**NAME:ROHAN NYATI**

**SAP ID:500075940**

**ROLL NO. : R177219148**

**BATCH-5(AI&ML)**

**Algorithm for Intelligent System and Robotic Lab**

**Practical 5**

***Aim – To implement a solution for any NP Complete problem using recursion.***

**Problem Chosen : 0-1Knapsack Problem**

--- Implementation in Python3 ---

class item :

def \_\_init\_\_( self , weight , value ) :

self.value = value

self.weight = weight

def knapsack( items , capacity , n ) :

if( n==0 ) :

return 0 ;

elif( capacity < items[n-1].weight ):

return knapsack( items , capacity , n - 1 ) ;

else :

sack\_with\_curr\_item = knapsack( items , capacity - items[n - 1].weight , n - 1 ) + items[ n - 1 ].value

sack\_without\_curr\_item = knapsack( items , capacity , n - 1 )

return max( sack\_with\_curr\_item , sack\_without\_curr\_item )

if \_\_name\_\_ == "\_\_main\_\_" :

items = []

capacity = int(input("Enter the capacity : "))

n = int(input("Enter the number of item : "))

for i in range( n ) :

items.append( item( \*( map(int , input(f"Enter the weight and value of each item {i+1} : ").split() ) ) ) )

print("Max sum --> " , knapsack( items , capacity , n ))

--- OUTPUT ---

