Announcement
Prelim 2 is graded now. Solution Set uploaded to CMS soon.

Distribution (max 40)

<table>
<thead>
<tr>
<th>Grade</th>
<th>Tue 4/13</th>
<th>Thurs 4/15</th>
</tr>
</thead>
<tbody>
<tr>
<td>80%</td>
<td>38.0</td>
<td>33.8</td>
</tr>
<tr>
<td>60%</td>
<td>35.7</td>
<td>30.3</td>
</tr>
<tr>
<td>40%</td>
<td>32.8</td>
<td>27.1</td>
</tr>
<tr>
<td>20%</td>
<td>25.5</td>
<td>22.5</td>
</tr>
</tbody>
</table>

Recap
For a Turing machine \(M\),

\[ L(M) = \{ x \mid M \text{ accepts } x \} \]

"The language of \(M\)"

Def.
1. A set of strings, \(L\), is called r.e. if \(\exists M\) such that \(L = L(M)\).
2. \(L\) is recursive if \(\exists M\) that always halts on every input s.t. \(L = L(M)\).
Important Facts, (From last Wed)

(A) \( L \) is recursive if and only if \( L \) and its complement are both r.e.

\((\Rightarrow)\) easy
\[(\Leftarrow)\] If \( L = L(M_1) \) and \( \overline{L} = L(M_0) \), then a machine \( M \) that decides \( L \) is defined by running \( M_0 \) and \( M_1 \) in parallel on two different tapes and accepting \( x \) if \( M_1 \) accepts it and rejecting \( x \) if \( M_0 \) accepts it.

(B) Let \( HP \) (for “halting problem”) denote \( HP = \{ x \# y : x \text{ describes a Turing machine that halts on input } y \} \).

Then \( HP \) is r.e. but not recursive.

(C) The complement of \( HP \), denoted \( coHP \), is neither r.e. nor recursive.

"\( coHP \) is not r.e." means "There is no algorithm to test that a machine is not going to"
halt on a specified input, even if we allow the algorithm to run for an unlimited length of time. Algorithms for this problem are guaranteed to either:

1. Not halt on some input $x \# y$ even though the answer is, "Yes, $M_x$ runs forever on input $y$.

2. They halt and output an answer on $x \# y$ but it's the wrong answer.

Today, using reductions, to show other problems are "undecidable" (not recursive).

**Theme.** Almost every task in program analysis (predicting the behavior of a program, given its code) is undecidable.

**Example.** Given $x \# y \# z$ where

- $x$ is a description of a TM
- $y$ is a description of its input
$z$ is a description of a special "forbidden symbol" 

$(z = 0^f1$ where $1 \leq f \leq |M|)$. When machine $M_x$ executes on input $y$, does it ever write $z$ on its working tape?

$$FSP = \{ x \# y \# z \mid M_x \text{ writes } z \text{ when processing } y \}$$

To prove $FSP$ is undecidable, we'll come up with a reduction $HP \leq FSP$.

That means a function $R$ mapping $a \# b \mapsto x \# y \# z = R(a\#b)$ such that

1. $R$ is computable by a Turing machine.
2. $a \# b \in HP \iff R(a\#b) \in FSP$.

This will show $FSP$ is not decidable because if $FSP = L(K)$ for some $K$ that halts on all inputs, then the following $M$ halts on all inputs and satisfies $HP = L(M)$. 
Pseudocode for $M$:

1. On input $a\#b$, compute $R(a\#b)$.
2. Run machine $K$ on $R(a\#b)$.

Since there is no $M$ that solves $HP$,
there is no $K$ that solves $ESP$.

Missing piece: what is $R$?

$R$ takes $a\#b$ and modifies a
(description of Turing machine $M_a$)
as follows:

- Working alphabet size increases
  from $m$ to $m+1$. (Extra symbol will be “forbidden symbol.”)
- Transition rule modified so that any transitions that enter
  the halting states, $t$ or $r$, also write symbol $m+1$ on the tape.
- Add “dummy rules” to transition rule that say if you read
  symbol $m+1$ in any state $q$,
  remain in state $q$, write symbol $m+1$,
  don’t move anywhere on the tape.

These instructions
are not important
for correctness.
Let $x$ be the string describing this new TM.

$$R(a \# b) = x \# b \neq 0 \# 1$$

new TM same input symbol
as described above

Why does this work?

1. IF $a\#b$ is a "yes" instance of HP then $M_a$ halts on input $b$,
   so $M_x$ halts on input $b$ and writes symbol $m+1$ as it is
   halting
   so $x \# b \neq 0 \# 1 \in FSP$

2. IF $R(a \# b) \in FSP$ then $M_x$ writes $m+1$ when processing $b$.
   The first time $M_x$ writes $m+1$
   on its tape, it is not yet reading $m+1$. ($m+1$ is not among the input symbols)
   That means $M_a$ halts at that moment when processing $b$.

$\Rightarrow a\#b \in HP$. 
Showing a set is not even r.e.

by reducing from \( \text{coHP} \).

**Example.** \( \text{INF} = \{ x \mid x \text{ is a description of a Turing machine } M \text{ and } L(M) \text{ is infinite.} \} \)

Prove \( \text{INF} \) is not r.e.

Reduction from \( \text{coHP} \) to \( \text{INF} \).

Again, same rules:

1. \( R(a\#b) \) can be computed by a TM.
2. \( a\#b \in \text{coHP} \iff R(a\#b) \in \text{INF} \)

Given \( a\#b \) let \( M \) be a TM that does the following.

1. Write \( a\#b \) on Tape 2.
   (Tape 1 is its input tape.)

2. Use universal TM to
simulate on Tape 2, the execution of a processing b.

3. At the same time, in every transition it moves right on Tape 1.

4. If it ever reaches blank space on Tape 1, it accepts its input.

5. If the simulation of $M_a$ halts before blank space reached on Tape 1, rejects the input.

This $M_a$ accepts input $y$ iff

length($y$) ≤ running time of $M_a$ on $b$.

If $M_a$ runs forever on $b$ infinitely many inputs $y$ satisfy this property.

If $M_a$ halts on $b$, only finitely
many y will be accepted.