This assignment is **due in Week 7** and should be submitted to Gradescope.

All work must be **done individually** without consulting anyone else's solutions in accordance with the University's "Academic Dishonesty and Plagiarism" policies.

Go to the last page of this document and read the **Submission Instructions**. For clarifications and updates, monitor "Assignment FAQ".

Problem 1. (25 marks)

Consider the following grammar G over alphabet $\Sigma = \{a, b\}$ with start variable S:

$$S \to TaU$$
$$T \to VTa \mid a$$
$$V \to a \mid b$$
$$U \to aU \mid b$$

1. Provide a clear and concise description of the language of strings that can be derived from U (2 marks) and from T (3 marks).
2. State five strings that are in $L(G)$, and five that are not. The strings should be over Σ.
3. Show that G is ambiguous by providing two distinct leftmost derivations of the same string.
4. Provide a grammar in Chomsky Normal Form that is equivalent to G.
5. Provide a CYK table for the string $abbaa$ and the following grammar:

$$S \to TU$$
$$T \to VT \mid a$$
$$U \to a \mid b$$
$$V \to UU$$

Problem 2. (25 marks)

For each of the following languages over $\Sigma = \{a, b\}$, provide a context-free grammar for that language.

1. The set of strings of the form ua where u is a palindrome.
2. The set of strings that contain a length 6 palindrome as a substring.
3. The set of strings of the form $a^i b a^j b a^k$ where $j = 2i + 3k + 1$ $(i, j, k \in \mathbb{Z}_0^+)$

4. The set of strings whose length is a multiple of 3, and do not contain an a in the final third.
5. The set of palindromes that do not contain baa as a substring.

Problem 3. (10 marks + 10 bonus marks)

Your adventures as a secret agent continue. Last time, the DFA told you to keep going, so you're still stuck behind enemy lines, trying to receive messages over the alphabet $\Sigma = \{a, b\}$. Your home base has come up with a plan: to prevent enemy action from altering the message, they'll send each message to you twice, and they'll provide you with a DFA to recognise the language $\{xx : x \in \Sigma^*\}$, so you can verify the message has not been tampered with.

Unfortunately, this plan has run into a snag. You remember from your advanced training in models of computation that this language is not regular, so no DFA will be able to recognise it.

Your country believes it has a solution: instead of sending the second message unaltered, it will apply a function f to it. The function f will be cleverly chosen to be a *length-preserving injection* $f : \Sigma^* \to \Sigma^*$. "Length-preserving" means that $len(f(x)) = len(x)$ for all strings x, i.e. applying f to any string always produces a string of the same length. An injection is a function such that for all x, y in the domain with $x \neq y$, we have $f(x) \neq f(y)$, i.e. different inputs map to different outputs.

Hence, researchers are trying to find a length-preserving injection f such that the language $\{xf(x)|x \in \Sigma^*\}$ will be regular — here $xf(x)$ is the concatenation of strings x and $f(x)$. But is this possible?

For instance, the function $f(x) = x$ is a length-preserving injection. We know that $\{xx : x \in \Sigma^*\}$ is not regular, so this particular f won't do. Another example of a length-preserving injection is $f(x) = reverse(x)$. In this case, $\{xf(x) : x \in \Sigma^*\}$ is the set of even-length palindromes, which is also not regular.

1. (5 marks) Let $f : \Sigma^* \to \Sigma^*$ be the function that turns every a into b and vice versa. For example, $f(abbabbaa) = baabaabb$. Prove that the language $\{xf(x) : x \in \Sigma^*\}$ is not regular.
2. (5 marks) Prove that for *every* length-preserving injection $f : \Sigma^* \to \Sigma^*$, the language $\{xf(x) : x \in \Sigma^*\}$ is not regular.
3. (Bonus part, 10 bonus marks, to be submitted in a separate pdf) Prove or disprove the following claim: For every length-preserving function $f : \Sigma^* \to \Sigma^*$, the language $\{xf(x) : x \in \Sigma^*\}$ is not regular. (Note that in the bonus part, f is not necessarily an injection.)

Notes:

- Use the methods taught in this course, based around analysing runs of automata. Do not use the pumping lemma.

- For the first part, it will be sufficient to use the technique given on slide 7 of the non-regularity lecture. For the second and bonus part, you will likely need to vary the technique a little. See previous slides.
- Be very careful when arguing that a string is not in a language. Ensure that this is actually the case.
- We believe that this bonus part is quite difficult, and recommend only attempting it after fully completing the rest of the assignment. Keep in mind the previous note.

Problem 4. (10 marks)

You have a context-free grammar G in Chomsky Normal Form, and you treasure it dearly. Unfortunately, context-sensitive goblins recently broke into your house and tampered with your grammar. They added various rules of the form $AB \to c$, where A, B are variables and c is a terminal. Of course, these rules mean that your new grammar H is no longer context-free. Since you love context-free grammars, you want to produce a grammar without these rules, while preserving the language of H.

Given a grammar H, whose rules are of the form

$A \to BC$

$A \to a$

$AB \to c$

where capital letters are variables and lowercase letters are terminals (in particular, $S \to \epsilon$ is not a rule), show how to produce an equivalent context-free grammar.

Partial marks will be awarded for the case where there is only a single rule of the form $AB \to c$.

Submission Instructions

You will submit answers to all the problems on Gradescope.

Problems 1 and 2 (except for 1.1) are autograded. It is essential that you ensure that your submission is formatted so that the autograder can understand it. Upon submitting your responses, you should **wait** for the autograder to provide feedback on whether your submission format was correct. **An incorrectly formatted submission for a question will receive zero marks for that question.** A scaffold will be provided on Ed with the file names the autograder expects.

Problem 1.1 format: Submit a text file named

`problem_1_1.txt`

This file will be manually marked.

Problem 1.2 format:

The first line of each answer should contain a comma separated sequence of five strings that are in the language, and the second line should contain a comma separated sequence of five strings that are not in the language. For example, if the CFG is $S \to bS \mid \epsilon$, an example of a correct text file would be:

```
epsilon, b, bb, bbb, bbbb
a, aa, aaa, aaaa, aaaaa
```

Problem 1.3 format:

The file should contain two lines, one for each derivation. Each derivation should contain sequences of variables and terminals separated by $=>$. Each step (one $=>$) should use exactly one rule once (ie. do not skip steps). For example, if the grammar was $S \to aa \mid T, T \to aT \mid \epsilon$, a correct answer would be:

```
S => aa
S => T => aT => aaT => aa
```

Problem 1.5 format:

This format is intended to replicate the CYK diagrams we have seen in this course.

The file should contain five lines. The n-th line should contain n entries separated by $|$. Each entry should contain the sequence of variables in that cell of the table. Commas may optionally be used within a cell to separate variables, and a lowercase x may be used to indicate that a cell is empty. The parser will ignore commas, lowercase x, and spaces — they are purely for your own readability.

For example, for the string aba and grammar $S \to AB \mid a, A \to a \mid AA, B \to b \mid BB$, one possible answer would be:

```
        x
    S   |  x
S,A |   B  |  S,A
```

which can also be written

```
x
S|x
SA|B|SA
```

Problem 1.4, Problem 2 format, CFGs:

```
V = S X
T = a b
start = S
S -> bX
X -> XSa | epsilon
```

1. The first line is the set of variables (V), as a space-separated list
2. The second line is the set of terminals (T), as a space-separated list
3. The third line indicates the start variable of the grammar
4. The fourth line is a rule, that S can produce Xa
5. The remaining lines are all rules. If a variable has multiple rules, they can be written on one line separated by | as shown, or on separate lines.

Problem 3,4 format:

Problems 3 and 4 are hand graded. You will submit **typed pdfs (no pdf containing text as images, no handwriting)**: one for problem 3, one for problem 4, and one for the problem 3 bonus (if attempted). Start by typing your student ID at the top of the first page of each pdf. Do **not** type your name. Do not include a cover page. Submit only your answers to the questions. Do **not** copy the questions. Your pdf must be readable by Turnitin.