Practical:9

- Implementation of a knapsack problem using dynamic programming.

Code:

```c
#include <stdio.h>
#include <stdlib.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 val[] = { 60, 100, 120 };
    int wt[] = { 10, 20, 30 };
    int W = 50;
    int n = sizeof(val) / sizeof(val[0]);
    printf("Max Value: %d", knapsack(W, wt, val, n));
    return 0;
}
```

Output: