Q1. The following is based on the proof given in *"Complexity of scheduling under precedence constraints"* (Lenstra and Kan, 1978). We prove the NP-hardness by reducing from the *linear arrangement problem*. Given an undirected graph $G = (V, E)$ and an integer k, the linear arrangement problem is to find a linear ordering of the vertices such that the sum of the lengths of all edges is no more than k. Formally, the problem is to find a label assignment $f : V \to \{1, \ldots, |V|\}$ over the vertices such that

$$\sum_{(u,v)\in E} |f(u) - f(v)| \leq k.$$

For each vertex v, create a job with of length 1 and weight $-\deg(v)$ (later we will get rid of the negative weights); for each edges (u, v), create a job of length 0 and weight 2 which may be scheduled only after both u and v have been scheduled. Consider the schedule that minimizes the weighted completion time; define the labeling f by numbering the vertices in the order in which they appear in the schedule (i.e., $f(v)$ is the finishing time of v). We claim that this labeling minimizes $\sum_{(u,v)\in E} |f(u) - f(v)|$, so it can be used to answer the decision problem for the linear arrangement problem. Observe that in the optimal schedule, (u, v) must be scheduled as soon as the later of u and v is finished, i.e, the finishing time of (u, v) is exactly $\max(f(u), f(v))$. Observe that the total weighted completion time is given by

$$\begin{aligned}\sum_{(u,v)\in E} 2\max(f(u), f(v)) - \sum_{v\in V} \deg(v) f(v) &= \sum_{(u,v)\in E} \max(f(u), f(v)) - \min(f(u), f(v)) \\ &= \sum_{(u,v)\in E} |f(u) - f(v)|\end{aligned}$$

Furthermore, notice that any valid labeling f can be translated to a valid schedule with weighted completion time equal to $\sum_{(u,v)\in E} |f(u) - f(v)|$, therefore the optimal schedule corresponds to an optimal linear arraignment. To get rid of the negative weights, we can add n to the weight of each vertex job; it is easy to see that this modification increases the total weighted completion time of every schedule by exactly $n^2(n + 1)/2$, therefore it does not change the optimal schedule.

Q2. We can obtain a 2-approximation of this problem by reducing it to the problem of minimizing the weighted completion time with precedence constraints. The reduction is as follows. We introduce a 0-length job with a weight of 1 for each client, and require this job to be scheduled only after all the jobs, in which the client is interested, have been scheduled. We also assign a weight of 0 to all of the original jobs. It is straightforward to see that the sum of the satisfactions times is equal to the sum of the weighted completion times.

Q3. We present a greedy algorithm that considers the requests in non-decreasing order of their end time and selects each request if doing so does not make the

current selection infeasible. We make use of the following routine for checking feasibility of a subset of requests and also for assigning resources to a feasible subset.

CheckFeasibility(A): for a subset A of requests, this routine checks whether all requests in A can be feasibly satisfied. This is done by sorting the requests in A in non-decreasing order of their starting time and greedily assigning resources to them in that order. If it fails to assign a resource to a request $r \in A$, there must be k other requests active at that time r starts, so A is infeasible. If A is feasible, this routine also gives a valid assignment of resources to requests.

Main algorithm: the main part of the algorithm is as follows.

1. sort and index the requests in non-decreasing order of their end time so that $t_e^1 \leq \cdots \leq t_e^n$.
2. initialize $A \leftarrow \emptyset$
3. for each i, from 1 to n:
   - if CheckFeasibility($A \cup \{i\}$) then $A \leftarrow A \cup \{i\}$.
4. Compute the assignment of resources to requests by running CheckFeasibility(A).

The proof of optimality is by contradiction. Let A be the set of requests satisfied by the greedy, and OPT be the set of requests satisfied by the optimal solution which is most similar to A (i.e., among all optimal solutions, it maximize $|OPT \cap A|$). Let i be the index of the first request in A which is not in OPT. Obviously $OPT \cup \{i\}$ is infeasible. Let i' be the smallest index of any request in $OPT \setminus A$ that conflicts with i. We claim that $OPT' = OPT \cup \{i\} \setminus \{i'\}$ is also a feasible optimal solution which is more similar to A, which contradicts the assumption that OPT was the optimal solution most similar to A, therefore it must be that $A = OPT$. Observe that $i' > i$ by our assumption that i had the smallest index in $OPT \setminus S$, therefore $t_e^i \leq t_e^{i'}$, so removing i' and adding i may not cause any conflict.

Q4. The proof is given in section 3.2 of Aspnes et al.