Dheeraj Satya Sushant Viswanadham
(z5204820)

COMP3121 Assignment 3 – Question 4

4) We are required to find the subset of jobs that maximises the total profit possible. Our algorithm should run in $O(n^2)$ time.

Since we know that only one job can be scheduled at a time and that since each job takes one unit of time, we can assume the deadlines to be 'integers'.

This helps set out our steps for creating a simple greedy algorithm – first we need to order the jobs in decreasing order of their respective profit value into a list.

Once we have this sorted list, we can then traverse through the list (this takes $O(n)$ time) and for each job, we will need to find whether this job can fit into the latest possible available timeslot that is before the deadline of that job – i.e. we will need to subtract the duration of that job [1 unit of time] from the deadline and find the latest possible timeslot that can cater to this job (again, this also takes $O(n)$ time). This way, we are executing our jobs at a time that will minimally affect the other jobs around it.

However, if we are not able to find a timeslot that is able to cater to that job then we will have to ignore it and simply have to go to the next job in the list. This way, even if we are not able to cater to a particular job, we are still able to get the next best job (which is down the list of profits). This greedy algorithm is quite optimal for our purposes.

Hence, the overall run-time of our algorithm will be $\boldsymbol{O(n^2)}$ as we will be using two for loops – one for traversing through jobs in our sorted list and the other for finding the timeslot which can cater to that job in the list.

Sample Pseudocode:

```
JobScheduler(jobs, deadlines):

    Num = length(jobs) # Find number of jobs in total


    # Sort the jobs in decreasing order of profit


    # Traverse through each of the jobs in the sorted job list and find the latest possible

    # timeslot that can accommodate the job and complete it before the deadline (as below):

    for job in range(length(sortedJobList)):

        for deadline in range(minimum(deadlines – 1, jobs[job][1] – 1), -1, -1):

            # Determine if there is a free slot and if so, schedule the job

            # Otherwise go to the next job
```

End of Solution