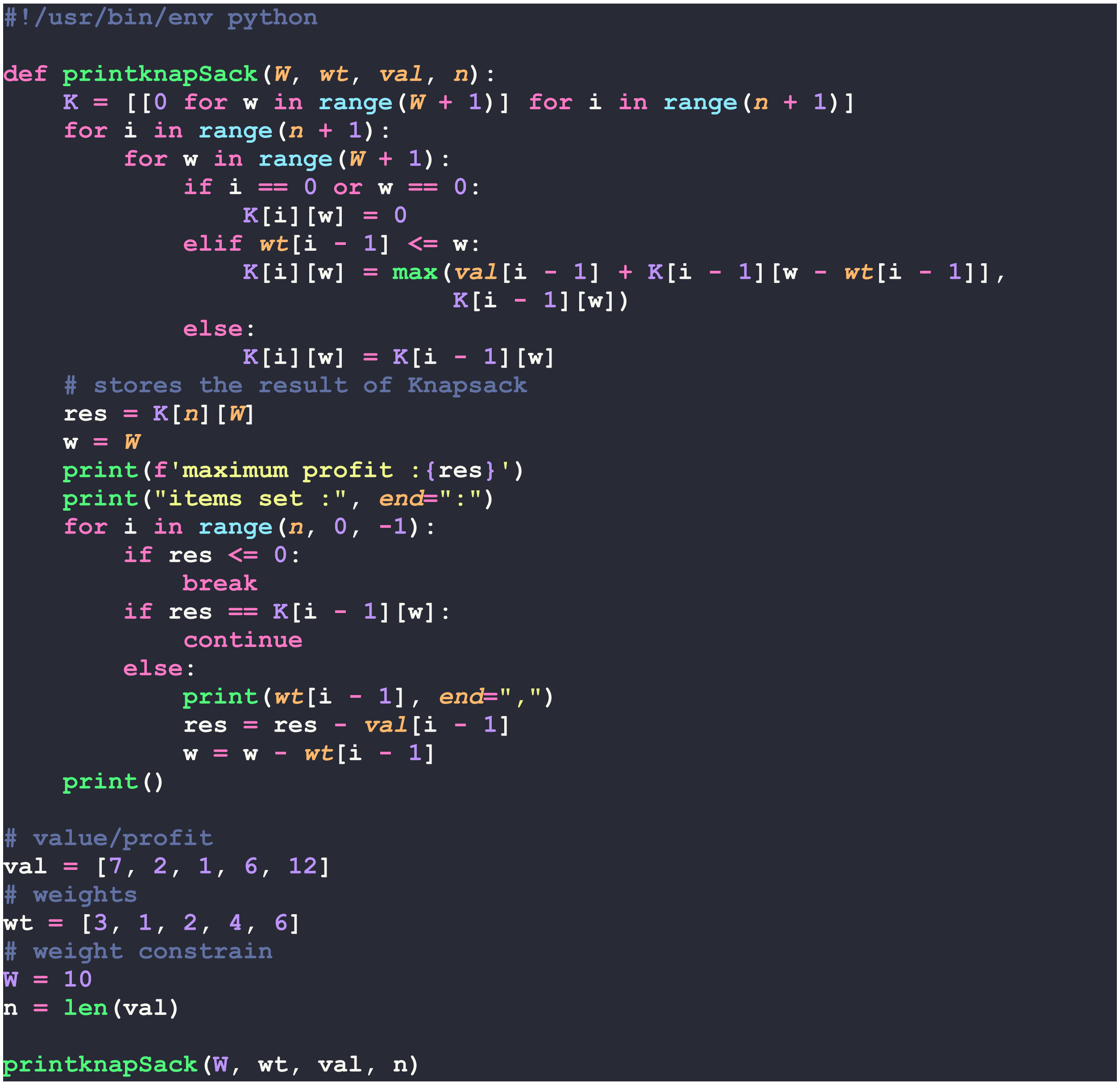
DAA Assignment-6

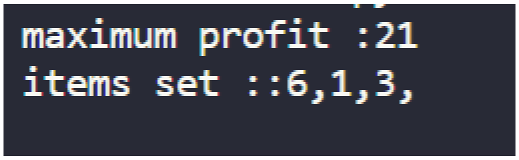
Name :Lohith N
Roll No :201it133

1)**0/1 Knapsack Problem**



```
#!/usr/bin/env python

def printknapSack(W, wt, val, n):
    K = [[0 for w in range(W + 1)] for i in range(n + 1)]
    for i in range(n + 1):
        for w in range(W + 1):
            if i == 0 or w == 0:
                K[i][w] = 0
            elif wt[i - 1] <= w:
                K[i][w] = max(val[i - 1] + K[i - 1][w - wt[i - 1]],
                              K[i - 1][w])
            else:
                K[i][w] = K[i - 1][w]
    # stores the result of Knapsack
    res = K[n][W]
    w = W
    print(f'maximum profit :{res}')
    print("items set :", end=":")
    for i in range(n, 0, -1):
        if res <= 0:
            break
        if res == K[i - 1][w]:
            continue
        else:
            print(wt[i - 1], end=",")
            res = res - val[i - 1]
            w = w - wt[i - 1]
    print()

# value/profit
val = [7, 2, 1, 6, 12]
# weights
wt = [3, 1, 2, 4, 6]
# weight constrain
W = 10
n = len(val)

printknapSack(W, wt, val, n)
```

Output:

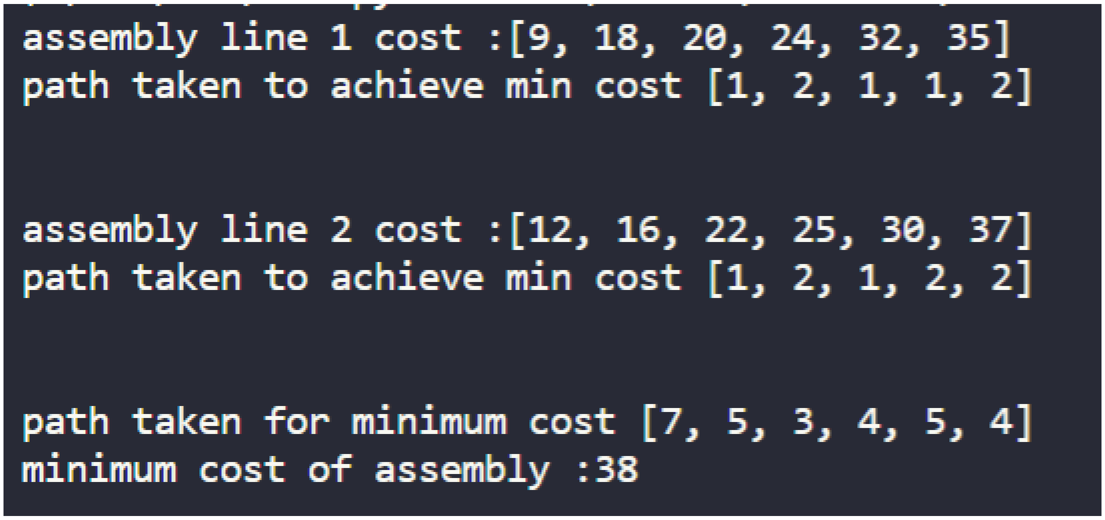


```
maximum profit :21
items set ::6,1,3,
```

2)Assembly Line Scheduling

```python
#!/usr/bin/env python
def carAssembly(a, t, e, x):
    path = []
    NUM_STATION = len(a[0])
    T1 = [0 for i in range(NUM_STATION)]
    T2 = [0 for i in range(NUM_STATION)]

    T1[0] = e[0] + a[0][0]  # time taken to leave

    T2[0] = e[1] + a[1][0]  # time taken to leave

    if T1[0] < T2[0]:
        path.append(a[0][0])
    else:
        path.append(a[1][0])

    l1 = l2 = 0
    line1 = []
    line2 = []
    for i in range(1, NUM_STATION):
        inline = T1[i - 1] + a[0][i]
        skipto2 = T2[i - 1] + t[1][i] + a[0][i]
        l1 = a[0][i]
        if inline < skipto2:
            line1.append(1)
            T1[i] = inline
        else:
            line1.append(2)
            T1[i] = skipto2

        inline = T2[i - 1] + a[1][i]
        skipto2 = T1[i - 1] + t[0][i] + a[1][i]
        l2 = a[1][i]
        if inline < skipto2:
            line2.append(2)
            T2[i] = inline
        else:
            line2.append(1)
            T2[i] = skipto2

        if T1[i] < T2[i]:
            path.append(l1)
        else:
            path.append(l2)

    print(f'assembly line 1 cost :{T1}')
    print(f'path taken to achieve min cost {line1}\n\n')

    print(f'assembly line 2 cost :{T2}')
    print(f'path taken to achieve min cost {line2}\n\n')

    print(f'path taken for minimum cost {path}')
    # consider exit times and return minimum
    sol = min(T1[NUM_STATION - 1] + x[0], T2[NUM_STATION - 1] + x[1])
    #path traced
    print(f'minimum cost of assembly :{sol}')
    return sol

# cost of each stage a[0] lane 1 a[1] lane2
a = [[7, 9, 3, 4, 8, 4], [8, 5, 6, 4, 5, 7]]
# cost of each tarnsfer t[0] lane 1 t[1] lane2
t = [[0, 2, 3, 1, 3, 4], [0, 2, 1, 2, 2, 1]]
# entrence cost
e = [2, 4]
# exit cost
x = [3, 2]

carAssembly(a,t,e,x)
```

Output:



```
assembly line 1 cost :[9, 18, 20, 24, 32, 35]
path taken to achieve min cost [1, 2, 1, 1, 2]


assembly line 2 cost :[12, 16, 22, 25, 30, 37]
path taken to achieve min cost [1, 2, 1, 2, 2]


path taken for minimum cost [7, 5, 3, 4, 5, 4]
minimum cost of assembly :38
```