Question 1. (20 marks) We say that a set of intervals \( \{ I_1, I_2, \ldots, I_n \} \) **covers** an interval \( I \) if \( I \subseteq \bigcup_{i=1}^{n} I_i \). We are given an interval \( I = [S, F] \) and a set of intervals \( \{ I_1, I_2, \ldots, I_n \} \) that covers \( I \). We wish to find a subset of \( \{ I_1, I_2, \ldots, I_n \} \) that covers \( I \) and has as few intervals as possible.

(Think of \( I \) as the period during which a building must be guarded, and \( I_1, I_2, \ldots, I_n \) as the intervals in which each of \( n \) guards can be on duty. We want to find the smallest number of guards so that at all times during \( I \) at least one guard is on duty.)

a. Consider the following greedy strategy. Assume that \( I_1, I_2, \ldots, I_n \) are sorted in decreasing order of the length of their intersection with \( I \). (Rename them, if needed.) Start with the empty subset, and consider each interval in turn, keeping the interval under consideration in the subset if and only if it covers a part of \( I \) that isn’t already covered by intervals kept so far. Give a counterexample showing that this strategy does not necessarily yield a desired subset of \( I_1, I_2, \ldots, I_n \).

b. Describe an efficient greedy algorithm to find a subset of \( I_1, I_2, \ldots, I_n \) that covers \( I \) and has as few intervals as possible. Prove that your algorithm is correct, and analyze its running time.

Question 2. (10 marks) Consider the Interval Scheduling problem. We have seen that the “earliest-finish-time-first” greedy algorithm finds an optimal set of jobs. Consider now the “shortest-job-first” greedy
algorithm, which initially sorts the jobs by increasing \textit{duration} — i.e., so that \( f(1) - s(1) \leq f(2) - s(2) \leq \ldots \leq f(n) - s(n) \). It then considers the jobs in this order; each job considered is added to a set \( A \) (initially empty) if and only if it does not overlap with any job currently in \( A \). Thus, at the end, the algorithm has produced a set of non-overlapping jobs \( A \). (In contrast, the “earliest-finish-time-first” algorithm initially sorts the jobs by increasing finish time — i.e., so that \( f(1) \leq f(2) \leq \ldots \leq f(n) \). Other than the initial sorting of jobs, the two algorithms are the same.)

We have seen that the “shortest-job-first” greedy algorithm does not necessarily produce an optimal set of jobs. In this problem you must prove that it produces a set of feasible jobs whose size is at least half the size of an optimal set. Thus, in some sense, the set of jobs produced by this algorithm is not too bad compared to the optimum. (Contrast this to the “earliest-start-time-first” algorithm, which produces a set of jobs whose size can be an arbitrarily small fraction of the optimal.)

More precisely, let \( m \) be the size of an optimal set of jobs, and let \( A \) be the set of jobs computed by the shortest-job first greedy algorithm. Prove that \(|A| \geq m/2|.

\textbf{Hint.} Let \( B \) be any optimal set of jobs and \( j \) be any job in \( A \). What is the maximum number of jobs in \( B \) that \( j \) can overlap?