1. (Improving guarantees of randomized algorithms)

(a) Solution: Let X denote the number of times needed to get the first non-fail outputs for an input of size n and let $T_n = O(n^2)$ be the time that a single run of the algorithm takes to get an output (either"fail" or correct output). Then the total time of getting a correct answer $T(n) = XT_n$. Let $p \geq 1 - 0.99 = 0.01$ be the probability that the algorithm will output a correct answer in a single run. Then we have $X \sim Geom(p)$. Thus

$$\begin{aligned}
\mathbf{E}[T(n)] &= \mathbf{E}[X]O(n^2) && (1)\\
&= \frac{1}{p}O(n^2) && (2)\\
&\leq \frac{1}{0.01}O(n^2) && (3)\\
&= O(n^2) && (4)
\end{aligned}$$

Note that we only stop our runs of the algorithm when we have a correct output, thus this new algorithm always computes a correct answer and runs in expected time $O(n^2)$.

(b) Solution: Let X be the random variable of the running time of algorithm $\mathcal{B}$ on an input of size n. The using Markov inequality, for any positive number b,

$$\begin{aligned}
Pr[X \geq b] &\leq \frac{\mathbf{E}[X]}{b} && (5)\\
&= \frac{T(n)}{b} && (6)\\
&\leq 1 - 0.95 = 0.05 && (7)
\end{aligned}$$

We can solve for $b \geq 20T(n)$. Thus we can choose $a = 20$ and run the algorithm $\mathcal{B}$ for $20 \cdot T(n)$ time, and the failure probability is less than 0.05 (i.e. with probability at least 0.95, the new algorithm can solve the problem).

If we know the variable of the algorithm to be at most $\sqrt{n}$, then we can use Chebyshev inequality instead of Markov inequality, for any $b > 0$,

$$\begin{aligned}
Pr[|X - \mathbf{E}[X]| \geq b] &\leq \frac{\mathbf{Var}[X]}{b^2} && (8)\\
Pr[X - \mathbf{E}[X] \geq b] &\leq \frac{\mathbf{Var}[X]}{b^2} && (9)\\
Pr[X \geq T(n) + b] &\leq \frac{\mathbf{Var}[X]}{b^2} && (10)\\
&\leq \frac{\sqrt{n}}{b^2} \leq 0.05 && (11)\\
& && (12)
\end{aligned}$$

We can solve for $b \geq 2\sqrt{5}n^{1/4}$. Thus, we can let the $\mathcal{B}$ algorithm run $T(n) + 2\sqrt{5}n^{1/4}$ time, and the success probability will be at least 0.95.

(c) Solution: For the algorithm to succeed if we run $\mathcal{C}$ algorithm k times, since we don't know if the incorrect solutions are the same, for the worst case, we need the number of correct solutions to be at least $0.5k$. Then the possible failure case is when the number of correct solution to be less than $0.5k$. Let X_i to be an indicator variable for correct solution returned at run i, and let $X = \sum_i^k X_i$ be the random variable of the number of correct solutions in k runs. $\mathbf{E}[X] = 0.7k$. Using Chernoff bounds, for $\delta = 2/7$,

$$Pr[X \leq (1-\delta)\mathbf{E}[X]] \leq e^{-\mathbf{E}[X]\delta^2/2} \tag{13}$$

$$Pr[X \leq (1-2/7)0.7k] \leq e^{-0.7k(2/7)^2/2} \tag{14}$$

$$Pr[X \leq 0.5k] \leq e^{-k/35} \leq e^{-t} \tag{15}$$

$$-k/35 \leq -t \tag{16}$$

$$k \geq 35t \tag{17}$$

Thus, if we want the algorithm to make a mistake in computing $f(x)$ with probability at most 2^{-t}, then we need to set $k \geq 35t$.