Question 1

(a) One greedy approach would first give out the job with the highest rating, then remove the person who does it from the list of people available, and remove the job from the list of available jobs. Then it would give out the remaining job with highest rating that one of the remaining people can do and remove that person and job from the list, and so on, until all people are assigned a job.

(b) In the graph given, the greedy person would assign person 1 the job with rating 10. The only job then left for person 2 is the job 2. The total rating of that assignment is 11. Clearly, the optimal solution is to have person 1 doing job 2 and person 2 doing job 1, which gives a total rating of 17.

Question 2

By way of contradiction, suppose that the max-cost edge $e$ of some cycle $C$ is in an MST $T$. Then the removal of $e$ defines a cut in the tree $T$, and some other edge $f \in C$ must cross this cut (and hence connect $T - e$). The graph $T' = T - e + f$ is a spanning tree because it has $n - 1$ edges and is connected, and furthermore, its cost is $c(T') = c(T) - c(e) + c(f) < c(T)$, due to $c(e) > c(f)$. This contradicts $T$ being minimum-cost.

Question 3

Let the suspect’s list of cities $C$ consist of $c_1, \ldots, c_n$, and the killer’s list of cities $D$ consist of $d_1, \ldots, d_m$. We give a greedy algorithm that finds the first city in $C$ that is the same as $d_1$, matches these two cities, then find the first city after this that is the same as $d_2$, and so on. We will use $k_1, k_2, \ldots$ to denote the match we have found so far, $i$ to denote the current position in $C$, and $j$ the current position in $D$. If we never match $d_m$ with any city in $C$ then we return that $D$ is not a subsequence of $C$. Otherwise, we return the subsequence $k_1, \ldots, k_m$. 
The running time is $O(n)$: we check each city in $C$ at most once, to compare it with the city from $D$ that we’re currently trying to match.

It is clear that the algorithm finds a correct match if it finds anything. It is harder to show that if the algorithm fails to find a match, then no match exists. Assume that $D$ is the same as some subsequence $c_i, \ldots, c_m$ of $C$. We prove by induction that the algorithm will succeed in finding a match and will have $k_j \leq l_j$ for all $j = 1, \ldots, m$.

Claim: For each $j = 1, \ldots, m$ the algorithm finds a match $k_j$ such that $k_j \leq l_j$.

Proof. The proof is by induction on $j$. First consider $j = 1$. The algorithm lets $k_1$ be the first city that is the same as $d_1$, so we must have that $k_1 \leq l_1$.

Now consider a case when $j > 1$. Assume that $j - 1 < m$ and assume by the induction hypothesis that the algorithm found the match $k_{j-1}$ and has $k_{j-1} \leq l_{j-1}$. The algorithm lets $k_j$ be the first city after $k_{j-1}$ that is the same as $d_j$ if such an city exists. We know that $l_j$ is such an city and $l_j > l_{j-1} \geq k_{j-1}$. So $c_j = d_j$, and $l_j > k_{j-1}$. The algorithm finds the first such index, so we get that $k_j \leq l_j$.

Question 4

We repeatedly select the shortest remaining request/shower time and add it to the end of the schedule. This runs in time $O(n \log n)$.

To see why this algorithm is correct, suppose by way of contradiction that some other algorithm is optimal. Then there are two showers $i$ and $j$ scheduled one after the other in the optimal solution ($i$ before $j$) despite $t_i > t_j$. Suppose in this schedule, the time shower $i$ completes is $c_i$, and so shower $j$ will complete at time $c_j = c_i + t_j$.

Now, we exchange: we consider the schedule obtained by swapping the order of $i$ and $j$. The new completion time of $i$ is $c'_i = c_j$. But the new completion time of $j$ is $c'_j = c_i + (t_j - t_i) < c_i$. All other completion times remain the same, and so the total sum has gone down. Thus, our other schedule cannot be optimal, a contradiction.