1. (12 pts) Short Answers.

(a) (4 pts) Suppose that \( Q \) is a decision problem. How do we express \( Q \) as a language?

\[
L = \text{the set of strings } x \text{ for which the answer to } Q(x) \text{ is } \text{Yes}
\]

(b) (4 pts) How is the input given to a Turing machine? What is the size of the input?

It is written on the tape. 

The length is the number of cells containing input characters.

(c) (4 pts) State Church's Thesis.

Anything that we would reasonably call a logical calculation can be modelled by a Turing machine.
2. (5 pts) The following algorithm tests whether a given integer can be expressed as the sum of three perfect squares:

Input: \( n \) (a positive integer)
for \( x = 0 \) to \( n \)
    for \( y = 0 \) to \( n \)
        for \( z = 0 \) to \( n \)
            if \( x^2 + y^2 + z^2 = n \) then return YES
return NO

Is this a polynomial time algorithm? Explain your answer.

\textbf{NO}

The running time is \( O(n^3) \) which is polynomial in the value of the input. It needs to be a polynomial in \( \log n \), which is the size of the input.
3. (19 pts) Consider the following language.

\[ A = \{ <P>, k \} | \text{there are fewer than 10 even integers } x \geq k \text{ such that } P(x) \text{ returns a value in the range } [x^2 + k + 10, x^2 + 2k + 10] \} \]

(a) (15 pts) Prove that \( A \) is not decidable.

Assume that \( A \) is decidable, and that a Turing machine \( M \) decides \( A \).

Then the following Turing machine decides the Halting problem:

---

Input \( <P>, y \)

Create the following description of a Turing machine:

\[ <B> = " \text{Input } x. \]

\[
\text{Run } U(<P>, y) \quad \# U \text{ is the universal T. M.} \\
\text{return } x^2 + 13 \quad \# \text{ is the universal T. M.}
\]

Run \( M(<B>, 2) \) to determine whether \( <B>, 2 \) \( \in A \)

If \( <B>, 2 \) \( \in A \) then \( \rightarrow 9R \) else \( \rightarrow 9A \)

---

We can create \( <B> \) by copying the input \( <P>, y \) into the appropriate part of the text.

Every step takes finite time, since \( M \) is a decider.

If \( P(y) \) halts then \( B(x) \) returns \( x^2 + 13 \) for every \( x \).

So it returns a number in the range \([x^2 + 12, x^2 + 19]\) for at least ten values of \( x \geq 2 \) and so \( <B>, 2 \) \( \in A \).

If \( P(y) \) does not halt then \( B(x) \) returns a value for zero inputs \( x \). Since \( Q < 10 \), \( <B>, 2 \) \( \in A \).

So this \( M \) returns the correct answer and decides the Halting problem. But the H.P. is undecidable, CONTRADICTION.

\( \therefore A \) is not decidable.
(b) (4 pts) Either $A$ or $\bar{A}$ is recognizable. Say which one you think is recognizable. You don't need to prove your answer. But write a few sentences explaining why you think it is the correct choice.

$\bar{A}$ is recognizable. I think this because recognizing $\bar{A}$ requires recognizing what $P(x)$ returns for only 10 values of $x$. I think we should be able to find those 10 values using dovetailing.
4. (16 pts) Give two proofs that the following problem is in NP.
One proof must use a verification algorithm, and the other proof must use a nondeterministic Turing machine.

**TRIPLE-SUM**

**Input:** A list of positive integers \(a_1, \ldots, a_n\).

**Question:** Can the list be partitioned into two parts \(A_1, A_2\) such that the sum of the numbers in \(A_1\) is exactly three times the sum of the numbers in \(A_2\)?

For example, if the input is 7,3,1,8,1 then the answer is YES \((A_1 = 7, 8)\).
If the input is 5,4,4,7,5 then the answer is NO.

**Verification algorithm:**
The certificate will be a partition where both parts have the same sum.

**Input:** A list \(A = a_1, \ldots, a_n\), two other lists \(A_1, A_2\)

1) Check that \(A_1, A_2\) is a partition of \(A\)
   For \(i = 1\) to \(n\), delete \(a_i\) from either \(A_1\) or \(A_2\) (there may be multiple copies of the integer \(a_i\)).
   If \(a_i\) is not in \(A_1\) or \(A_2\), then reject.
   If \(A_1 \neq \emptyset\) or \(A_2 \neq \emptyset\), then reject.

2) Reset \(A_1, A_2\); the sets given as input.
Compute the sum of the elements in \(A_1\) and the sum of the elements in \(A_2\).

If \(\text{sum}(A_1) = 3 \cdot \text{sum}(A_2)\) then ACCEPT
else REJECT.

Step 1 takes \(O(n)\) time. Step 2 can be done in \(poly(\log a_1, \log a_2, \ldots, \log a_n)\) time. Both are polynomial in the size of the input file.

If \(\text{TRIPLE-SUM}(A) = \text{YES}\) then there is a partition \(A_1, A_2\) with \(\text{sum}(A_1) = 3 \cdot \text{sum}(A_2)\). The algorithm will accept \((A, (A_1, A_2))\).
If \(\text{TRIPLE-SUM}(A) = \text{NO}\) then there is no partition of \(A\), where \(\text{sum}(A) = 3 \cdot \text{sum}(A_2)\), so the algorithm will reject \((A, (A_1, A_2))\) for every \(A_1, A_2\).
NDTM:

Input: \( A = a_1, \ldots, a_n \)
\( A_1 = A_2 = \emptyset \)
For \( i = 1 \) to \( n \)
put \( a_i \) into either \( A_1 \) or \( A_2 \)
compute the sum of \( A_1 \) and the sum of \( A_2 \)
if \( \text{sum}(A_1) = 3 \text{sum}(A_2) \) then accept
else reject.

The non-deterministic loop takes \( O(n) \) steps.
The two sums can be computed in \( \text{poly}(\log a_1 + \log a_2 + \ldots + \log a_n) \) steps.
Both are polynomial in the size of the input file.

If \( \text{TRIPLE-SUM}(A) = \text{YES} \) then there is a partition \( A_1, A_2 \)
with \( \text{sum}(A_1) = 3 \text{sum}(A_2) \). It is possible that the
non-deterministic loop will choose this partition and
so the algorithm will accept.

If \( \text{TRIPLE-SUM}(A) = \text{NO} \), then no matter what partition
is chosen, we will have \( \text{sum}(A_1) \neq \text{sum}(A_2) \) and so
the algorithm will reject.
5. (12 pts) Recall that the Halting Problem is:

Input: \(<P>\), the description of an algorithm (i.e. Turing Machine), and a string \(X\).

Question: Does \(P\) halt when given \(X\) as input?

Prove that there is no algorithm (i.e. Turing machine) that solves the Halting Problem.
Do not use a reduction; in other words, you cannot make use of the fact that some other language is not decidable.

Hint: You saw this proof in the second lecture.

Suppose that Turing machine \(M\) decides the Halting Problem.

Then we can use \(M\) to create the following TM

\[
\langle D \rangle
\]

Input \(<P>\)

1) Run \(M(<P>,<P>)\)

2) If \(M(<P>,<P>) \rightarrow \text{accept}\) then loop forever.

3) If \(M(<P>,<P>) \rightarrow \text{reject}\) then halt accept

\(M(<P>,<P>)\) takes finite time since \(M\) is a decider.

Consider running \(D(<D>)\)

If \(D(<D>)\) halts then \(M(<D>,<D>) \rightarrow \text{accept}\) so \(D(<D>)\) enters an infinite loop in line 2.

If \(D(<D>)\) does not halt then \(M(<D>,<D>) \rightarrow \text{reject}\) so \(D(<D>)\) halts in line 3.

\text{CONTRADICTION}

\(M\) does not exist, i.e. the HP is not decidable.