Problem 1

For each of the following sets, give the smallest complexity class in which it is contained. Your choices are:

i. DECIDABLE

ii. UNDECIDABLE but RECOGNIZABLE

iii. UNDECIDABLE and UNRECOGNIZABLE but its compliment is RECOGNIZABLE

iv. UNDECIDABLE and both it and its compliment are UNRECOGNIZABLE

Part A: \{\langle M \rangle \mid M \text{ accepts fewer than } 13 \text{ inputs}\}

**Undecidable and Unrecognizable but its compliment is Recognizable.**

To recognize the compliment of this language, build a machine which, for given \langle M \rangle, runs \(M\) stepwise on all inputs. If \(M\) ever accepts 13 or more, accept.

This set cannot be recognized because we cannot determine the behavior of a given \(M\) over all inputs. Suppose \(M\) accepts 12 inputs and loops on all others. Then there is no finite procedure for determining that \(M\) is in the set, because we have no general finite procedure for identifying loops.

Part B: \{\langle M \rangle \mid M \text{ accepts greater than } 51 \text{ inputs}\}

**Undecidable but Recognizable.**

The recognition procedure for this set is similar to Part A. Build a machine which, for given \langle M \rangle, runs \(M\) stepwise on all inputs. If \(M\) ever accepts more than 51 inputs, accept.

This set is not decidable because, again, we cannot determine the behavior of a given \(M\) over all inputs. Suppose \(M\) accepts 10 inputs and loops on all others. Then, again, there is no finite procedure for determining that \(M\) is in the set, because we have no general finite procedure for identifying loops.

Part C: \{\langle M \rangle \mid M \text{ accepts exactly } 7000 \text{ inputs}\}

**Unrecognizable and both it and its compliment are Unrecognizable.**

This set cannot be recognized because, if a machine \(M\) does accept exactly 7000 inputs, we may not be able to determine its behavior on all other inputs. Suppose \(M\) accepts 7000 inputs and loops on all others. Since we have no general finite procedure for identifying loops, in this case we cannot determine that \(M\) is in the set.

The set’s compliment also cannot be recognized. If \(M\) accepts more than 7000 inputs, its membership in the set can be determined. However, suppose \(M\) accepts 6999 inputs and loops on all others. Again, because we have no general finite procedure for identifying loops, in this case we cannot determine that \(M\) is not in the set.
**Part D:** $\{\langle M, w \rangle \mid M \text{ accepts } w \text{ and uses at most 111 cells on the tape}\}$  
Decidable.

This set is decidable. For a given $M$, there are a finite number of possible inputs, and configurations on those inputs, on a 111 cell tape. So, for each input, we can run $M$ through all of its finitely many configurations and check if it ever halts in an accepting state. If so accept, else reject.

**Part E:** $\{\langle M, w \rangle \mid M \text{ accepts } w \text{ within 1130 steps}\}$  
Decidable.

To decide this set, we build a machine $N$ which takes a given $\langle M, w \rangle$ and runs $M$ on $w$ for 1130 steps. If during those steps, $M$ halts and accepts, $N$ accepts. Otherwise, it rejects.
Problem 2

Prove that there exists an undecidable subset of \( \{1\}^* \).

(A language over an alphabet of one symbol is called a Tally language.)

Proof idea

We present an encoding of Turing machines into unary numerals. We define a subset, \( A \), of \( \{1\}^* \) to be the set of all machine encodings which accept their own machine code. We thereby reduce the decidability of \( K \) to the decidability of \( A \). Since \( K \) is not decidable, \( A \) is not decidable.

Proof. Let \( E \) be a computable enumeration of the set \( TM_C \) - the set of all Turing machine codes:

\[
E = \langle M_1 \rangle, \langle M_2 \rangle, \langle M_3 \rangle, \langle M_4 \rangle, \langle M_5 \rangle, \ldots
\]

Let the function \( f : TM \to \{1\}^* \) such that \( f(\langle M_i \rangle) \) is a sequence of \( i \) ‘1’s:

\[
f(M_i) = 111111 \ldots 111
\]

Define the set \( A \):

\[
A = \{ f(x) \mid \text{for some } y, x = \langle M_y \rangle \text{ and } M_y \text{ accepts } \langle M_y \rangle \}\]

Clearly \( A \subseteq \{1\}^* \).

Lemma : \( K \leq_m A \). By definition, \( f \) satisfies the property

\[
x \in K \iff f(x) \in A
\]

So, we need only to show that \( f \) is computable. Let the functions \( g : TM \to \mathbb{Z}^+ \) and \( h : \mathbb{Z}^+ \to \{1\}^* \) (where \( \mathbb{Z}^+ \) is the set of positive integers) be defined:

\[
\begin{align*}
g(M_x) &= \langle M_1 \rangle, \langle M_2 \rangle, \langle M_3 \rangle, \langle M_4 \rangle, \ldots \\
h(g(M_x)) &= 1, 11, 111, 1111, \ldots
\end{align*}
\]

\[
f(x) = h(g(x)).
\]

So, if \( g \) and \( h \) are computable, \( f \) is computable.

To show \( g \) computable, we construct a machine \( G \) on input \( \langle M_y \rangle \in TM_C \). \( G \) lists all members of \( TM_C \) in order \( E \) and increments a counter for each member. When \( \langle M_x \rangle \) is found, \( G \) returns the value of the counter. ¹

To show \( h \) computable, we construct a machine on input \( \langle i \rangle \), for \( i \in \mathbb{Z}^+ \), which outputs ‘1’ \( i \) times.

Since \( f \) is computable, \( K \leq_m A \).

By the lemma, \( A \) is decidable only if \( K \) is decidable. Therefore, since \( K \) is not decidable, \( A \) is not decidable.

¹Providing a rigorous proof that \( g \) is computable would require an explicit encoding procedure for TM’s, which we have not provided.
Problem 3

Show that a language, $L$, is decidable if and only if $L \leq_m \{1\}^\ast$.

Proof Idea

We suppose that $L$ is decidable, and show that $L \leq_m \{1\}^\ast$. We use $L$’s decider to prove the reduction. The function $f(x)$ whose value is 1 if $x$ is in $L$, and 0 otherwise serves as the reduction. We then suppose that $L \leq_m \{1\}^\ast$ and prove $L$ is decidable from the fact that $\{1\}^\ast$ is decidable.

Lemma 1: if $L$ is decidable, then $L \leq_m \{1\}^\ast$.

Proof. Suppose $L$ is decidable. Then there is some machine $M_L$ that decides it. Define a function $f : L \rightarrow \{1\}^\ast$ as above:

$$f(x) = \begin{cases} 1 & \text{if } M_L \text{ accepts on } x \\ 0 & \text{if } M_L \text{ rejects on } x \end{cases}$$

$f$ is computed by the following valid machine $M$:

Define $M$: "on input $x$

i. Run $M_L$ on $x$

ii. If $M_L$ accepts, OUTPUT 1.

iii. If $M_L$ rejects, OUTPUT 0."

Therefore, $f$ is computable.

Suppose $x \in L$. Then $M_L$ accepts on $x$ and $f(x) = 1 \in \{1\}^\ast$.

Suppose $x \notin L$. Then $M_L$ rejects on $x$ and $f(x) = 0 \notin \{1\}^\ast$.

Therefore, $L \leq_m \{1\}^\ast$. □

Lemma 2: if $L \leq_m \{1\}^\ast$, then $L$ is decidable.

Suppose $L \leq_m \{1\}^\ast$. It is trivial to show that $\{1\}^\ast$ is decidable—simply check to make sure every symbol in the input is a ‘1’. Since $\{1\}^\ast$ is decidable, and $L \leq_m \{1\}^\ast$, $L$ is decidable.

Proof. (of theorem)

By lemmas 1 and 2, $L$ is decidable if and only if $L \leq_m \{1\}^\ast$. □
Problem 4

Show that for any language $A$, a language $B$ exists, where $A \leq_T B$ and $B \not\leq_T A$.

Proof idea

We define $B$ to be the set $K^A$. That is, the set of machines $M^A$ with oracle access to $A$ that accept their own machine code $\langle M \rangle$.

To show $A \leq_T B$, we construct a machine $M_A^B$ which accepts on input $x$ just when $x \in A$. By querying the oracle for $B$ on $M_A$, we can determine whether or not $x$ is a member of $A$.

We show $B \not\leq_T A$ by contradiction. We suppose $B \leq_T A$ and infer the existence of an $A$-oracle decider for $B$. Given such a decider, we construct a diagonal machine $D^A$ which is a member of $B$ if and only if the decider does not accept. This gives us our contradiction.

Let $B = \{ \langle M \rangle \mid M^A \text{ accepts } \langle M \rangle \}$.

Lemma 1: $A \leq_T B$

Define $M_A^B$: "on input $x"

i. Construct $N$: ‘on input $y$

(a) Query $x \in A$.
(b) If ‘yes’, ACCEPT.
(c) If ‘no’, REJECT.’

ii. Query $N \in B$.

iii. If ‘yes’, ACCEPT.
iv. If ‘no’, REJECT."

$M_A^B$ decides $A$.

Suppose $x \in A$. Then $N$, given oracle access to $A$, accepts all inputs (including its own machine code $\langle N \rangle$). So $M_A^B$'s oracle to $B$ responds ‘yes’ on $N$. So $M_A^B$ accepts.

Suppose $x \not\in A$. Then $N$, given oracle access to $A$, rejects. So $M_A^B$'s oracle to $B$ responds ‘no’ on $N$. So $M_A^B$ rejects.

Therefore, $A \leq_T B$. 
Lemma 2: $B \not\leq_T A$. [By contradiction.]

Suppose $B \leq_T A$. Then there is a machine $M^A_B$ which has oracle access to $A$ and decides $B$.

Define $D^A$: "on input $x$

i. Run $M_B$ on input $x$

ii. If $M_B$ accepts, REJECT.

iii. If $M_B$ rejects, ACCEPT."

Suppose $\langle D \rangle \in B$. Then $D^A$ accepts on $\langle D \rangle$ and $M^A_B$ accepts on input $\langle D \rangle$. Since, $M^A_B$ accepts, $D^A$ rejects on input $\langle D \rangle$. So, $\langle D \rangle \notin B$.

Suppose $\langle D \rangle \notin B$. Then $D^A$ does not accept on input $\langle D \rangle$, so $M^A_B$ rejects on $\langle D \rangle$. Since $M^A_B$ rejects, $D^A$ accepts on $\langle D \rangle$.

Therefore, $B \leq_T A$.

Proof. By Lemmas 1 and 2, $A \leq_T B$ and $B \not\leq_T A$. □
Problem 5

Recall the following definition from the textbook

The minimal description of a binary string, \( x \), written \( d(x) \) is the shortest length and lexicographically first string of the form \( \langle M, w \rangle \) where Turing machine \( M \) on input \( w \) halts with \( x \) on its tape. The Kolmogorov complexity of a binary string \( x \) denoted \( K(x) \), is the length of \( d(x) \), \( |d(x)| \).

Show that \( K(x) \) can be computed by a procedure that has access to a Turing machine with oracle access to \( A_{TM} \).

Proof idea

We place an upper bound on the number of machine-input pair strings that could possibly be the shortest such pair that halts with \( x \) on the tape. With this upper bound, we can infer a finite number of strings that must be checked. For each of these finite strings, there is a finite number possible ways to partition the string into machine and input components. Since, in the computation of \( K(x) \), the number of elements that must be checked is finite, and we have a finite procedure for deciding each of them, \( K(x) \) is computable.

\textit{Lemma :} \( K(x) \leq |x| + c \), where \( c \) is some constant.

Define \( M \): "on input \( w \)

i. ACCEPT."

Let \( c \) be the length of \( M \)'s description. Clearly \( M \) halts with \( x \) on its tape when given \( x \) as input. Since \( \langle M, x \rangle \) satisfies this condition and has length \( |x| + c \), the shortest such description will be at most as long as \( \langle M, x \rangle \).

Proof. Let \( m = |x| + c \). Also, let \( f \) be the reduction from \( HALT \) to \( A_{TM} \).

Define \( M^{ATM} \): "on input \( x \)

i. \( s := m. \)

ii. For each of the first \( m \) binary strings \( y \):

(a) For \( i = 0 \) to \( |y| = k \)

i. Partition \( y \) into substrings \( a = y_1y_2...y_i \) and \( b = y_{(i+1)}y_{(i+2)}...y_k \).

ii. Query \( f(\langle a, b \rangle) \in A_{TM}. \)

iii. If \('yes', \) and if \( j = |\langle a, b \rangle| < s. \)

A. Run \( a \) on \( b. \)

B. If \( a \) halts with \( x \) on the tape, \( s := j. \)

C. If \( a \) halts in any other configuration, CONTINUE.

iv. If \('no', \) CONTINUE.

iii. Output \( s. \)"

7
$M^{A_{TM}}$ computes $K(x)$.

Suppose $K(x) = y$. Then $y$ is the length of the shortest $|\langle z, w \rangle| \leq m$ such that $z$ halts on input $w$ with $x$ on the tape. So $\langle z, w \rangle \in HALT$ and $f(\langle z, w \rangle) \in A_{TM}$. So $\langle z, w \rangle$ will be among the first binary strings of length $m$ or less, and the oracle responds ‘yes’ when queried on $\langle z, w \rangle$. So, $M^{A_{TM}}$ runs $z$ on $w$ and, since $z$ halts with $x$ on the tape, $s$ is assigned $y$. Since, by supposition, there are no shorter strings which satisfy the condition, $M^{A_{TM}}$ outputs $y$.

Suppose $M^{A_{TM}}$ outputs $y$. Then $y$ is the shortest string that can be divided into substrings $z$ and $w$ such that $z$ halts on input $w$ with $x$ on the tape. Therefore, $K(x) = y$.

$M^{A_{TM}}$ is a well-defined oracle Turing machine. Each process of $M^{A_{TM}}$ is finite, relying on the $A_{TM}$ oracle to perform all infinite tasks.

Therefore, $K(x)$ can be computed by a procedure that has access to a Turing machine with oracle access to $A_{TM}$. □