This assignment is **due on Sunday Nov 22, 23:59** and has a coding part and a written part.

- Submit your written part (Problems 1 and 2) as a single pdf on Gradescope.
- Submit your code (Problem 3) on Ed.
- All work must be *done individually* without consulting anyone else's solutions in accordance with the University's "Academic Dishonesty and Plagiarism" policies.
- For clarifications and more details on all aspects of this assignment (e.g., level of justification expected, late penalties, repeated submissions, what to do if you are stuck, etc.) you are expected to regularly monitor the Ed Forum post "Assignment Guidelines".

**Problem 1.** Let $\Sigma = \{a, b\}$ and consider the language $L_1 \subseteq \Sigma^*$ of all strings of the form $a^m u$ where $m \geq 0$ and $u$ has at most $m$ many $a$s in it. Thus, e.g., *aababa* and *aabba* and *aa* are in $L_1$, but *aababaabb* is not in $L_1$.

1. Give a context-free gra age $L_1$
2. Explain why your gram
3. Explain whether or not $L_1$ is regular



der

**Problem 2.** In this problem you will design a context-free grammar that generates the language associ $\{p, m\}$. We say that a string t to right, and interpreting $p$ as ning sum is non-negativ lid counter strings, but *pmmp* and *mpp* are not valid counter string.



1. Give a context-free grammar $G_2 = (V, \Sigma, \ldots$ guage $L_2$ of valid counter strings.
2. Explain why your grammar is correct.
3. Explain whether or not $L_2$ is regular.

**Problem 3.** In this problem you will implement the CYK algorithm and extend it to provide rightmost-derivations and detect ambiguous strings. Your algorithm will have three modes: membership, rightmost-derivation, and ambiguous. Input is read from standard input (stdin). The first line of input indicates which mode is to be executed. The remaining input is the data to act on. Examples of usage, and of input and output are provided in Appendix B.

1. **Membership mode**
   - Input: `membership` followed by a context-free grammar in Chomsky normal-form followed by a sequence of input strings.
   - Output: One line per input string, giving the string, a comma, and then `1` if the string is generated by the grammar and `0` otherwise.

2. **Rightmost-derivation mode**

   - Input: `rightmost-derivation` followed by a context-free grammar in Chomsky normal-form followed by an input string that is generated by the context-free grammar.
   - Output: A sequence of lines, each line containing the next step of a *rightmost* derivation of the input string.

3. **Ambiguous mode**

   - Input: `ambiguous` followed by a context-free grammar in Chomsky normal-form followed by a sequence of input strings.
   - Output: One line per input string, giving the string, a comma, and then `1` if the string is ambiguous and `0` otherwise.



# A Marking

**Problem 1 (25 marks)**



- The first part is for 10 marks. For full credit your context-free grammar should not be more complicated than needed.



- The second part is for 5 marks. For full credit your answer should be clear, and explain why your grammar o $L_1$.



- The third part is for NFA or RE for the l , res that you provide a proof using the PHP as in lectures.

**Problem 2 (25 marks)**



- The first part is for 10 marks. For full credit your context-free gra complicated than needed.
- The second part is for 5 marks. For full credit your answer should be clear, and should explain why your grammar only generates strings in $L_2$ and why it generates all strings in $L_2$.
- The third part is for 10 marks. An explanation of regularity requires that you provide a DFA, NFA or RE for the language and prove it is correct. An explanation of non-regularity requires that you provide a proof using the PHP as in lectures.

**Problem 3 (50 marks)**

- For passing the test-cases:
  - 20 marks for membership mode
  - 10 marks for rightmost derivation mode
  - 10 marks for ambiguous mode.

  You are not allowed to hard-code any of the test-cases.
- The remaining 10 marks are based on the overall quality of the code. Quality is judged based on our answers to questions such as:
  - How readable is the code? e.g. commenting, variable naming, space, line length, etc.

- Are the data structures appropriate to the problem? e.g. how the grammars are represented.
- Are the algorithms implemented in a reasonably efficient way?
- How much of the assignment was attempted.

# B Input/Output formats

## Input context-free grammar in Chomsky normal-form

A sequence of lines:

1. A comma separated list of variable symbols
2. A comma separated list of terminal symbols
3. The start variable
4. One or more lines of the form:
   - `A -> B C`
   - `A -> a`
   - `A -> epsilon`
5. the string `end`

For example:

```
A,B,C,D,S,T
a,b
T
T -> A B
T -> B A
T -> S S
T -> A C
T -> B D
T -> epsilon
S -> A B
S -> B A
S -> S S
S -> A C
S -> B D
C -> S B
D -> S A
A -> a
B -> b
end
```

... represents the grammar:

$$\mathbf{T} \to \mathbf{AB} \mid \mathbf{BA} \mid \mathbf{SS} \mid \mathbf{AC} \mid \mathbf{BD} \mid \epsilon$$
$$\mathbf{S} \to \mathbf{AB} \mid \mathbf{BA} \mid \mathbf{SS} \mid \mathbf{AC} \mid \mathbf{BD}$$
$$\mathbf{C} \to \mathbf{SB}$$
$$\mathbf{D} \to \mathbf{SA}$$
$$\mathbf{A} \to a$$
$$\mathbf{B} \to b$$

### Membership mode

1. The input is a sequence of lines:
   (a) `membership`
   (b) A context-free grammar in Chomsky normal-form (in the format described above)
   (c) Several strings to check, one on each line, followed by `end`
2. The output is `1` or `0` for each input string, where `1` means the string is generated by the grammar and `0` means it is not, all followed by `end`.

Here is an example. If the input is:

```
membership
A,B,C,D,S,T
a,b
T
T -> A B
T -> B A
T -> S S
T -> A C
T -> B D
T -> epsilon
S -> A B
S -> B A
S -> S S
S -> A C
S -> B D
C -> S B
D -> S A
A -> a
B -> b
end
aab
abababab
end
```



the output is:

```
0
1
end
```

### Rightmost-derivation mode

1. The input is a sequence of lines:
   (a) `rightmost-derivation`
   (b) A context-free grammar in Chomsky normal-form (in the format described above)
   (c) A single non-empty string to derive, followed by `end`
2. The output is a sequence of lines:
   (a) Each line should be a non-empty string over the alphabet of variables and terminals
   (b) The sequence of lines should represent a *rightmost* derivation in which each line yields the next
   (c) The last line should be `end`

Here is an example. If the input is:

```
rightmost-derivation
A,B,C,D,S,T
a,b
T
T -> A B
T -> B A
T -> S S
T -> A C
T -> B D
T -> epsilon
S -> A B
S -> B A
S -> S S
S -> A C
S -> B D
C -> S B
D -> S A
A -> a
B -> b
end
ab
end
```



a possible output is:



```
T
AB
Ab
ab
end
```



which encodes the rightmost derivation



$$T \Rightarrow AB \Rightarrow Ab$$

of the string *ab*.

### Ambiguous mode

1. The input is a sequence of lines:
   (a) `ambiguous`
   (b) A context-free grammar in Chomsky normal-form (in the format described above)
   (c) Several non-empty strings to check, one on each line, followed by `end`
2. The output is a sequence of lines:
   (a) The output is `1` or `0` for each input string, where a `1` means the string is ambiguous, and a `0` that it is not

Here is an example. If the input is:

```
ambiguous
A,B,C,D,S,T
a,b
T
T -> A B
T -> B A
```

```
T -> S S
T -> A C
T -> B D
T -> epsilon
S -> A B
S -> B A
S -> S S
S -> A C
S -> B D
C -> S B
D -> S A
A -> a
B -> b
end
ab
a
abab
end
```

then the expected output is:

```
0
0
1
end
```

because *ab* has one rightmost derivation, *a* has no rightmost derivations, and *abab* has two rightmost derivations.