\u0000')

{

j--;

i--;

}

else

{

j--;

}

System.out.print(lcs[i][j]);

}

else

{

System.exit(1);

}

}

}

System.out.print("\n\nProgram By Mihir Gandhi B1 1611077 \n");

}

public static void again(int x[][], char a[], int i, int j, int k,int l, char lcs[][])

{

if(i>0&&j>0)

{

if(x[i][j]==x[i-1][j]&&x[i][j-1]==x[i][j])

{

again(x,a,i-1,j,k,l,lcs);

l++;

q++;

again(x,a,i,j-1,k,l,lcs);

}

else

{

if(x[i][j]==x[i-1][j])

{

again(x,a,i-1,j,k,l,lcs);

}

else if(x[i][j]==x[i][j-1])

{

again(x,a,i,j-1,k,l,lcs);

}

else

{

lcs[l][k]=a[i-1];

k++;

again(x,a,i-1,j-1,k,l,lcs);

}

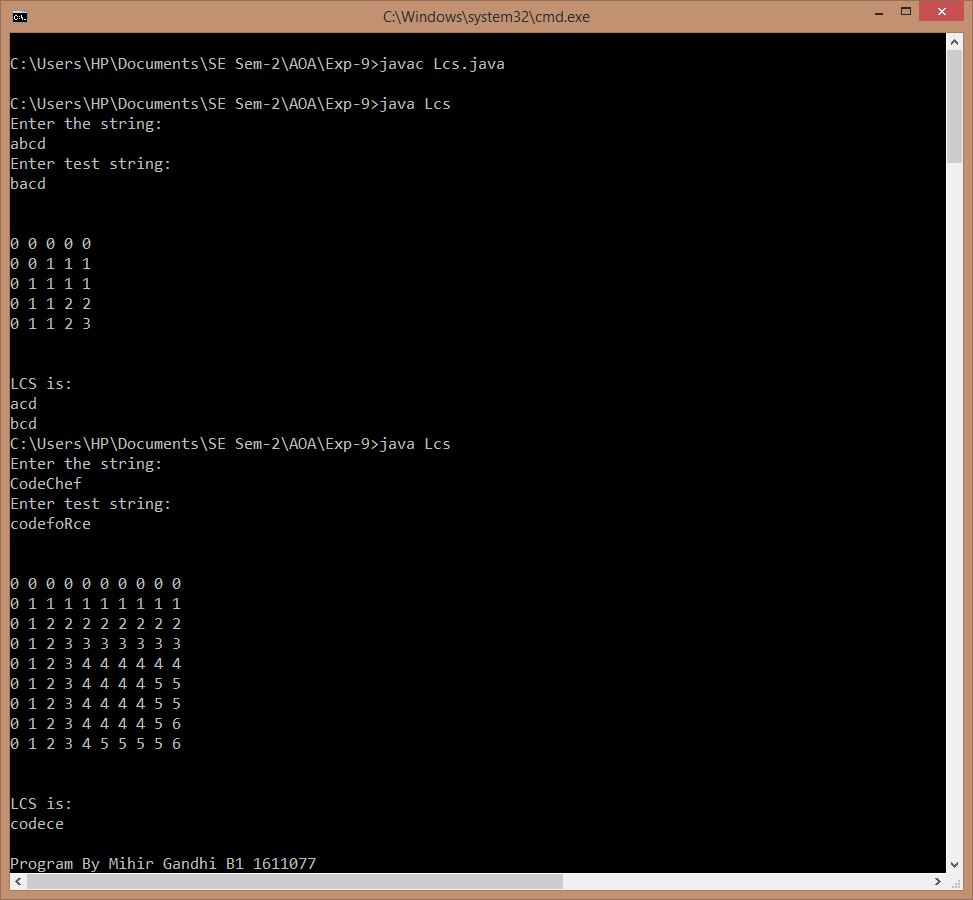
}

}

}

}

**OUTPUT**

****

**CONCLUSION:**

Thus, a program to compute longest common subsequence for the given two strings has been implemented successfully. The actual outcome matched with the expected outcome.