**6. Design and implement C/C++ Program to solve 0/1 Knapsack problem using Dynamic**

**Programming method.**

#include<stdio.h>

int w[10],p[10],n;

int max(int a,int b)

{

return a>b?a:b;

}

int knap(int i,int m)

{

if(i==n) return w[i]>m?0:p[i];

if(w[i]>m) return knap(i+1,m);

return max(knap(i+1,m),knap(i+1,m-w[i])+p[i]);

}

int main()

{

int m,i,max\_profit;

printf("\nEnter the no. of objects:");

scanf("%d",&n);

printf("\nEnter the knapsack capacity:");

scanf("%d",&m);

printf("\nEnter profit followed by weight:\n");

for(i=1; i<=n; i++)

scanf("%d %d",&p[i],&w[i]);

max\_profit=knap(1,m);

printf("\nMax profit=%d",max\_profit);

return 0;

}

**OUTPUT**

Enter the no. of objects:4

Enter the knapsack capacity:5

Enter profit followed by weight:

12 3

43 5

45 2

55 3

Max profit=100

**8. Design and implement C/C++ Program to find a subset of a given set S = {sl , s2,.....,sn} of n positive integers whose sum is equal to a given positive integer d.**

#include<stdio.h>

#define MAX 10

int s[MAX],x[MAX],d;

void sumofsub(int p,int k,int r)

{

int i;

x[k]=1;

if((p+s[k])==d)

{

for(i=1; i<=k; i++)

if(x[i]==1)

printf("%d ",s[i]);

printf("\n");

}

else if(p+s[k]+s[k+1]<=d)

sumofsub(p+s[k],k+1,r

-s[k]);

if((p+r

-s[k]>=d) && (p+s[k+1]<=d))

{

x[k]=0;

sumofsub(p,k+1,r

-s[k]);

}

}

int main()

{

int i,n,sum=0;

printf("\nEnter the n value:");

scanf("%d",&n);

printf("\nEnter the set in increasing order:");

for(i=1; i<=n; i++)

scanf("%d",&s[i]);

printf("\nEnter the max subset value:");

scanf("%d",&d);

for(i=1; i<=n; i++)

sum=sum+s[i];

if(sum<d || s[1]>d)

printf("\nNo subset possible");

else

sumofsub(0,1,sum);

return 0;

}

**OUTPUT**

Enter the n value:9

Enter the set in increasing order:1 2 3 4 5 6 7 8 9

Enter the max subset value:9

1 2 6

1 3 5

1 8

2 3 4

2 7

3 6

4 5

9