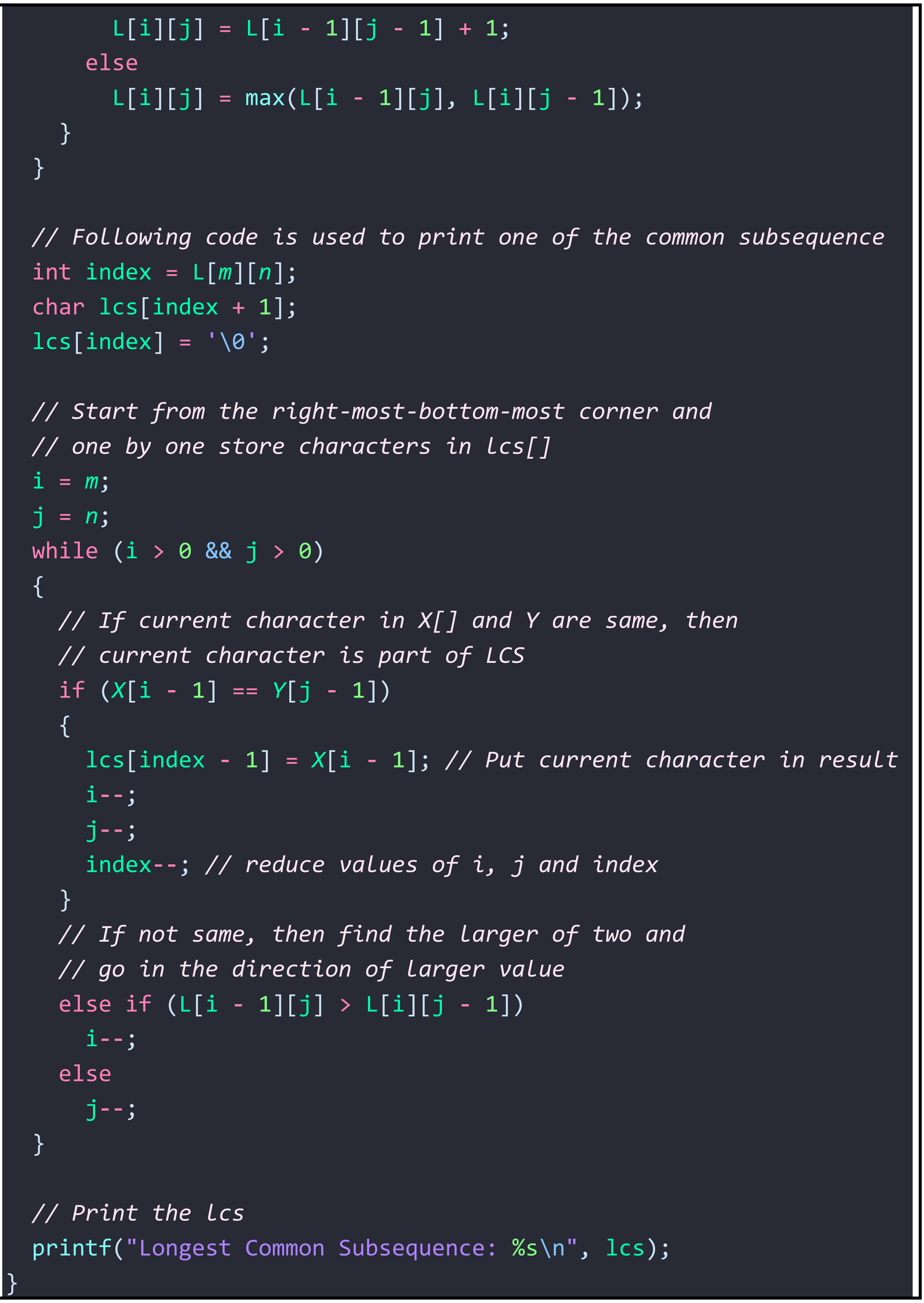
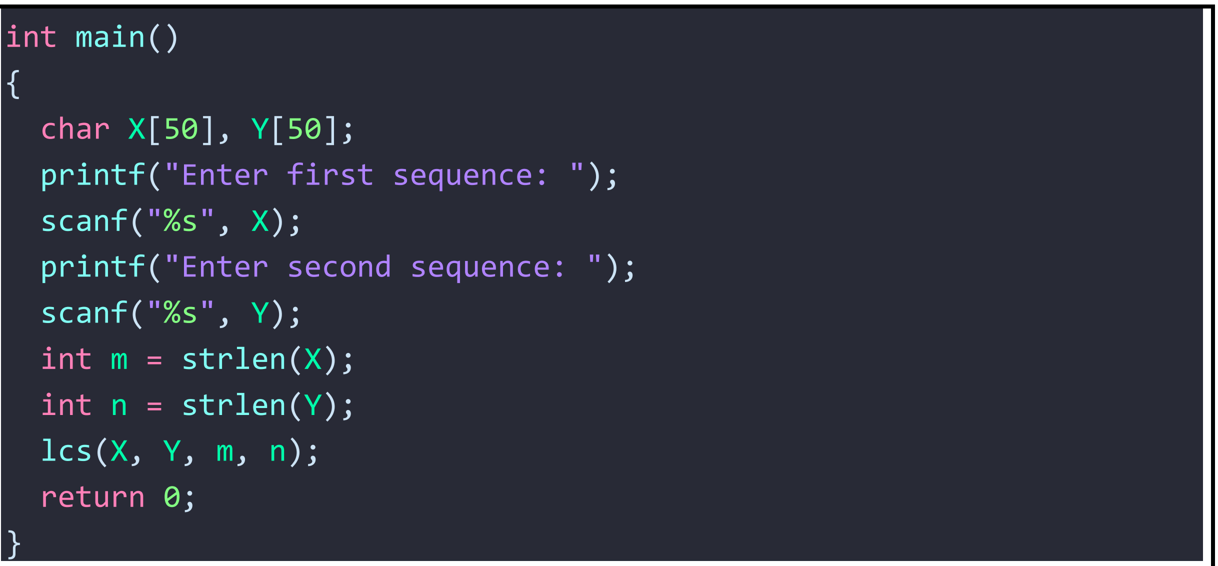
BHARATIYA VIDYA BHAVAN'S
SARDAR PATEL INSTITUTE OF TECHNOLOGY
(Empowered Autonomous Institute Affiliated to Mumbai University)
Department Of Computer Engineering

Name	Manish Shashikant Jadhav
UID	2023301005
Subject	Design and Analysis of Algorithms (DAA)
Experiment No.	4
Aim	To implement Dynamic Algorithms. a) Assembly Line Scheduling. b) Longest Common Subsequence.
Code:	**Longest Common Subsequence (LCS):**

```
#include <stdio.h>
#include <string.h>

// Function to find the maximum of two integers
int max(int a, int b)
{
  return (a > b) ? a : b;
}

// Function to find the length of longest common subsequence
// and print one of the common subsequences
void lcs(char *X, char *Y, int m, int n)
{
  int L[m + 1][n + 1];
  int i, j;

  // Building the L[m+1][n+1] in bottom-up fashion
  for (i = 0; i <= m; i++)
  {
    for (j = 0; j <= n; j++)
    {
      if (i == 0 || j == 0)
        L[i][j] = 0;
      else if (X[i - 1] == Y[j - 1])
```

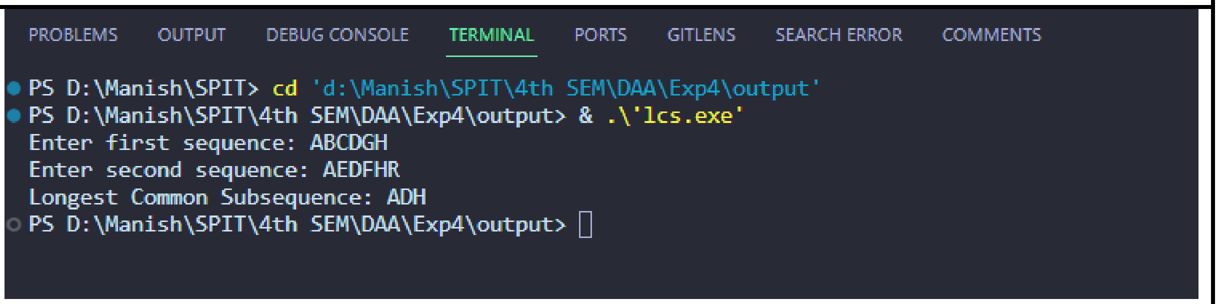


```
        L[i][j] = L[i - 1][j - 1] + 1;
      else
        L[i][j] = max(L[i - 1][j], L[i][j - 1]);
    }
  }

  // Following code is used to print one of the common subsequence
  int index = L[m][n];
  char lcs[index + 1];
  lcs[index] = '\0';

  // Start from the right-most-bottom-most corner and
  // one by one store characters in lcs[]
  i = m;
  j = n;
  while (i > 0 && j > 0)
  {
    // If current character in X[] and Y are same, then
    // current character is part of LCS
    if (X[i - 1] == Y[j - 1])
    {
      lcs[index - 1] = X[i - 1]; // Put current character in result
      i--;
      j--;
      index--; // reduce values of i, j and index
    }
    // If not same, then find the larger of two and
    // go in the direction of larger value
    else if (L[i - 1][j] > L[i][j - 1])
      i--;
    else
      j--;
  }

  // Print the lcs
  printf("Longest Common Subsequence: %s\n", lcs);
}
```



```c
int main()
{
  char X[50], Y[50];
  printf("Enter first sequence: ");
  scanf("%s", X);
  printf("Enter second sequence: ");
  scanf("%s", Y);
  int m = strlen(X);
  int n = strlen(Y);
  lcs(X, Y, m, n);
  return 0;
}
```

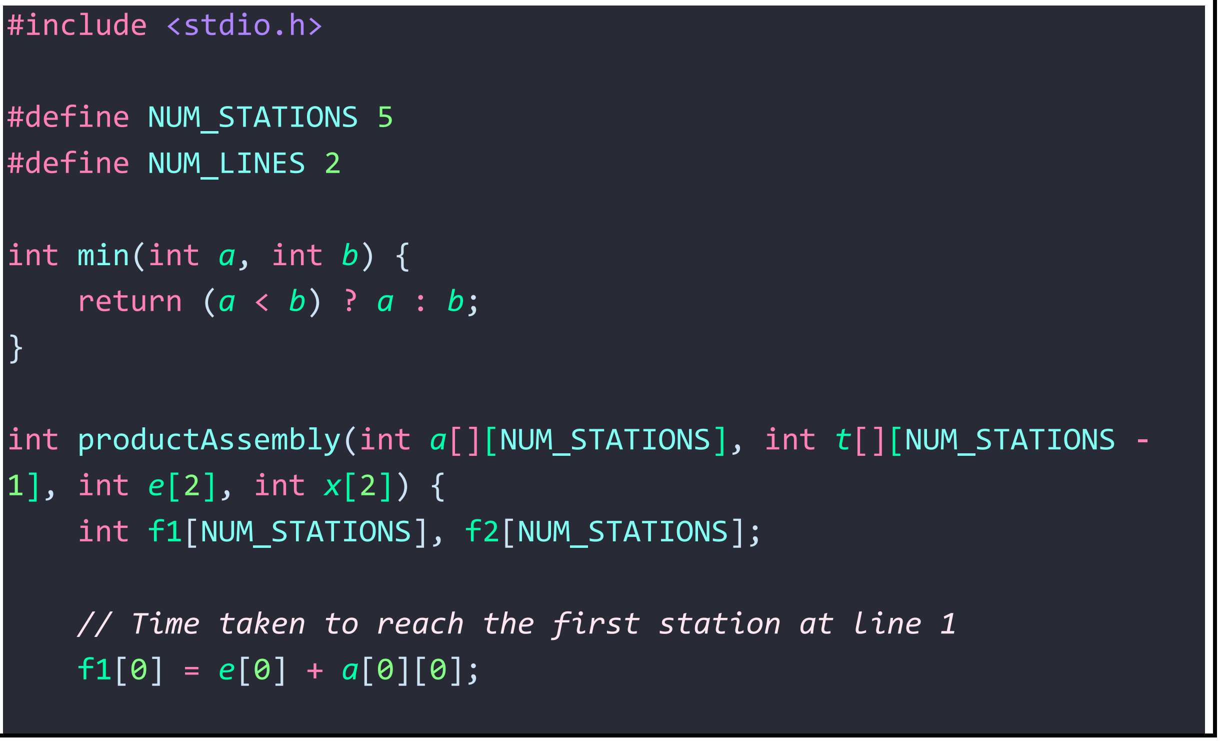
Output



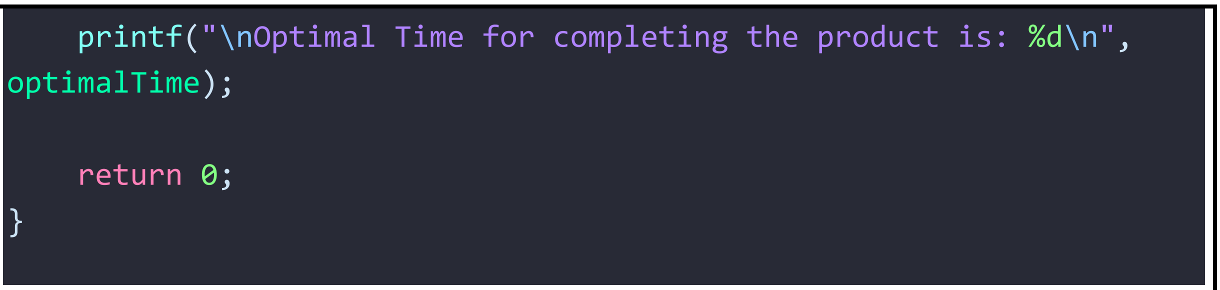
```
PS D:\Manish\SPIT> cd 'd:\Manish\SPIT\4th SEM\DAA\Exp4\output'
PS D:\Manish\SPIT\4th SEM\DAA\Exp4\output> & .\'lcs.exe'
Enter first sequence: ABCDGH
Enter second sequence: AEDFHR
Longest Common Subsequence: ADH
PS D:\Manish\SPIT\4th SEM\DAA\Exp4\output>
```

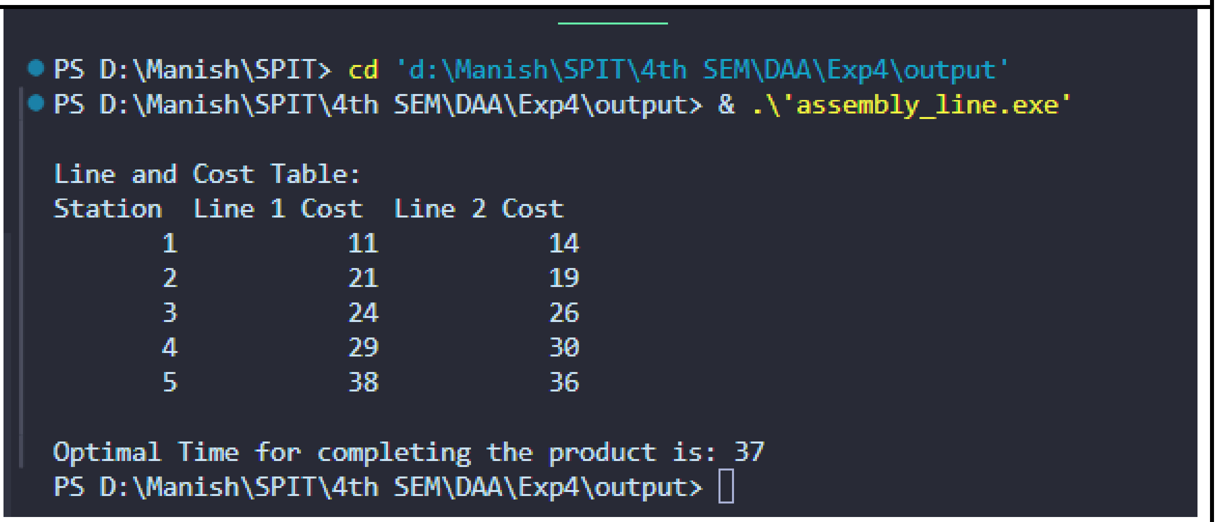
Code

Assembly Line Scheduling:



```c
#include <stdio.h>

#define NUM_STATIONS 5
#define NUM_LINES 2

int min(int a, int b) {
    return (a < b) ? a : b;
}

int productAssembly(int a[][NUM_STATIONS], int t[][NUM_STATIONS - 1], int e[2], int x[2]) {
    int f1[NUM_STATIONS], f2[NUM_STATIONS];

    // Time taken to reach the first station at line 1
    f1[0] = e[0] + a[0][0];
    // Time taken to reach the first station at line 2
    f2[0] = e[1] + a[1][0];

    // Fill tables f1[] and f2[] using the given recursive relations
    for (int j = 1; j < NUM_STATIONS; j++) {
        f1[j] = min(f1[j - 1] + a[0][j], f2[j - 1] + t[1][j - 1] + a[0][j]);
        f2[j] = min(f2[j - 1] + a[1][j], f1[j - 1] + t[0][j - 1] + a[1][j]);
    }

    // Display the table of line and cost for each line
    printf("\nLine and Cost Table:\n");
    printf("Station  Line 1 Cost  Line 2 Cost\n");

    for (int i = 0; i < NUM_STATIONS; i++) {
        printf("%8d %12d %12d\n", i + 1, f1[i], f2[i]);
    }

    // Consider exit times and return minimum
    return min(f1[NUM_STATIONS - 1] + x[0], f2[NUM_STATIONS - 1] + x[1]);
}

int main() {
    int a[NUM_LINES][NUM_STATIONS] = {{8, 10, 4, 5, 9}, {9, 6, 7, 5, 6}};
    int t[NUM_LINES][NUM_STATIONS - 1] = {{2, 3, 1, 3}, {2, 1, 2, 2}};
    int e[NUM_LINES] = {3, 5};
    int x[NUM_LINES] = {2, 1};

    // Calculate and display the optimal time for completing the product
    int optimalTime = productAssembly(a, t, e, x);
```

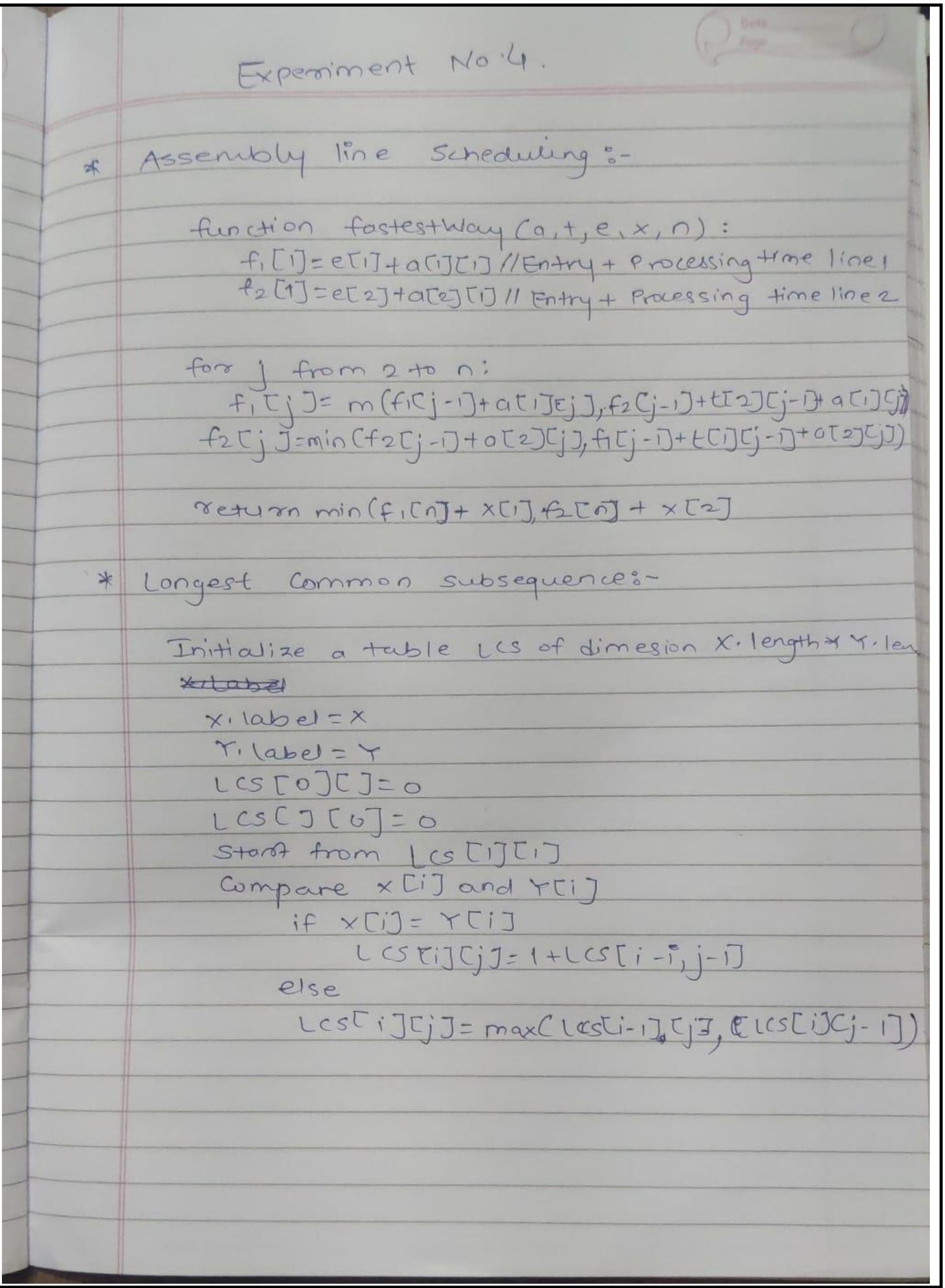
<table>
<tr>
<td></td>
<td>
<pre>
    printf("\nOptimal Time for completing the product is: %d\n",
optimalTime);

    return 0;
}
</pre></td>
</tr>
<tr>
<td><b>Output</b></td>
<td>
<pre>
● PS D:\Manish\SPIT> cd 'd:\Manish\SPIT\4th SEM\DAA\Exp4\output'
● PS D:\Manish\SPIT\4th SEM\DAA\Exp4\output> & .\'assembly_line.exe'

  Line and Cost Table:
  Station Line 1 Cost Line 2 Cost
         1   11   14
         2   21   19
         3   24   26
         4   29   30
         5   38   36

  Optimal Time for completing the product is: 37
  PS D:\Manish\SPIT\4th SEM\DAA\Exp4\output>
</pre></td>
</tr>
</table>

Pseudo Code



Experiment No.4.

* Assembly line Scheduling :-

```
function fastestWay (a, t, e, x, n) :
    f1 [1] = e[1] + a[1][1] //Entry + Processing time line1
    f2 [1] = e[2] + a[2][1] // Entry + Processing time line2

for j from 2 to n:
    f1 [j] = m (f1[j-1] + a[1][j], f2[j-1] + t[2][j-1] + a[1][j])
   f2 [j] = min (f2[j-1] + a[2][j], f1[j-1] + t[1][j-1] + a[2][j])

return min (f1[n] + x[1], f2[n] + x[2]
```

* Longest Common subsequence:-

```
Initialize a table LCS of dimesion X.length * Y.len
~~X.label~~
  X.label = X
  Y.label = Y
  LCS [0][] = 0
  LCS[] [0] = 0
  Start from LCS [1][1]
  Compare x[i] and Y[i]
        if x[i] = Y[i]
             LCS [i][j] = 1 + LCS[i-i, j-1]
       else
        LCS[i][j] = max(LCS[i-1], [j], LCS[i][j-1])
```

Conclusion	Hence, by completing this experiment I came to know about implementation of Dynamic Algorithms. a) Longest Common Subsequence. b) Assembly Line Scheduling.