EXP 6

Implement 0/1 Knapsack problem using dynamic programming.

CODE:

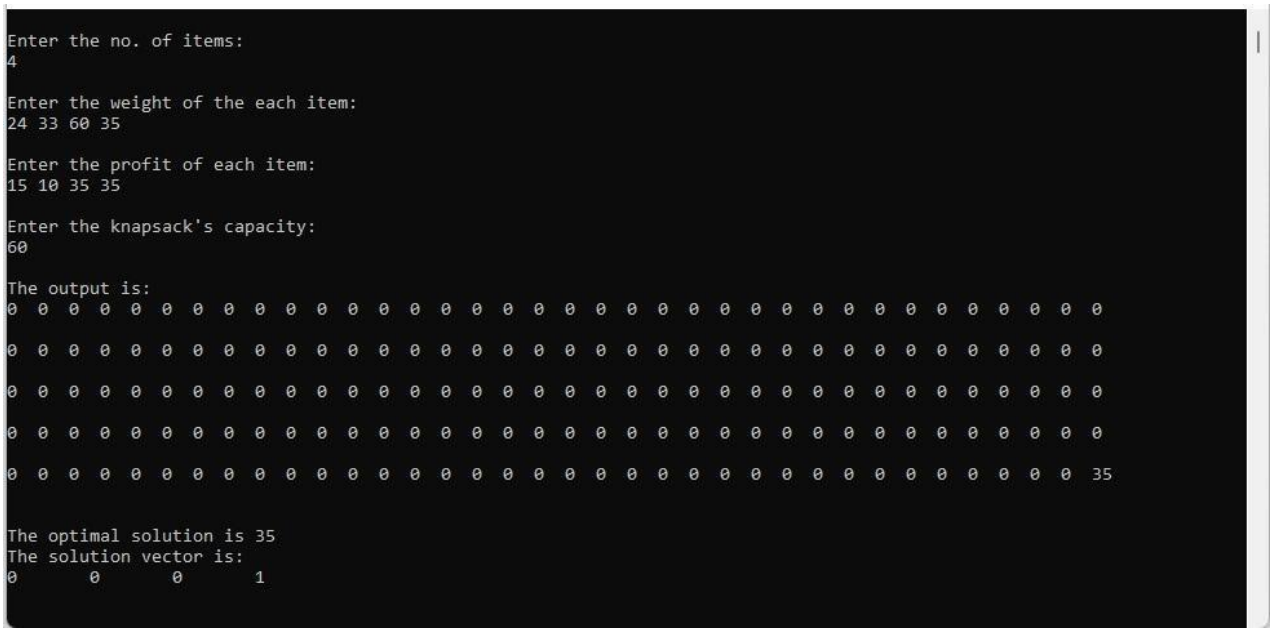
```
#include <stdio.h> #include
<conio.h> void knapsack(); int
max(int, int); int i, j, n, m, p[10], w[10],
v[10][10]; void main()
{
   printf("\nEnter the no. of items:\n");
   scanf("%d", &n); printf("\nEnter the weight of
   the each item:\n"); for (i = 1; i <= n; i++)
   {
      scanf("%d", &w[i]);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

```
        return y;
    }
}
```

OUTPUT:



EXP 5

Sort a given set of N integer elements using Quick Sort technique

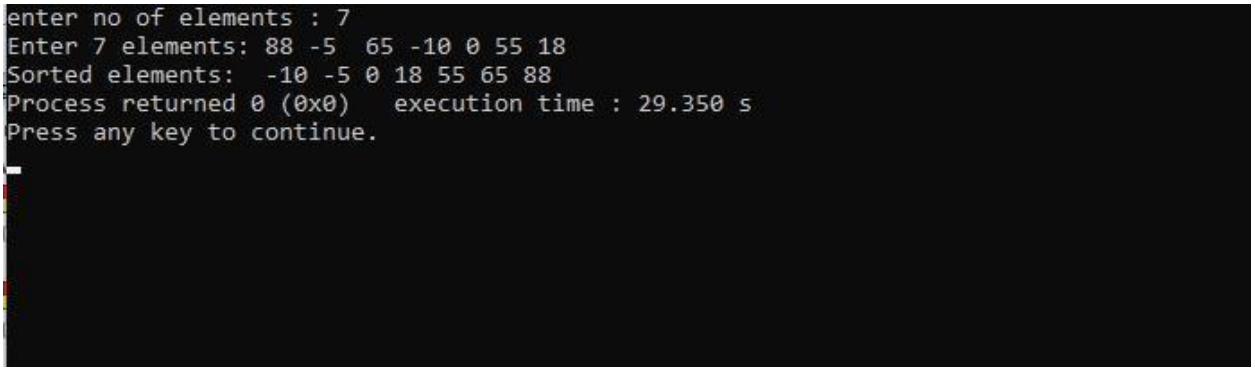
CODE:

```
#include<stdio.h>

void quicksort(int number[25],int first,int last)
```

```
{ int i, j, pivot, temp;
    if(first<last)
    { pivot=first;
        i=first; j=last;
        while(i<j)
        {
            while(number[i]<=number[pivot]&&i<last
            ) i++; while(number[j]>number[pivot]) j--;
            if(i<j)
            {
                temp=number[i]; number[i]=number[j];
                number[j]=temp;
            }
        }
        temp=number[pivot];
        number[pivot]=number[j];
        number[j]=temp; quicksort(number,first,j1);
        quicksort(number,j+1,last);
    }
}
int main()
{
    int i, count, number[25];
    printf("enter no of elements : ");
    scanf("%d",&count); printf("Enter
    %d elements: ", count);
    for(i=0;i<count;i++)
    scanf("%d",&number[i]);
    quicksort(number,0,count-1);
    printf("Sorted elements: ");
    for(i=0;i<count;i++) printf("
    %d",number[i]); return 0;
}
```

OUTPUT:


enter no of elements : 7
Enter 7 elements: 88 -5 65 -10 0 55 18
Sorted elements: -10 -5 0 18 55 65 88
Process returned 0 (0x0) execution time : 29.350 s
Press any key to continue.