Problem 0:

i. Consider the CFG with {S,A,B) as the non-terminal alphabet, {a,b) as the terminal alphabet, S as the start symbol and the following set of production rules

S --> aB | bA

B --> b | bS | aBB

A --> a | aS | bAA

Which of the following strings is generated by the grammar?
(A) aaaabb **(B)** aabbbb **(C)** aabbab **(D)** abbbba

ii. Consider a CFG
E → E + T | T
T → T x F | F
F → (E) | a
Show how following strings are derived from the CFG.
(a) a (b) a+a (c) a+a+a (d) ((a))

iii. Give context-free grammars that generate the following languages. In all parts, the alphabet Σ is {0,1}.
(a) {w∈ Σ∗ | w contains at least three 1s}
(b) {w∈ Σ∗ | w starts and ends with the same symbol}
(c) {w∈ Σ∗ | the length of w is odd}
(d) {w∈ Σ∗ | the length of w is odd and its middle symbol is a 0}
(f) The empty set

Problem 1: Designing CFGs For each of the following languages, design a CFG for that language. T

i. Given Σ = {a, b, c}, write a CFG for the language L= {w ∈ Σ∗ | w contains **aa** as a substring}. For example, the strings aa, baac, and ccaabb are all in the language, but aba is not.

ii. In our lecture on regular expressions, we wrote the following regular expression that matched email addresses:

$$a^+(.a^+)^*@a^+(.a^+)^+$$

Given Σ = {@, ., a}, write a CFG whose language is the same as the language of this regular expression.

iii. Given Σ = {a, b}, write a CFG for the language L= {w ∈ Σ ⋆ | w is not a palindrome}, the language of strings that are not the same when read forwards and backwards. For example, aab ∈ L and baabab ∈ L, but aba ∉ L, bb ∉ L, and ε ∉ L.

Problem 2: The Complexity of Addition This problem explores the following question: ***How hard is it to add two numbers?***

Suppose that we want to check whether x+y=z where x, y, and z are all natural numbers. If we want to phrase this as a problem as a question of strings and languages, we will need to find some way to standardize our notation. In this problem, we will be using the ***unary number system***, a number system in which the number is represented by writing out n 1's. For example, the number 5 would be written as 11111, the number 7 as 1111111, the number 12 as 111111111111, and the number 0 as ε.

Given the alphabet Σ = {1, +, =}, we can consider strings encoding x+y=z by writing out x, y, and z in unary. For example:

4 + 3 = 7 would be encoded as 1111 + 111 = 1111111

7 + 1 = 8 would be encoded as 1111111 + 1 = 11111111

0 + 1 = 1 would be encoded as +1 = 1

Consider the following language, which we'll call ADD:

$\{1^m+1^n= \ 1^{m+n} \mid m, n \in N\}$

For example, the strings 111+1=1111 and +1=1 are in the language, but 1+11=11 is not, nor is the string 1+1+1=111.

i. Prove or disprove: the language defined above is regular.
Hint: To prove a language is regular, you could, for example, build a DFA, an NFA, or a regex for it. To prove a language is not regular, use the Myhill-Nerode theorem. You'll need to decide which case is applicable here.

ii. Write a context-free grammar for ADD, showing that ADD is context-free.
Hint: You may find it easier to solve this problem if you first build a CFG for this language where you're allowed to have as many numbers added together as you'd like. Once you have that working, think about how you'd modify it so that you have exactly two numbers added together on the left-hand side of the equation.

Problem 3:

Let's begin with a new definition. If L is a language over Σ and x ∈ Σ*, then the ***derivative of L with respect to* Σ,** denoted ∂xL, is the language defined as follows:

∂xL = {w ∈ Σ* | xw ∈ L}.

i. Let Σ = {a, b} and let L = {ε, aba, bab, aaab, bbbaaa}. Answer each of the following questions; no justification is necessary.

1. What is the longest string in ∂aL? If two or more strings are tied for having the longest length, list any one of them.

2. What is the longest string x ∈ Σ* for which |∂xL| = 1? If two or more such strings are tied for having the longest length, list any one of them.

3. What is the shortest string for x ∈ Σ* for which |∂xL| = 0? If two or more such strings are tied for having the shortest length, list any one of them.

ii. Let L = $\{a^n b^n \mid n \in N\}$. Write a CFG for ∂aaaL.

iii. Prove Brzozowski's Theorem.

Brzozowski's Theorem: Let L be a language over Σ and let S ⊆ Σ* be a set containing infinitely many strings. If the following statement is true, then L is not regular:

∀x ∈ S. ∀y ∈ S. (x ≠ y → ∂xL ≠ ∂yL)

Problem 4: Programming Turing Machines

The Church-Turing thesis might seem surprising in light of just how simple Turing machines are. Can they really do any computation that could be done by any feasible computing system? That is indeed the case, and one of the best ways to see this is to try your hand at designing some TMs to see how moving around and marking up a tape is all that's needed to perform arbitrary computations.

i. Let Σ = {a, b}. Program a TM that's a decider for { w ∈ Σ* | w has odd length and its middle character is a }.

ii. Let Σ = {a}. Program a TM that's a decider for $\{a^{2^n} \mid n \in N\}$. That is, your TM should accept all strings of a's whose lengths are powers of two and reject all other strings.

iii. Let Σ = {a, b, =}. Program a TM that's a decider for the language L = {w=w | w ∈ {a, b}*}. For example, the strings aaa=aaa, bab=bab, b=b, and = are all in , while the strings aaa=aab, =aaa, aaa=, and a=a=a are all not in L.

Problem 6: (Optional) What Does it Mean to Solve a Problem?

Let L be a language over Σ and M be a TM with input alphabet Σ. Here are three potential traits of M:

1. ∀w ∈ Σ∗. M halts on w.
2. ∀w ∈ Σ∗. (M accepts w → w ∈ L).
3. ∀w ∈ Σ∗. (M rejects w → w ∉ L).

At some level, for a TM to claim to solve a problem, it should have at least some of these properties. Interestingly, though, just having two of these properties doesn't say much.

i. Prove that if L is any language over Σ, then there is a TM that satisfies properties (1) and (2). Note that this is an existential statement, so to prove it, you will need to give a concrete example of a TM with these properties, or at least show how to construct one. The whole point of this problem is to show that you have to be extremely careful about how you define "solving a problem," since if you define it incorrectly then you can "solve" a problem in a way that bears little resemblance to what we'd think of as solving a problem. Keep this in mind as you work through this one.

ii. Prove that if L is any language over Σ, then there is a TM that satisfies properties (1) and (3).

iii. Prove that if L is any language over Σ, then there is a TM that satisfies properties (2) and (3).

iv. Suppose L is a language over Σ for which there is a TM M that satisfies all of properties (1), (2), and (3). What can you say about L? Prove it.

v. In the earlier parts of this problem, you showed that just having two of these three properties doesn't tell you much about L. As you're writing your proof for this problem, make sure you see exactly where you're citing all three of the relevant properties. If you don't use all three properties, chances are that your proof is incorrect.