WEEK-11

Aim: Apply dynamic programming methodology to implement 0/1 knapsack problem.

Code:

```
#include <stdio.h>

#include <string.h>

int i, j, m, n, LCS_table[20][20];

char S1[20] = "ACADBCBB", S2[20] = "CBDABB", b[20][20];

void lcs() {

  m = strlen(S1);

  n = strlen(S2);

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

    LCS_table[i][0] = 0;

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

    LCS_table[0][i] = 0;

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

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

      if (S1[i - 1] == S2[j - 1]) {

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

      } else if (LCS_table[i - 1][j] >= LCS_table[i][j - 1]) {

        LCS_table[i][j] = LCS_table[i - 1][j];

      } else {

        LCS_table[i][j] = LCS_table[i][j - 1];
```

```
 }}
   int index = LCS_table[m][n];
   char lcs[index + 1];
   lcs[index] = '\0';
   int i = m, j = n;
   while (i > 0 && j > 0) {
     if (S1[i - 1] == S2[j - 1]) {
       lcs[index - 1] = S1[i - 1];
       i--;
       j--;
       index--;}
     else if (LCS_table[i - 1][j] > LCS_table[i][j - 1])
       i--;
     else
       j--;
   }
   printf("S1 : %s \nS2 : %s \n", S1, S2);
   printf("LCS: %s", lcs);
 }
 int main() {
   lcs();
   printf("\n");
}
```

Output:

S1 : ACADBCBB

S2 : CBDABB

LCS: CBBB

WEEK-12

Aim: Solve the longest common subsequence problem using dynamic programming.

Code:

```
#include<stdio.h>

int max(int a, int b) {

    return (a > b)? a : b;

}

int knapSack(int W, int wt[], int val[], int n)

{

   int i, w;

   int K[n+1][W+1];

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

   {

       for (w = 0; w <= W; w++)

       {

           if (i==0 || w==0)

               K[i][w] = 0;

           else if (wt[i-1] <= w)

                 K[i][w] = max(val[i-1] + K[i-1][w-wt[i-1]],
K[i-1][w]);

           else

                 K[i][w] = K[i-1][w];
```

```
    }}
    return K[n][W];
 }
 int main()
 {
     int i, n, val[20], wt[20], W;
     printf("Enter number of items:");
     scanf("%d", &n);
     printf("Enter value and weight of items:\n");
     for(i = 0;i < n; ++i){
      scanf("%d%d", &val[i], &wt[i]);
     }
     printf("Enter capacity of knapsack:");
     scanf("%d", &W);
     printf("maximum profit earned:%d\n", knapSack(W, wt, val,
 n));
     return 0;
}
```

Output:

```
Enter number of items:3
Enter value and weight of items:
100 20
50 20
150 30
Enter capacity of knapsack:150
maximum profit earned:300
```