1 Theoretical Part

1. General definition of the problem:
   Recall that our goal is to predict the correct boolean conjunction.
   
   (a) Describe the instance domain (X) and the label set (Y):
       The instance domain is a sequence of bits (0 or 1), of length $d$. The label set is $y \in \{0, 1\}$.
   
   (b) Present the entire hypothesis class for $d=2$:
       \[
       \{ \phi, x_1, \bar{x}_1, x_2, \bar{x}_2, x_1 \land x_2, \bar{x}_1 \land \bar{x}_2, x_1 \land \bar{x}_2, \bar{x}_1 \land x_2, \bar{x}_1 \land \bar{x}_2 \land \bar{x}_2 \} \]
   
   (c) Suggest an equation that describes the size of the hypothesis class for any given $d$:
       A hypothesis could be represented by a set of bits, where every bit stands for one of the following three states: the literal appears in the hypothesis, the literal’s negation appears in the hypothesis, or the literal doesn’t appear in the hypothesis. Additionally, the hypothesis class includes the all negative hypothesis. Thus, the size of the hypothesis class would be: $3^d + 1$.
   
   (d) Suppose that the true conjunction is $\bar{x}_1 \land x_2 \land x_3$:
       i. Can the following example exist in the training set: $((1,0,1,1), 0)$? If not, suggest a correction:
           This example could exist in the training set: $\bar{1} \land 0 \land 1 = 0$
       ii. Can the following examples co-exist in the training set: $((0,1,1,0), 1)$, $((0,1,1,1), 1)$? If not, suggest a correction:
           These examples could co-exist in the training set: $0 \land 1 \land 1 = 1$

2. First Strategy: the Halving Algorithm:
(a) Suggest an implementation of this solution for the problem (provide a general concept - no need to write a code).

$H$ would consist of the entire hypothesis class for a given $d$. The algorithm would iterate over the training examples; for a given example $i$, it would predict $\hat{y}_i$ to be the result given by the majority of the hypotheses. For the next iteration, we discard the hypotheses that predicted incorrectly. Under the assumption that there is a correct hypothesis, in the last iteration, it would be the only one left.

(b) What is the number of mistakes made by $a - M(a)$?

We will start with some notation:

- $Experts_1$: the initial set of hypotheses.
- $Experts_t$: the hypotheses that 'survived' up to iteration $t$.

For every iteration $t$ at which there has been a mistake, at least half of the hypotheses in $Experts_t$ are wrong, and won’t continue into $Experts_{t+1}$.

Therefore, assuming that the 'real' hypothesis exists, at the last iteration $T$ we know that: $1 \leq |Experts_T|$. Further, the size of $|Experts_T|$ is less than or equal to $|H|/2^M$ where $M$ is the number of mistakes the algorithm made until now, the reason being that at every iteration $|Experts_t|$ has less than half of the previous hypotheses. Thus, the hypothesis class $H$ has been divided by 2 (in the worst case), $M$ times.

\[
1 \leq \frac{|H|}{2^M} \quad (1)
\]

\[
log(1) \leq log(|H| \cdot 2^{-M}) \quad (2)
\]

\[
0 \leq log(|H|) + log(2^{-M}) \quad (3)
\]

\[
M \leq log(|H|) \quad (4)
\]

Thus, the number of mistakes made by $a$ is: $M(a) \leq log(|H|) = log(3^d + 1)$, where $|H|$ is the size of the hypothesis class.

(c) What is the run-time per iteration?

In every iteration the algorithm runs through every one of the hypotheses to get its prediction ($\hat{y}_t$). To predict $\hat{y}_t$ it has to go through all $d$ bits; thus meaning that every iteration takes $O(d \cdot |H|)$.

(d) Without assuming realizability - does the solution still hold? What might happen?

Without assuming realizability - the solution obviously doesn’t hold. What might happen is that the set $Experts_t$ in the final iteration would be empty ($\phi$).

3. Second Strategy:
(a) Prove that the number of mistakes made by $a - M(a) \leq d + 1$. (Hints: (1) use induction to prove $M(a) \leq 2d$, (2) Infer the desired bound by observing the outcome of the first iteration in the algorithm).

Prove $M(a) \leq d + 1$:

i. $M(a) \leq 2d$:
   The initial hypothesis is the all negative hypothesis, so it has $2d$ literals.
   Suppose that for every iteration the algorithm predicts incorrectly, and therefore on every iteration it has to remove a certain $x_i$ or $\overline{x}_i$. The maximal number of times it could make a mistake would be $2d$.

ii. In the first iteration, the prediction $\hat{y}$ will always be wrong, because the hypothesis has 2 versions of every literal. In this case, the algorithm will have to enter the if loop in line 3, and will remove a single version of every $x_i/\overline{x}_i$, thus leaving a hypothesis with $d$ literals. In the next iterations, the algorithm could predict wrongly only another $d$ times (at most). Thus, $M(a) \leq d + 1$

(b) What is the run-time per iteration?
   The run-time per iteration: $d$; the algorithm loops over the $d$ bits of the current training example.