

LAB 6

Implement 0/1 Knapsack problem using dynamic programming.

CODE:

```
#include <stdio.h> #include
<conio.h> void knapsack(); int
max(int, int); int i, j, n, m, p[10],
w[10], v[10][10]; void main()
{
    printf("\nEnter the no. of items:\n");
    scanf("%d", &n); printf("\nEnter the weight of
    the each item:\n"); for (i = 1; i <= n; i++)
    {
        scanf("%d", &w[i]);
    }
    printf("\nEnter the profit of each item:\n");
    for (i = 1; i <= n; i++)
    {
        scanf("%d", &p[i]);
    }
    printf("\nEnter the knapsack's capacity:\n");
    scanf("%d", &m); knapsack(); getch();
}
void knapsack()
{ int x[10];
    for (i = 0; i <= n; i++)
    {
        for (j = 0; j <= m; j++)
        {
            if (i == 0 || j == 0)
            { v[i][j] = 0;
              }
            else if (j - w[i] < 0)
            { v[i][j] = v[i - 1][j];
              }
            else
```

```

        {
            v[i][j] = max(v[i - 1][j], v[i - 1][j] - w[i] + p[i]);
        }
    }
}

printf("\nThe output is:\n");
for (i = 0; i <= n; i++)

{
    for (j = 0; j <= m; j++)
    { printf("%d ", v[i][j]);
      }
    printf("\n\n");
}

printf("\nThe optimal solution is %d", v[n][m]);
printf("\nThe solution vector is:\n"); for (i = n;
i >= 1; i--)
{
    if (v[i][m] != v[i - 1][m])
    {
        x[i] = 1;
        m = m - w[i];
    }
    else
    {
        x[i] = 0;
    }
}
for (i = 1; i <= n; i++)
{ printf("%d\t", x[i]);
}
}

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

```

}

OUTPUT:

[illegible]