```
// 0-1 knapsack problem
#include <iostream>
using namespace std;
#define MAX_ITEMS 100
#define MAX_WEIGHT 100
int B[MAX_ITEMS + 1][MAX_WEIGHT + 1];
void knapsack(int n, int W, int weights[], int profits[]) {
    for (int w = 0 ; w <= W; w++) {
        B[0][w] = 0;}
    for (int i = 1 ; i <= n; i++) {
        B[i][0] = 0;}
    for(int i = 1 ; i <= n; i++) {
        for (int w = 1; w <= W; w++) {
            if (weights[i - 1] <= w) {
                if (profits[i - 1] + B[i - 1][w - weights[i - 1]] > B[i - 1][w]) {
                 B[i][w] = profits[i - 1] + B[i - 1][w - weights[i - 1]];
               }else {
                B[i][w] = B[i - 1][w];
            }} else {
                B[i][w] = B[i - 1][w];}}}
cout << "Maximum Profit: " << B[n][W] << endl;
int i = n, k = W;
cout << "Selected items: ";
    while (i > 0 && k > 0) {
        if (B[i][k] != B[i - 1][k]) {
            cout << i << " ";
            k -= weights[i - 1];}
        i--;}
    cout << endl;}
int main() {
    int n, W;
    cout << "Ente r number of items: ";
    cin >> n;
    cout << "Enter knapsack capacity: ";
    cin >> W;
int weights[MAX_ITEMS], profits[MAX_ITEMS];
    cout << "Enter weights of items: ";
    for (int i = 0; i < n; i++) {
        cin >> weights[i];}
cout << "Enter profits of items: ";
    for (int i = 0; i < n; i++) {
        cin >> profits[i];}
knapsack(n, W, weights, profits);
return 0;}
```