What is your name?: ___________________________ (0 points)

This test is closed book and closed notes. Use the back of the sheet if you need more room to write, although you shouldn’t need to. State your assumptions wherever it makes a difference. Each question is worth 10 points. You have 50 minutes.

1. Why were interpretive systems tolerated in the late 1940’s to mid 1950’s, given that an interpretive system adds so much overhead?

2. Imagine a programming language that did not have function calls or compound statements. Briefly state the implication of this. (Suggestion: consider how you might implement an if-then statement.)

3. The Spanish word *manana* literally is translated as *tomorrow*, however it also carries with it an implication of "some day" or "later on," not necessarily exactly within the 24 hour period from now. Someone who speaks Spanish might understand this, while a person who speaks English only may not have ever thought of this particular concept. This is an example of what hypothesis? Explain the hypothesis.
4. Write a grammar to accept the language: \( \{a^n b^n \mid n \geq 0\} \)

5. Draw the LISP linked-list interpretation of:
   
   \((A (B C (D E) (F (G H))) K)\)

6. Consider the attribute grammars such as the one discussed in class. Give an example of:
   
   a) A synthesized attribute

   b) An inherited attribute

   c) An intrinsic attribute
7. Suppose you were writing code to find flight information from Chicago to Nome, Alaska, where Chicago has a large busy airport and Nome has very few flights. Which would be better: forward chaining or backwards chaining? Explain your answer.

8. Does the following grammar maintain correct associativity for addition and multiplication? Explain why or why not.

```
<assign> --> <id> := <expr>
<id> --> A | B | C
<expr> --> <expr> + <term> | <term>
<term> --> <factor> * <term> | <factor>
<factor> --> ( <expr> ) | <id>
```

9. Rewrite the following EBNF grammar as a syntax diagram

```
B -> ( [‘b’], {‘c’ [<B>] ‘b’ } )
```
10. Write the PROLOG propositions to delete all instances of an element in a list. Call it `deleteAll`. For example, calling:
   
   ```prolog
   deleteAll( a, [a,b,c,a,d], Answer).
   
   would give:
   
   Answer = [b,c,d].
   ```