PROGRAM 12

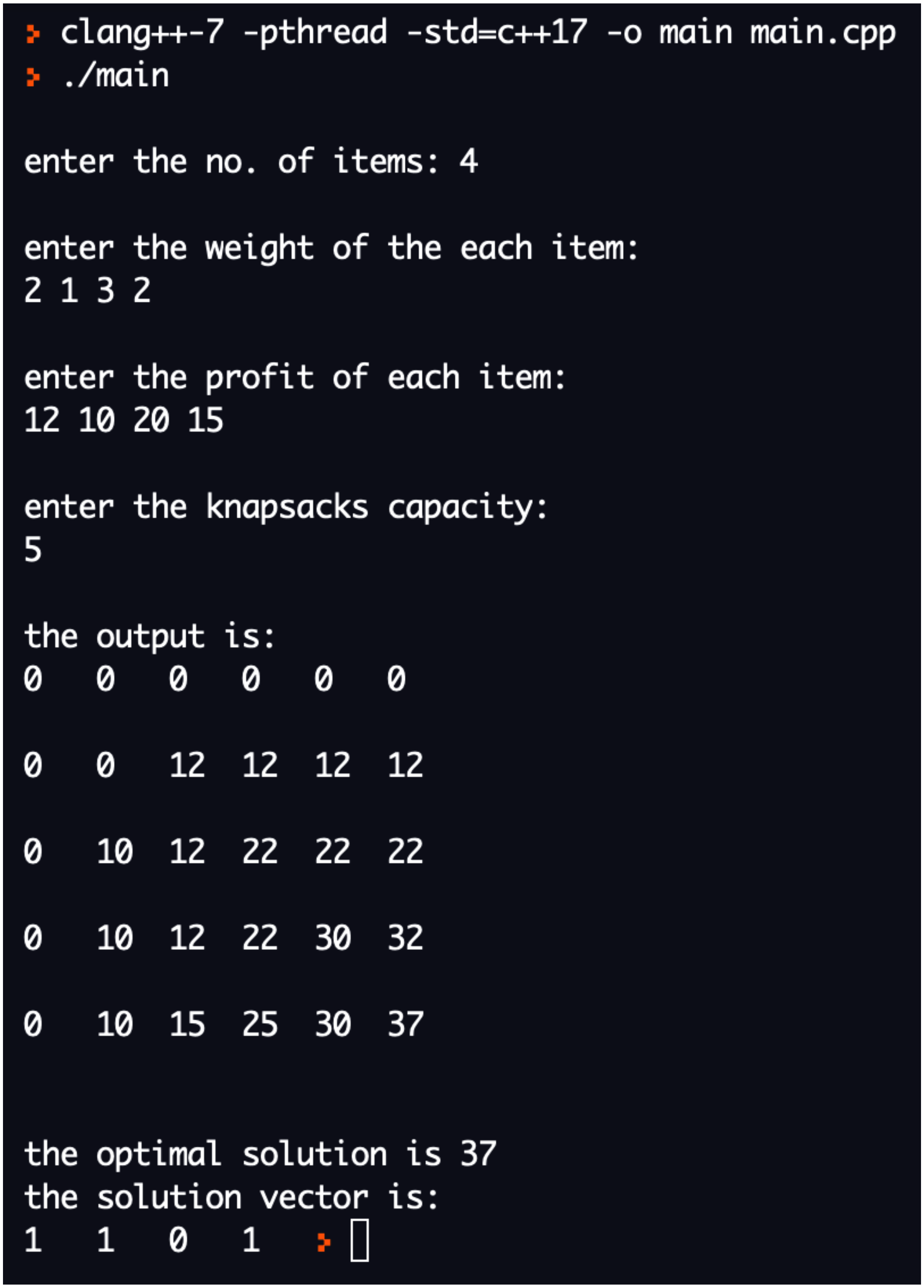
Implement 0/1 Knapsack problem using dynamic programming.

//Code

```
#include<stdio.h>

void knapsack();
int max(int,int);
int i,j,n,m,p[10],w[10],v[10][10];
int main()
{

printf("\nenter the no. of items:\t");
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 knapsacks capacity:\t");
scanf("%d",&m);
knapsack();

}
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\t",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



```
› clang++-7 -pthread -std=c++17 -o main main.cpp
› ./main

enter the no. of items: 4

enter the weight of the each item:
2 1 3 2

enter the profit of each item:
12 10 20 15

enter the knapsacks capacity:
5

the output is:
0   0   0   0   0   0

0   0   12  12  12  12

0   10  12  22  22  22

0   10  12  22  30  32

0   10  15  25  30  37


the optimal solution is 37
the solution vector is:
1   1   0   1   › ▯
```