This test is **online**, posted via Canvas and the course homepage, and handed in via GradeScope.

The GradeScope submission page will close at 2:30pm (Eastern Time), but we suggest that you start wrapping up by around 2:10pm.

You have 170 minutes to earn up to \(1 + 6 + 4 + 5 + 5 + 5 = 26\) points, the test is graded out of 25.

This booklet contains 5 **questions on 6 pages**, including this one.

Write your solutions in the space provided. If you run out of space, continue your answer on the back of the last page, and make a notation on the front of the sheet.

You may use any written or locally stored resources.

However, the only internet resources that you could use during the test are: specifically:

1. The BlueJeans Office Hours link at [https://gatech.bluejeans.com/242024735](https://gatech.bluejeans.com/242024735), where clarifications will be posted in the chat.

2. Piazza Test 4 clarification page: [https://piazza.com/class/k4xbfrttfnc687?cid=661](https://piazza.com/class/k4xbfrttfnc687?cid=661)

You may use any of the theorems/facts/lemmas from the lecture notes, homeworks, or textbook without re-proving them unless explicitly stated otherwise. However, we may check with you about the citation / reference afterwards, possibly via regrades.

If necessary make reasonable assumptions but please be sure to state them clearly

Do not spend too much time on any one problem. Generally, a problem’s point value is a good indication of how many minutes to spend on it.

Good luck!

0. (1 point) Submit your test through GradeScope before 2:30pm, under the right name/id, and with the pages corresponding to each problem clearly indicated.
1. (6 points, 3 each) Provide solutions to the following two instances of the post correspondence problem. Make sure to specify your answer as the sequence of indices.

Recall that the post correspondence problem is given strings \(a_1 \ldots a_n\) and \(b_1 \ldots b_n\), find a sequence of (possibly repeating) indices \(i_1 \ldots i_m\) such that

\[a_{i_1}a_{i_2} \ldots a_{i_m} = b_{i_1}b_{i_2} \ldots b_{i_m}.\]

Here the strings are provided as \(\frac{a_i}{b_i}\).

(a)

\[
\begin{align*}
1: & \quad \frac{baaa}{bb}, & 2: & \quad \frac{ab}{a}, & 3: & \quad \frac{a}{aa}.
\end{align*}
\]

**SOLUTION:** 2, 1, 3, 3, 3.

(b)

\[
\begin{align*}
1: & \quad \frac{000000}{0}, & 2: & \quad \frac{0}{00000}.
\end{align*}
\]

**SOLUTION:** 1, 1, 1, 1, 2, 2, 2, 2, 2, 2.

**GRADING:** −1 point per each edit distance to a correct solution.
2. (4 points) Show that the following language is undecidable:

\[ A = \{ M \mid M \text{ is a Turing machine and } M \text{ halts on all inputs} \} . \]

You may reduce from any undecidable problem that we discussed, or in the textbook.

**SOLUTION:** We reduce the halting problem to deciding \( A \).

For any TM/input pair \( < M, w > \) that we want to decide whether it halts, we convert to a machine \( M' \) that on any input, simulates running \( M \) on \( w \). If \( M \) terminates, \( M' \) will accept (regardless of its input). Otherwise \( M' \) will not terminate.

Then \( M' \in A \) if and only if \( M \) terminates. As \( \text{Halt} \) is undecidable, \( A \) is undecidable as well.

**GRADING:**

1 point for reducing from some undecidable problem.

1 point for creating a \( M' \) to check whether \( M' \in A \).

.5 point for mapping the \( M' \in A \) case correctly: that is, properly creating a case where \( M' \) halts all strings.

.5 point for describing the case where \( M' \) doesn’t halt for some strings.

1 point on how to use whether \( M' \in A \) to answer the original input to the undecidable problem.
3. ( /5 points) Consider the language over the binary alphabet \( \Sigma = \{0, 1\} \):

\[ L = \{ w \in \{0, 1\}^* | \text{every prefix of } w \text{ has more 0s than 1s} \} \]

Use pumping lemma to show that \( L \) is not regular.
Note that ‘more’ here means strictly more, and does not include the equality case.

**SOLUTION:** We prove by contradiction. Take \( w = 0^p1^{p-1} \in L \), for given pumping length \( p \). Now, \(|w| \geq p\) and for all splits \( w = xyz \) with \(|xy| \leq p\), \( y \) must contain all zeros. But string \( xy^0z \not\in L \), as now the full string will contain at least as many 0s as 1s. So, by pumping lemma the given language is not regular.

**GRADING:**

(a) 1 point - Proving that the language is not regular by proof of contradiction (contradicting the assumption made that the language is regular)
(b) 2 points - Considering all valid splits of a string \( s \) in \( L \)
(c) 2 points - Showing that there exists an \( i \) such that \( xy^iz \not\in L \)
4.  (  /5 points)

Consider the PDA below with alphabet $\Sigma = \{0,1\}$ and stack symbols $\Gamma = \{x_0, x_1, \$\}$.

Exhibit a sequence of states / stack states by which this PDA accepts the input string

0110.

Make sure to indicate the direction of the stack in the stack states.

**SOLUTION:**  Stack is denoted with bottom of the stack on the right.

<table>
<thead>
<tr>
<th>state</th>
<th>stack</th>
<th>symbol consumed</th>
</tr>
</thead>
<tbody>
<tr>
<td>$q_0$</td>
<td>$\emptyset$</td>
<td>$\epsilon$</td>
</tr>
<tr>
<td>$q_1$</td>
<td>$$ $</td>
<td>0</td>
</tr>
<tr>
<td>$q_1$</td>
<td>$x_0 $</td>
<td>1</td>
</tr>
<tr>
<td>$q_1$</td>
<td>$x_1 x_0 $</td>
<td>$\epsilon$</td>
</tr>
<tr>
<td>$q_2$</td>
<td>$x_1 x_0 $</td>
<td>1</td>
</tr>
<tr>
<td>$q_2$</td>
<td>$x_0 $</td>
<td>0</td>
</tr>
<tr>
<td>$q_2$</td>
<td>$$</td>
<td>$\epsilon$</td>
</tr>
<tr>
<td>$q_3$</td>
<td>$\emptyset$</td>
<td></td>
</tr>
</tbody>
</table>

**GRADING:**

(a) -1 for not indicating the direction of the stack in the stack states

(b) -1 for missing the empty stack in the beginning or end for the PDA

(c) -1 for each mistake in stack symbol or state

(d) -2 for incorrect stack symbols
5. (5 points)

Describe a deterministic Turing machine that takes as input (encoded as binary):

(a) a context free grammar $G$ (given as variables, terminals/alphabet, rules, starting variable),
(b) a string $s$
(c) a positive integer $k$

and decides whether one can modify at most $k$ characters from $s$ to create a string that’s in $L(G)$ (the set of strings that can be generated from the grammar $G$).

You may use any modifications to Turing machines discussed in the text book, or in class. However, you need to be specific about how you manipulate the given CFG grammar. Furthermore, note that the problem is asking for a deterministic Turing machine.

SOLUTION: Here we design a TM $S$ similar with the one of Theorem 4.7 in textbook.

$S =$ “On input $\langle G, s, k \rangle$, where $G$ is a CFG, $S$ is a string and $k$ is a positive integer.

1. Convert $G$ to an equivalent CFG in Chomsky norm form.

2. List all derivations with $2n - 1$ steps, where $n$ is the length of $s$, except if $n = 0$, then instead list all derivations with 1 step.

3. List all strings whose editing distance with $s$ is at most $k$.

4. Check whether the above two lists intersect or not. If no intersection, return false, otherwise return true.

GRADING:

-1 if the TM isn’t deterministic.

-2 the TM doesn’t halt, e.g. if you try to simulate the PDA obtained from the CFG.

-1 Missing conversion to CNF.

-1 for not listing all possible strings whose editing distance with $s$ is at most $k$.