1. [*15 marks*] A group of hackers from an enemy organization has attempted to install a virus to n of your company's computers. Your software engineers have designed a test, called `TEST-EACH-OTHER`, that takes two computers c_A and c_B, where each input computer tests the other and outputs whether the other one is infected with the virus (+) or not infected with the virus (−). If a computer is actually − than it will always output a correct result. Unfortunately, if it is +, its reply is unrelated to the real state of the other computer and hence cannot be trusted. In other words, a computer c_A that is infected with the virus can be "dishonest" and output the correct or the incorrect state of c_B.

The following table summarizes the four possible outcomes of running `TEST-EACH-OTHER` on two computers c_A and c_B, and what we can conclude from it. Please review the table to ensure that these are indeed the possible outcomes.

c_A's output	c_B's output	Conclusion
c_B is −	c_A is −	either both − or both +
c_B is −	c_A is +	at least one is +
c_B is +	c_A is −	at least one is +
c_B is +	c_A is +	at least one is +

Luckily your security experts have told you that more than $n/2$ computers were not infected (so they are −). Your goal is to identify all the + and − computers. Below, running one instance of `TEST-EACH-OTHER` constitutes one test.

(a) [*12 marks*] Describe an algorithm to find a single − phone by performing $O(n)$ tests. [Hint: Think of how you can use $O(n)$ tests to reduce the problem size by a constant factor.]

(b) [*3 marks*] Using part (a), show how to identify the condition of each computer by performing $O(n)$ tests.

(a) My algorithm is as follow:
We let S be the set of all computers
① If n is odd, we pick one computer C. Then, we use the rest $(n-1)$ computers to test it. Since we have more than half (−) computers, if there are equal or more than half (−) results, C is (−). Otherwise, C is (+).
If C is (−), we find one (−) computer.
If C is (+), we drop it and work on the rest.
② If C is even, we will try to reduce the size of S. We divide S into $\frac{n}{2}$ pairs of computers. Then, we test each pair. We only keep one in each pair with (−) result and make them new S. Then repeat algorithm from ①.
The worst case is that ② can only reduce n into $\frac{n}{2}$. Therefore,

$$T(n) = n + \frac{n}{2} + T(\frac{n}{2})$$
$$= n + \frac{n}{2} + \frac{n}{2} + \frac{n}{4} + \frac{n}{4} + \frac{n}{8} + \cdots$$
$$= n + n + \frac{n}{2} + \frac{n}{4} + \cdots = 3n \in O(n)$$

(b) Once we find a (−) computer, we will use it to test the rest computers.
Thus, $T(n) = O(n) + O(n) \in O(n)$

2. [*12 marks*] Consider the recurrence:

$$T(n) = 2T(\lfloor n/9 \rfloor) + \sqrt{n} \qquad \text{if } n \geq 9$$

$$T(n) = 5 \qquad \text{if } n < 9$$

Prove $T(n) = O(\sqrt{n})$ by induction (i.e., guess-and-check or substitution method). Show what your c and n_0 are in your big-oh bound. Note that depending on the choice of your n_0, you might have to cover multiple base cases in your inductive proof.

Proof:

Guess: $T(n) \leq 8\sqrt{n}$ for $n \geqslant 1$

- Base Cases:

  $n < 9$, $T(n) = 5 \leq 8 \cdot \sqrt{1} \leq \cdots \leq 8 \cdot \sqrt{8}$

  $n = 9$, $T(n) = 2 \cdot T(9/9) + \sqrt{9} = 13 \leq 24$

  Thus, base cases hold.

- Induction Hypothesis:

  Suppose $T(n) \leq 8 \cdot \sqrt{n}$ for $n \leq k-1$

- Induction Conclusion:

  For $n = k$,

  $T(k) = 2 \cdot T(\lfloor k/9 \rfloor) + \sqrt{k} \leq 2 \cdot 8 \cdot \sqrt{k/9} + \sqrt{k}$ by IH.

  $= \frac{19}{3}\sqrt{k} \leq 8\sqrt{k}$

  Thus, IC holds.

Therefore, by POMI $T(n) \leq 8\sqrt{n}$ for $n \geqslant 1$. $(c = 8, n_0 = 1)$

Therefore, $T(n) \in O(\sqrt{n})$

3. [*16 marks*] Give tight asymptotic (Θ) bounds for the solution to the following recurrences by using the recursion-tree method or the induction method (your choice). You may assume that n is a power of 10 in (a), or a power of 3 in (b). Show your work.

(a) [*8 marks*]

$$T(n) = \begin{cases} 2\,T(n/10) + \sqrt{n} & \text{if } n > 1 \\ 7 & \text{if } n \le 1 \end{cases}$$

(b) [*8 marks*]

$$T(n) = \begin{cases} 10\,T(n/3) + n^2 & \text{if } n > 1 \\ 1 & \text{if } n \le 1 \end{cases}$$

(a) Guess: $2\sqrt{n} \le T(n) \le 16\sqrt{n}$ for $n \ge 1$

- Base cases:

  $n=1$, $T(n)=7$, $2 \le 7 \le 16$

  $n=10$, $T(n) = 2\cdot T(10/10) + \sqrt{n} = 2\cdot 7 + \sqrt{10}$, $2\sqrt{10} \le 14+\sqrt{10} \le 16\cdot\sqrt{10}$

  Thus, base cases hold.

- Induction Hypothesis:

  Suppose $2\sqrt{n} \le T(n) \le 16\sqrt{n}$ for $n \le 10^{k-1}$

- Induction Conclusion:

  For $n = 10^k$, $T(n) = 2\cdot T(k/10) + \sqrt{k}$

  From IH, we get

  $$2\sqrt{k/10} \le T(k/10) \le 16\sqrt{k/10}$$

  Thus,

  $$2\sqrt{k} \le \left(\frac{4}{\sqrt{10}}+1\right)\sqrt{k} \le 2\cdot T(k/10)+\sqrt{k} \le \left(\frac{32}{\sqrt{10}}+1\right)\sqrt{k} \le 16\sqrt{k}$$

  Therefore, IH holds.

And so, by POMI, $2\sqrt{n} \le T(n) \le 16\sqrt{n}$ for $n \ge 1$

Therefore, $T(n) \in \Theta(\sqrt{n})$.

(b) $T(n) = 10\cdot T(n/3) + n^2$ $(n>1)$, $T(1)=1$

Level 0: $T(n)$ $\Rightarrow n^2$

Level 1: $T(n/3)$ $T(n/3)$ $\cdots$ $T(n/3)$ $\Rightarrow 10\cdot(\frac{n}{3})^2$

$\vdots$

Level($\log_3 n$) $T(1)$ $\cdots$ $T(1)$ $\Rightarrow 10^{\log_3 n}\cdot 1$

Thus, the total work $= n^2 + 10\cdot\frac{n^2}{9} + \cdots + 10^{\log_3 n}\cdot 1$

Since $10^{\log_3 n} > (3^2)^{\log_3 n} = (n)^2$, $10^{\log_3 n}$ is the dominant term.

Therefore, $T(n) \in \Theta(10^{\log_3 n})$ / $(T(n) \in \Theta(n^{\log_3 10}))$

4. (a) By master theorem, from $T(n) = 2T(\frac{n}{10}) + \sqrt{n}$ we can get,

$$\begin{cases} a = 2 \\ b = 10 \\ d = \frac{1}{2} \end{cases}$$

Since $a < b^d$, $T(n) \in \Theta(n^{\frac{1}{2}})$

(b) By master theorem, from $T(n) = 10 \cdot T(\frac{n}{3}) + n^2$ we can get,

$$\begin{cases} a = 10 \\ b = 3 \\ c = 2 \end{cases}$$

Since $a > b^d$, $T(n) \in \Theta(10^{\log_3 n}) = \Theta(n^{\log_3 10})$