1. LCS + DP

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

int max(int x, int y);

// Function to find length of LCS for S1[0..m-1], S2[0..n-1]
int lcs(const char *S1, const char *S2) {
    int m = strlen(S1);
    int n = strlen(S2);

    // Initializing a matrix of size (m+1)*(n+1)
    int dp[m + 1][n + 1];

    // Building dp[m+1][n+1] in bottom-up fashion
    for (int i = 0; i <= m; i++) {
        for (int j = 0; j <= n; j++) {

            if (i == 0 || j == 0)
                dp[i][j] = 0;

            else if (S1[i - 1] == S2[j - 1])
                dp[i][j] = dp[i - 1][j - 1] + 1;

            else
                dp[i][j] = max(dp[i - 1][j], dp[i][j - 1]);
        }
    }

    return dp[m][n];
}

int max(int x, int y) {
    return (x > y) ? x : y;
}

int main() {
    const char *S1 = "AGGTAB";
    const char *S2 = "GXTXAYB";
    printf("Length of LCS is %d\n", lcs(S1, S2));

    return 0;
}
```

2. 0/1 knapsack

```
#include <stdio.h>

// Online C compiler to run C program online
#include <stdio.h>

int max(int a, int b) { return (a > b) ? a : b; }


int knapSack(int C, int n, int weight[], int value[])
{

    int i, k;
    int dp[n + 1][C + 1];


    // Build table dp[][] in bottom up manner
    for (i = 0; i <= n; i++) {
        for (k = 0; k <= C; k++) {
            if (i == 0 || k == 0)
                dp[i][k] = 0;
            else if (weight[i-1] <= k){

                dp[i][k] = max(value[i-1]
                              + dp[i - 1][k - weight[i-1]],
                          dp[i - 1][k]);
            }
            else
                dp[i][k] = dp[i - 1][k];
        }
    }

    return dp[n][C];
}

int main()
{
    int value[] = { 15,10, 9, 5};
    int weight[] = { 1, 5, 3, 4 };
    int C = 8;
    int n = sizeof(value) / sizeof(value[0]);
    printf("%d", knapSack(C, n, weight, value));
    return 0;
}
```