Problem 1

For each of the following sets, give the smallest complexity class in which it is contained. Classes are listed from smallest to largest. Write your choice A through D under each numbered language.

Your choices are:

A. DECIDABLE
B. UNDECIDABLE but RECOGNIZABLE
C. UNDECIDABLE and UNRECOGNIZABLE but its complement is RECOGNIZABLE
D. UNDECIDABLE and both it and its complement are UNRECOGNIZABLE

I: \{⟨M, x, y⟩ | M accepts x and rejects y⟩\}
Solution: (B) UNDECIDABLE but RECOGNIZABLE
Consider that M accepts x and M rejects y are, individually, recognizable but undecidable problems. So, in order to verify that ⟨M, x, y⟩ is in the language, we can run one recognizer, and then the other.
However, in order to determine that ⟨M, x, y⟩ is not in the language, we must determine that M does not accept on x or that M does not reject on y, and we already know that both are impossible.

II: \{⟨M₁, M₂⟩ | M₂ rejects every input that M₁ accepts\}
Solution: (D) UNDECIDABLE and both it and its complement are UNRECOGNIZABLE
In order to determine that ⟨M₁, M₂⟩ is in the language, we must determine for every input s that either M₁ does not accept s, or M₂ rejects s. If M₂ happens to accept all inputs, then we must determine that M₁ does not accept any inputs. Since M₁ might be looping on any of them, we know this is impossible.
In order to determine that ⟨M₁, M₂⟩ is not in the language, we would need to find some input on which M₁ accepts and M₂ does not reject. Since M₂ might be looping, we know this is impossible.

III: \{⟨M₁, M₂⟩ | ∃x, y s.t. M₁ accepts x and rejects y, but M₂ rejects x and accepts y\}
Solution: (B) UNDECIDABLE but RECOGNIZABLE
If ⟨M₁, M₂⟩ is in the language, then we can build a machine that finds the first xᵢ inputs on which M₁ accepts and yⱼ inputs on which M₁ rejects. Then we can run M₂ on all of these inputs for i · j steps. If it fails to accept one of the x’s or reject one of the y’s, then we can find the next x or y and try again. Eventually, this procedure will succeed.
However, in order to determine that ⟨M₁, M₂⟩ is not in the language, we must determine that for every pair of inputs, one of them is not accepted (or not rejected) by one of the machines, and we know this is impossible.
IV: \{\langle M \rangle \mid M \text{ only rejects strings beginning with } 0\}
Solution: (C) UNDECIDABLE and UNRECOGNIZABLE but its compliment is RECOGNIZABLE

Consider the compliment: \( M \) rejects some string that does not begin with ‘0’. We can see this is recognizable: run \( M \) on the first \( i \) strings starting with 1 for \( i \) steps and wait for it to reject.

However, in order to determine that \( \langle M \rangle \) is not in the compliment, we would have to determine that for every string starting with ‘1’, \( M \) does not reject it. Since \( M \) could be looping on any of those strings, we know this is impossible.

V: \{\langle M, w \rangle \mid M \text{ accepts } w \text{ using } |w|^2 \text{ tape cells}\}
Solution: (A) DECIDABLE

We decide the language by building a 2-tape TM. On the first tape, we simulate \( M \) on \( w \). On the second tape, we keep a list of \( M \)'s configurations at each step of its computation. If \( M \) ever exceeds \(|w|^2\) tape cells, we reject. If \( M \) repeats a configuration before accepting (i.e. enters a loop), we reject. If \( M \) rejects, we reject. Otherwise, \( M \) accepts and we accept. We know one of these things must happen because there is only a finite number of configurations \( M \) can enter in \(|w|^2\) tape cells.
Problem 2

Let $\text{ACCEPTMORE} = \{ \langle M_1, M_2 \rangle \mid |L(M_1)|$ and $|L(M_2)|$ are finite, and $|L(M_1)| < |L(M_2)| \}$. Prove that $\text{ACCEPTMORE}$ is unrecognizable.

Solution: We reduce $\text{ACCEPTMORE}$ from $\overline{A_{TM}}$.

Define the reduction function $f$

$$f(\langle M, w \rangle) = \langle M_A, M_B \rangle$$

where $M_A$ and $M_B$ are defined as follows

\begin{align*}
M_A & \text{ on } x \\
1. \text{ If } x = 1, \text{ run } M & \text{ on } w \\
& \text{ if } M \text{ accepts, ACCEPT} \\
& \text{ if } M \text{ rejects, loop} \\
2. \text{ Else, REJECT}
\end{align*}

\begin{align*}
M_B & \text{ on } x \\
1. \text{ If } x = 1 & \text{ ACCEPT} \\
2. \text{ Else, REJECT}
\end{align*}

Since $M_A$ and $M_B$ are finite and each of their instructions is computable, $f$ is computable.

Suppose $\langle M, w \rangle \in \overline{A_{TM}}$. Then $M$ either rejects or loops on $w$. In either case, $M_A$ loops on ‘1’ and rejects all other inputs. So $|L(M_A)| = 0$. Since $M_B$ accepts only ‘1’, $|L(M_B)| = 1$. Since $0 < 1$, $\langle M_A, M_B \rangle \in L$.

Suppose $\langle M, w \rangle \notin \overline{A_{TM}}$. Then $\langle M, w \rangle \in A_{TM}$ and $M$ accepts on $w$. So $M_A$ accepts on ‘1’ and rejects all other inputs. So $|L(M_A)| = 1$. Since $M_B$ accepts only on ‘1’, $|L(M_B)| = 1$. Since $1 \notin L$, $\langle M_A, M_B \rangle \notin L$.

Therefore, $f$ is a reduction from $\overline{A_{TM}}$ to $\text{ACCEPTMORE}$, and $\text{ACCEPTMORE}$ is unrecognizable.
Problem 3

Prove that ACCEPTMORE is recognizable by a machine that has oracle access to $A_{TM}$.

Solution: We construct a recognizer with oracle access to $A_{TM}$ for ACCEPTMORE.

In order to do so, we first note that we can recognize the language

$$L = \{ \langle M, i \rangle \mid M \text{ accepts at least } i \text{ inputs} \}$$

Consider a machine that runs $M$ for $k$ steps on the first $k$ inputs. If the machine increments a counter each time $M$ accepts a new input, it can accept when the counter reaches $i$ and thereby recognize $L$. (We omit case analyses for this recognizer.) Let $R_L$ be such a machine.

Using $R_L$, we now construct a recognizer for ACCEPTMORE.

$R^{A_{TM}}$ on $\langle M_1, M_2 \rangle$

1. Let $b = 0$

2. For $j = 0$ to $\infty$

   Query the oracle to determine if $\langle R_L, \langle M_1, j \rangle \rangle \in A_{TM}$.

   (a) If no, query the oracle to determine if $\langle R_L, \langle M_2, j \rangle \rangle \in A_{TM}$

   if yes, let $b = 1$ and continue

   if no and $b = 0$, continue

   if no and $b = 1$, accept

   (b) If yes, continue

$R$ is finite and, given oracle access to $A_{TM}$ and the existence of $R_L$, each of its instructions is computable. So, $R$ specifies an oracle Turing machine.

Suppose $\langle M_1, M_2 \rangle \in$ ACCEPTMORE. Then there is some number $n$ such that $M_1$ accepts at most $n$ inputs, and $M_2$ accepts at least $n + 1$ inputs. Also, there is some $k$ such that $M_2$ accepts at most $k$ inputs. So, for all $c \leq n$, $R_L$ accepts on $\langle M_1, c \rangle$ and $\langle M_2, c + 1 \rangle$, but does not accept on $\langle M_1, n + 1 \rangle$. So, for $j = 0$ to $n$, the oracle responds ‘yes’ when asked if $\langle R_L, \langle M_1, j \rangle \rangle \in A_{TM}$. However, for $j = n + 1$, the oracle responds ‘no’. Since the oracle responds ‘yes’ when asked if $\langle R_L, \langle M_2, n+1 \rangle \rangle \in A_{TM}$ the flag $b$ is set to 1. Since $M_2$ accepts at most $k$ inputs, the oracle responds ‘no’ when asked if $\langle R_L, \langle M_2, k + 1 \rangle \rangle \in A_{TM}$. Since $b$ was set to ‘1’, $R^{A_{TM}}$ accepts $\langle M_1, M_2 \rangle$.

Suppose $\langle M_1, M_2 \rangle \not\in$ ACCEPTMORE. Then either (i) $M_1$ and $M_2$ accept the same number of inputs, or (ii) $M_1$ accepts more inputs, or (iii) $L(M_2)$ is infinite. In cases (i) and (ii), for any $n$, if $M_1$ does not accept $n$ inputs, then $M_2$ does not accept $n$ inputs. So, for all $j$, if $R_L$ does not accept input $\langle M_1, j \rangle$, then $R_L$ does not accept on input $\langle M_2, j \rangle$. So, if the oracle responds ‘no’ when asked if $\langle R_L, \langle M_1, j \rangle \rangle \in A_{TM}$, then the oracle responds ‘no’ when asked if $\langle R_L, \langle M_2, j \rangle \rangle \in A_{TM}$. So, $R$ never enters an accepting state, and so does not accept $\langle M_1, M_2 \rangle$. In case (iii), for any $j$ the oracle responds ‘yes’ when asked if $\langle R_L, \langle M_2, j \rangle \rangle \in A_{TM}$, so $R^{A_{TM}}$ does not accept $\langle M_1, M_2 \rangle$.

Therefore, $R^{A_{TM}}$ is a recognizer for ACCEPTMORE, and ACCEPTMORE is recognizable by a machine with oracle access to $A_{TM}$.

\footnote{The case where $L(M_1)$ is infinite is covered by cases (i) and (ii).}
Problem 4

Consider the following languages:

\[ L_1 = \{ \langle M, w \rangle \mid M \text{ is an oracle machine with access to } A_{TM} \text{ and } M \text{ does not accept } w \} \]

\[ L_2 = \{ \langle M_1, M_2 \rangle \mid M_1 \text{ and } M_2 \text{ are oracle machines with access to } A_{TM} \text{ and } L(M_1) = L(M_2) \} \]

Prove that if you can recognize \( L_2 \), you can recognize \( L_1 \). Further argue that if you can recognize \( L_1 \), you can recognize \( \overline{A_{TM}} \).

Solution: We first show that if you can recognize \( L_2 \), then that you can recognize \( L_1 \).

Suppose \( R_2 \) recognizes \( L_2 \). We define a many-one reduction from \( L_1 \) to \( L_2 \). For any \( x \in L_1 \), this allows us to run \( R_2 \) on \( f(x) \in L_2 \), thereby recognizing \( L_1 \).

Define the reduction function \( f \)

\[ f(\langle M, w \rangle) = \langle M_1, M_2 \rangle \]

where \( M_1 \) and \( M_2 \) are defined as follows

\[ M_1^{A_{TM}} \text{ on } x \]

1. Run \( M \) on \( w \) (simulating \( M \)'s oracle queries using \( M_1 \)'s oracle)

   if \( M \) accepts, ACCEPT

   if \( M \) rejects, loop

\[ M_2^{A_{TM}} \text{ on } x \]

1. REJECT

Since \( M_1 \) and \( M_2 \) are both finite and, given oracle access to \( A_{TM} \), each of their instructions is computable, \( f \) is a computable function.

Suppose \( \langle M, w \rangle \in L_1 \). Then \( M \) is an oracle TM that either rejects or loops on \( w \). In either case, \( M_1 \) loops on all inputs, and \( M_2 \) rejects all inputs. So \( \langle M_1, M_2 \rangle \in L_1 \).

Suppose \( \langle M, w \rangle \notin L_1 \). Then \( M \) is an oracle TM that accepts on \( w \). So \( M_1 \) accepts on all inputs and \( M_2 \) rejects all inputs. So, \( \langle M_1, M_2 \rangle \notin L_1 \).

Suppose \( L_1 \) is recognizable. Then some machine \( R_1 \) recognizes it. We use \( R_1 \) to construct a recognizer for \( \overline{A_{TM}} \).

\[ R_{\overline{A_{TM}}} \text{ on } \langle M, w \rangle \]

1. Run \( R_1 \) on \( \langle M, w \rangle \)

   if \( R_1 \) accepts, ACCEPT

   if \( R_1 \) rejects, loop

\[ M_{\langle M, w \rangle} \text{ on } x \]

1. Query the oracle: \( \langle M, w \rangle \in A_{TM} \)?

   if yes, ACCEPT

   if no, REJECT

Since \( R_{\overline{A_{TM}}} \) and \( M_{\langle M, w \rangle} \) are finite and, given \( M_{\langle M, w \rangle} \)'s oracle access to \( A_{TM} \), each of their instructions is computable, \( R_{\overline{A_{TM}}} \) specifies a Turing machine.

Suppose \( \langle M, w \rangle \in \overline{A_{TM}} \). Then \( M \) either rejects or loops on \( w \). Then the oracle responds ‘no’ in \( M_{\langle M, w \rangle} \). So \( M_{\langle M, w \rangle} \) rejects all inputs, including ‘1’. So, \( R_1 \) accepts on \( \langle M_{\langle M, w \rangle}, 1 \rangle \) and \( R_{\overline{A_{TM}}} \) accepts \( \langle M, w \rangle \).

Suppose \( \langle M, w \rangle \notin \overline{\langle M, w \rangle} \). Then \( \langle M, w \rangle \in A_{TM} \) and \( M \) accepts on \( w \). So the oracle responds ‘yes’ in \( M_{\langle M, w \rangle} \) and \( M_{\langle M, w \rangle} \) accepts on all inputs, including ‘1’. So \( R_1 \) either rejects or loops on \( \langle M_{\langle M, w \rangle}, 1 \rangle \). In either case, \( R_{\overline{A_{TM}}} \) loops and does not accept \( \langle M, w \rangle \).
Problem 5

Prove that the function $C(x) = \min |\langle M, w \rangle|$, where $M$ is a Turing machine which outputs $x$ when started on input $w$ is computable by a machine that has oracle access to $A_{TM}$.

Solution: We construct a machine that computes $C(x)$.

In order to do so, we first stipulate an ordering on machine-input pairs:

$$\langle M_{a_1}, w_{b_1} \rangle, \langle M_{a_2}, w_{b_2} \rangle, \langle M_{a_3}, w_{b_3} \rangle, \ldots$$

such that for any $i$, $|\langle M_{a_i}, w_{b_i} \rangle| \leq |\langle M_{a_{i+1}}, w_{b_{i+1}} \rangle|$. Given that such pairs are binary encodings, we can see that this ordering is generable by a TM.

Next, we identify a machine-input pair that we know satisfies the given condition. This gives us an upper bound on $C(x)$. $\langle N_x, \lambda \rangle$ serves as such a pair, where $\lambda$ is the empty string and $N_x$ is defined $N_x$ on input $z$.

1. Return $x$

For any given $x$, let $p(x)$ be the position of $\langle N_x, \lambda \rangle$ in the ordering of machine-input pairs. We can see that $p(x)$ is computable by a standard TM.

Next, we show that the language

$$L = \{ \langle \langle M, w \rangle, x \rangle \mid M \text{ outputs } x \text{ when run on input } w \}$$

is recognizable. For any $\langle \langle M, w \rangle, x \rangle$, a machine can simulate $M$ on $w$ and accept if $M$ halts with $x$ on the tape. (We omit case analysis for this recognizer.) Let $R_L$ be such a machine.

We are now ready to construct a machine that computes $C(x)$.

$M_C^{A_{TM}}$ on $x$

1. For $i = 0$ to $p(x)$, query the oracle: $\langle R_L, \langle \langle M_{a_i}, w_{b_i} \rangle, x \rangle \rangle \in A_{TM}$?

   - If yes, return $\langle M_{a_i}, w_{b_i} \rangle$
   - If no, continue

Since $M_C$ is finite and, given oracle access to $A_{TM}$, each of its instructions is computable, $M_C$ specifies an oracle Turing machine.

Suppose $C(x) = \langle M_{a_y}, w_{b_y} \rangle$. Then $\langle M_{a_y}, w_{b_y} \rangle$ is the first pair in the ordering such that $M_{a_y}$ accepts on $w_{b_y}$ and outputs $x$. So, $y \leq p(x)$ and for all $i < y$, $M_{a_i}$ fails to output $x$ when run on $w_{b_i}$. So, $M_C$’s oracle responds ‘no’ when queried. However, the oracle responds ‘yes’ to $\langle R_L, \langle \langle M_{a_y}, w_{b_y} \rangle, x \rangle \rangle$, so $M_C$ returns $\langle M_{a_y}, w_{b_y} \rangle$.

Suppose $M_C$ on $x$ returns $\langle M_{a_y}, w_{b_y} \rangle$. Then $M_{a_y}$ outputs $x$ when run on $w_{b_y}$ and, by the reasoning above, this is true for no previous pair in the machine-pair ordering. So $C(x) = \langle M_{a_y}, w_{b_y} \rangle$. 

6