Fill in the blanks (1 point each)
1. ______syntactic sugar__________________________ is a term coined for additions to the syntax of a computer language that do not affect its expressiveness but make it nicer for humans to use.

2. ______abstraction___________________________ means complicated structures can be stated in simple ways by ignoring many of the details.

3. In the pre-parsing phase termed ______lexical analysis________________________ , characters are grouped into logical chunks (keywords, constants, etc). The actual strings that are matched are termed ______lexemes___________.

4. A metalanguage is a language used to describe____another language ____________.

5. _____Generality____________________________ refers to combining closely related constructs into a single more general one.

6. _____Orthogonality____________________ means features can be combined in any combination.

7. When similar things look the same, and different things look different we term it _______ Uniformity__________________________

8. With understatement independent, a user need only know about _____subset____ explicitly used to write the program.

9. ____Preciseness____________________________ means there exists a language definition that can answer programmers’ and implementers’ questions.

10. A glue language used for connected software components is considered to be part of the _______scripting_________________ paradigm.

Multiple Choice (2 points each) Circle the single best answer.

1. When a language feature is independent of the context in which it is used, we term it
   a. portable
   b. **orthogonal
   c. readable
   d. unified

2. Why do we care if a grammar is ambiguous?
   a. We don’t really. While ambiguity is interesting, it is not important.
   b. It takes longer to parse.
   c. **A different semantic meaning is associated with a different parse tree
   a. none of the above

3. Lexical analysis is separated from syntax analysis because:
   a. **lexical analysis is less complex.
   b. optimizing the lexical analyzer provides more benefits than optimizing the syntax analyzer.
   c. lexical analysis has to perform input/output operations and, therefore, is platform dependent.
   d. all of the above.
4. The C++ programming language is very popular because it is:
   a. backward compatible with C.
   b. object-oriented.
   c. widely available.
   d. **all of the above

5. An alternate term for lexical analyzer is:
   a. **scanner
   b. parser
   c. pushdown automaton
   d. all of the above

6. A concept of _______ means that a small number of primitive constructs can be combined in a small number of ways to build the control and data structures of the language.
   a. **simplicity
   b. parsimony
   c. relativity
   d. orthogonality

7. The most common programming languages, dating back to the 1940s are called:
   a. functional.
   b. object-oriented.
   c. rule-based.
   d. **imperative.

8. Which of the following represent a static semantic error in C++?
   a. **type mismatch
   b. missing semi-colon
   c. illegal character
   d. division by zero
   e. all of the above

9. Which of the following represent a dynamic semantic error?
   a. type mismatch
   b. missing semi-colon
   c. illegal character
   d. **division by zero
   e. all of the above

10. Ruby has the advantage that we can return multiple values by
    a. Using an asterisk before the variable name
    b. **Listing multiple values on the left hand side
    c. Returning a structure
    d. All of the above

11. In Ruby, when a code block is passed to a method, the scope of the non-local variables in the block are
    a. **the scope from which the block was called
    b. the scope where the block is executed
    c. non-local variables cannot be used in a block
1. (7 points) Write a regular expression for strings of 0’s and 1’s in which the string 101 never occurs as a substring (but all other combinations of 0 and 1 are allowed). For example, 0000, 11001, and 1100 are legal. 10011011 is illegal.

\[0^*1^*(00^+1^*)^*0^*\]

Some of you thought you could say something like \[^{101}\] – but the not doesn't work that way.

2. (8 points) What does the following Ruby code do?

```ruby
def generate (length=16)
  chars = 'abcdefghjkmnpqrstuvwxyzABCDEFGHJKLMNOPQRSTUVWXYZ23456789'
  word = ''
  length.downto(1) { |i| word << chars[rand(chars.length - 1)] }
  word
end
```

returns a random string of length 16 (or whatever is specified)

3. (6 points) Describe differences between the type checking and the evaluation of an expression which is dynamically typed. Carefully explain the roles of static and dynamic types.

static: known at compile time

dynamic: not known until compile time. Possibly may change at run time.

To do type checking at run time, I have to look up what the types are and see if they are legal.

At compile time, I can tell from the declared types.

Similarly, at run time, I need to determine and call the appropriate methods. If the types are known at compile time, it is a compile time determination.
4. (10 points) Write the Ruby code to do the following: Given an array of string words, build an array of only those words in the array that are anagrams of the first word in the array. Recall, two words are anagrams if they contain exactly the same letters.

myList = [posh hops are purchased at the food shop]

newList = [posh, hops, shop]

def match(a,b)
  aa = a.split('').sort
  bb = b.split('').sort
  aa==bb
end

mylist = ["posh", "hops", "are", "purchased", "at", "the", "food", "shop"]
p mylist
newlist = []
first = mylist[0];
mylist.each{|w|
  newlist << w if match(first,w)
}
p newlist
5. (10 points) in Ruby, define a method to count the number of times a vowel occurs in the given string.

```ruby
def vowel_count(myString)
  mystring = "Now is the time for all good men"
p mystring.count('aeiou')
```
6. (12 points) Consider the Ruby programming language as defined in this class. You are charged with making design changes to Ruby so that it can be used by professional developers to **rapidly** produce **correct, efficient, and maintainable** code. Identify **four** problems with Ruby that prevent it from attaining those goals. Explain each problem, explain how you would change Ruby, and justify your change. Be concrete. Choose problems and solutions to show off your knowledge. The aspects you change can occur at any level (e.g., lexical analysis, parsing, semantic analysis, runtime, language features, etc.).

Possible problems include:

1. dynamic type makes it hard to maintain
2. inserting into middle of an array makes it less efficient
3. All the special variables makes it hard to learn
4. The multiple ways to do the same thing makes it hard to maintain

How to fix them varies…
7. (6 points) Algol family languages are typically compiled, while Lisp family languages, where many issues cannot be settled until run time, are typically interpreted. Is interpretation simply what one “has to do” when compilation is infeasible, or are there actually some advantages to interpreting a language, even when a compiler is available? Explain.

Coding in an interpreted environment can be easier to develop with as the errors are shown immediately.

8. (8 points) Discuss the reasons why languages such as Fortran, Algol, and PL/I designed in 1950s and 1960s are less widely used than languages designed in the last 20 years.

Actually, there are a lot of reasons. Algol didn't have standardized I/O. PL/I was too big. Fortran didn't have recursion or objects. The old languages don't take advantage of new machines or needs of new applications.