**Name: Shashank K**

**SE21UCSE198**

**DAA class assignment**

**Program**

def deadline\_driven\_greedy(jobs):

    sorted\_jobs = sorted(jobs, key=lambda x: (x['profit'], -x['deadline']), reverse=True)

    max\_deadline = max(job['deadline'] for job in sorted\_jobs)

    schedule = [-1] \* max\_deadline

    total\_profit = 0

    for job in sorted\_jobs:

        for i in range(job['deadline'] - 1, -1, -1):

            if schedule[i] == -1:

                schedule[i] = job['id']

                total\_profit += job['profit']

                break

    return schedule, total\_profit

jobs = [

    {'id': 'Job1', 'profit': 50, 'deadline': 2},

    {'id': 'Job2', 'profit': 60, 'deadline': 1},

    {'id': 'Job3', 'profit': 20, 'deadline': 3},

    {'id': 'Job4', 'profit': 70, 'deadline': 2},

    {'id': 'Job5', 'profit': 30, 'deadline': 1}

]

schedule, profit = deadline\_driven\_greedy(jobs)

print("Scheduled Jobs:", schedule)

print("Total Profit:", profit)

**Approach**

Deadline based activity selection results in empty time periods being left out. Since we are not making use of this time it would be a waste of CPU power. My approach to this problem is to sort the jobs based on profit as first priority and deadline as second priority. This helps us solve the time gap issue. The simple deadline scheduling algorithm doesn’t consider less profitable options but with this algorithm we can choose some unlikely to be chosen jobs to fill up empty time thus filling up our time array.

Output for the mentioned input  
