z5158334
Yuanzhi Shang

Question 4

Using greedy method, every time we only want to insert the job with maximum profit and then insert next job. If they are conflict, reallocate the location of next job. The algorithm could be described as follow:

1. Sort the jobs based on the profit in descending order into an array.
2. For every job, insert it to the time that just beyond the deadline.
   a) If the time is occupied, find the available time that as close as possible to its deadline.
3. if it can find such a time, discard this job

pseudo code:

```
/* jobs */
struct jobs {
    int index;
    int profit;
    time_t time; /* the time that is allocated be follwing algorithm */
    time_t deadline;
}
/* sort the jobs in descending order of profit */
Array[] = sort_desc(jobs)
/* some empty time slots */
/* maintain a largest time range*/
time[]
/* for every jobs */
for i < (the number of total job):
    /* start from the time just beyond the deadline, j would go back
     * stop at when reach the start of the time[]
     * insert the job as close as possible to the deadline
    */
    for j in range(array[i].deadline - 1, 0, -1):
        if time[j] is empty:
            time[j] is filled;
            array[i].time = j   /* the current jobs has been put in tim
e slot j*/
            break;
```

The above pseudo code shows a algorithm runs in $O(n^2)$