Solution 1:

1)
   - Hyperparameter tuning means finding the hyperparameter vector $\boldsymbol{\lambda}$ s.t. the model trained by learner $\mathcal{I}$ with hyperparameter vector $\boldsymbol{\lambda}$ on $\mathcal{D}_{\text{train}}$ and evaluated on $\mathcal{D}_{\text{test}}$, results in the lowest estimated generalization error.
   - Tuning is a bi-level optimization problem because the **empirical risk minimization problem** on the level of the learner $\mathcal{I}$ depends on the hyperparameters provided to $\mathcal{I}$, and is therefore nested within the **hyperparameter optimization problem**.

2) An example tuning problem
   - Data set $\mathcal{D}$ (given in the task),
   - A random forest learner $\mathcal{I}$ (given in the task),
   - $d = 2$ hyperparameters of the random forest learner: $\{mtry,\ ntrees\}$ and the configuration space $\boldsymbol{\Lambda} = \{1, 2\} \times \{n \in \mathbb{N} \mid 10 \leq n \leq 1000\}$,
   - accuracy ρ_{ACC} (it must be a measure suited for classification, though)
   - 3-fold cross-validation (CV).

3) This depends on two factors:
   - Since the performance was estimated wrt a model trained on $|J_{\text{train}}| < n$, the estimate is pessimistically biased.
   - If the performance estimate used to evaluate the hyperparameter configurations is used as an estimate of the generalization error, an optimistic bias is introduced. This is because the *untouched test set principle* has been violated. By repeatedly evaluating the learner on the same CV splits, information about the CV splits "leaks" into the evaluation.

   Therefore, the answer depends on which of the two effects is larger in magnitude. To eliminate overfitting to the resampling splits / overtuning, one could apply **nested resampling**, which leaves the performance estimate with only the pessimistic bias due to $|J_{\text{train}}| < n$.