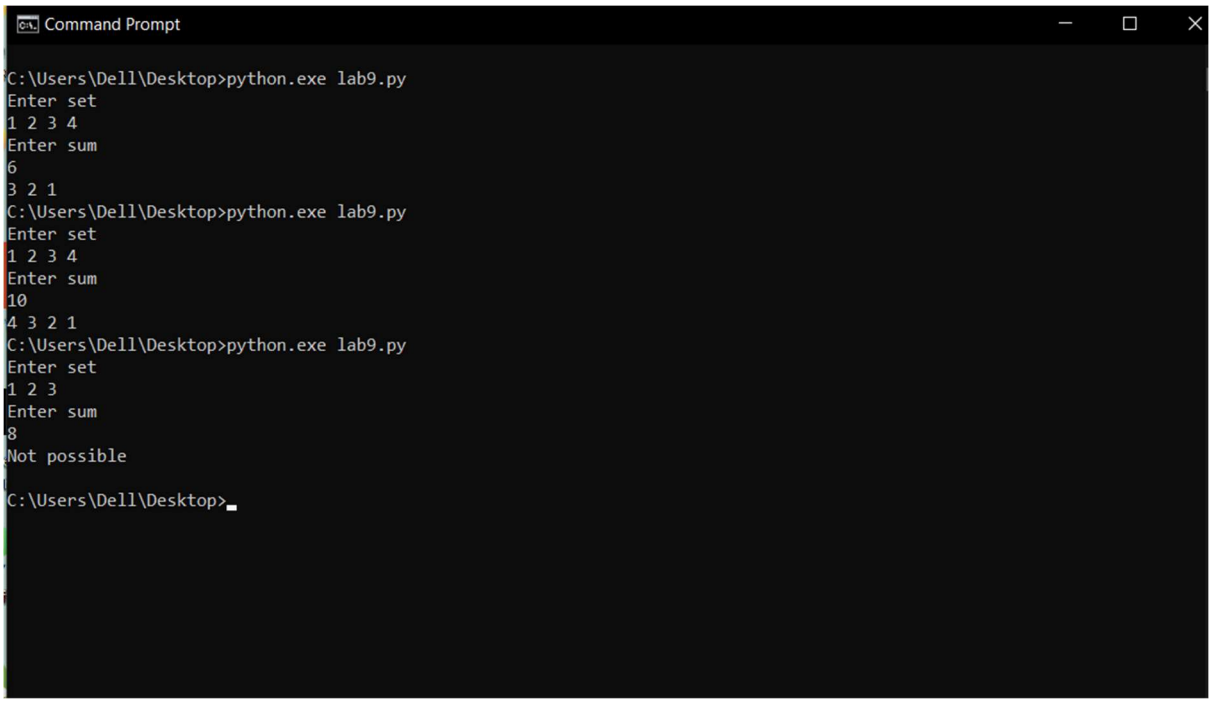
LAB-9

Saloni Patel

- **SUBSET OF SUM**

```
s = input("Enter set\n").split(" ")
m = int(input("Enter sum\n"))
n = len(s)
t = []
p = []
w = []
for i in range(n):
    p.append(int(s[i]))
    w.append(int(s[i]))
for i in range(n+1):
    temp = []
    for j in range(m+1):
        temp.append(0)
    t.append(temp)
for i in range(1, n+1):
    for j in range(1, m+1):
        if j < w[i-1]:
            t[i][j] = t[i-1][j]
        else:
            t[i][j] = max(t[i-1][j], p[i-1]+t[i-1][j-w[i-1]])
if m == t[n][m]:
    i = n
    j = m
    while i > 0 and j > 0:
        if t[i][j] == t[i-1][j]:
            i -= 1
        else:
            j = j-w[i-1]
            i -= 1
            print(s[i], end=" ")
else:
    print("Not possible")
```

Snapshot:



Conclusion:

Here we are solving sum of subset using 0/1 knapsack .

We are considering set of elements as weight and profit array of 0/1 knapsack and capacity constraints is same as sum that we want. So we can reduce sum of subset to knapsack problem