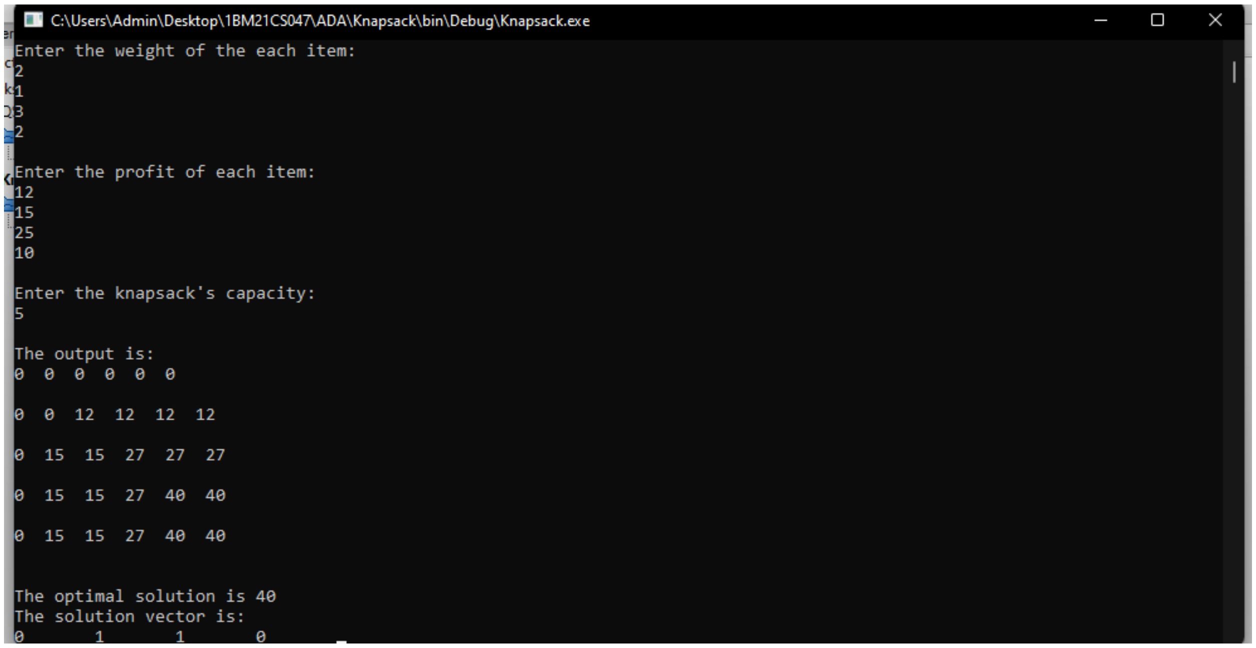
6.Implement 0/1 Knapsack problem using dynamic programming.

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:



```
C:\Users\Admin\Desktop\1BM21CS047\ADA\Knapsack\bin\Debug\Knapsack.exe
Enter the weight of the each item:
2
1
3
2

Enter the profit of each item:
12
15
25
10

Enter the knapsack's capacity:
5

The output is:
0  0  0  0  0  0

0  0  12  12  12  12

0  15  15  27  27  27

0  15  15  27  40  40

0  15  15  27  40  40


The optimal solution is 40
The solution vector is:
0       1       1       0
```

Observation:

Sort method

20-07-23

a) Implement 0/1 knapsack problem using dynamic Programming

```
#include <stdio.h>.
int ~~[illegible]~~ i, j, n, m, P[10], w[10], V[10][10];
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;
}
```

```
void main() {
  ~~int n[10];~~
  printf("\nEnter the no. of items: \n");
  scanf("%d", &n);
  print ("\n Enter the weight of each item: \n");
  for (i= 1; i<=n; i++)
       scanf("%d", &w[i]);
  printf("\n Enter profit of each item: \n");
    for (i= 1; i<=n; i++)
         scanf("%d", &p[i]);
  printf("\nEnter knapsack's capacity: \n");
  scanf("%d", &m);
  knapsack();
  getch();
}.
```

Output:

```
Enter the no.of items: 4
Enter the weight of each item: 2 1 3 2
Enter the profit of each item: 12 15 25 10
Enter the knapsack's capacity: 5
The output is :
  0  0  0  0  0  0
  0  0  12 12 12 12.
  0  15 15 27 27 27
  0  15 15 27 40 40
  0  15 15 27 40 40.

The Optimal Solution is 40
The solution vector is:
 0  1  1  0.
```