Much of the answers are based off of the slides. The slides are very informative, so please review those if you haven't yet, or if you missed class.

Fundamental Concepts

1. State the definition of each of the following.
    a. Precondition

        A condition that must be satisfied when the execution of a program begins. This generally involves the algorithm's *inputs* as well as initial values of *global variables*.

    b. Postcondition

        A condition that should be satisfied when the execution of a program ends. This might be

        - A set of relationships between the values of inputs (and the values of global variables when execution started) and the values of outputs (and the values of global variables on a program's termination), or
        - A description of output generated, or exception(s) raised.

    c. correctness of an algorithm (for a given problem)

        Suppose that a problem is specified by a *single* precondition-postcondition pair *(P, Q)*. An algorithm for this problem is correct if it satisfies the following condition: If

        - Inputs satisfy the given precondition *P* and
        - The algorithm is executed

        then

        - the algorithm eventually halts, and the given postcondition *Q* is satisfied on termination.

    d. partial correctness of an algorithm (for a given problem)

        An algorithm is partially-correct if

        - Inputs satisfy the precondition *P*, and
        - Algorithm or program *S* is executed,

        then either

        - *S* halts and its inputs and outputs satisfy the postcondition *Q*

        or

        - *S* does not halt, at all.

    e. termination of an algorithm (for a given problem)

        If

        - inputs satisfy the precondition *P*, and
        - algorithm or program *S* is executed,

        then

        - *S* is guaranteed to halt!

    f. loop invariant

A condition R is a Loop Invariant if:

- Base Property: P implies that R is True before the first iteration of the loop and after testing G
- Inductive Property: if R is satisfied at the beginning of the ith execution of the loop body and there is an i + 1st execution, then the loop invariant holds immediately before that execution

g. loop variant

A function of f_L of program variables that satisfies the following additional properties:

- fL is integer-valued
- The value of f_L is decreased by at least one every time the loop body S is executed
- If the value of f_L is less than or equal to zero then the loop guard G is False (ie., the loop terminates)

2. Explain how “correctness,” “partial correctness,” and “termination” are related.

partial correctness + termination => total correctness

Background: Propositional Logic

1. Give truth tables for each of the following expressions. (Consult your textbook for MATH 271 or 273, or one of the references for discrete mathematics recommended for this course if you do not know what a “truth table” is.)
   a. p ⋁ q
   b. p ⋀ q
   c. p ⇒ q
   d. p ⇒ (q ⇒ p)
   e. p ⇔ ¬ p

p	q	p v q	p ⋀ q	p ⇒ q	p ⇒ (q ⇒ p)	p ⇔ ¬ p
T	T	T	T	T	T	F
T	F	T	F	F	T	F
F	T	T	F	T	T	F
F	F	F	F	T	T	F

2. The expression “p ⋁ q” is spoken aloud as “p or q,” and the expression “p ⇒ q” is spoken aloud as “p implies q.” Do these expressions have the truth values that you would expect, considering this information?

Let’s just say there are unexpected values…

If they do not always have the truth values you would expect then how (and when) are they different than expected?

“p or q” can be both inclusive and exclusive. Inclusive or means either p or q,

whereas exclusive or means either p or q but not both.
"p implies q" is false if p is true but q is not true (false). This statement is a one-way conditional, and p→q is always true if p is false. This should not be confused with p↔q, which is biconditional and is only true if p and q have the same truth values.

3. Are the following expressions well defined? What else (beyond the material presented in the lecture notes) would you need to interpret these?
   a. p ∧ q ∨ r
   b. p ∨ ¬ p ∧ q
   c. p ⇒ p ⇒ p
   parenthesis
4. This problem might be helpful if you were not sure about your answer for the previous one: Show that the following two expressions have different values when p is false.
   a. p ⇒ (p ⇒ p)
      p = F
      F ⇒ (F ⇒ F)
      F ⇒ T
      Since the condition is false, it doesn't matter what the implied value is, and the statement is true.
   b. (p ⇒ p) ⇒ p
      p = F
      (F ⇒ F) ⇒ F
      T ⇒ F
      Since the conditional is true, then the implied value should also be true. Since the implied value is actually false, the statement is false.
5. Some (but not all) of the logical operators we are considering are available as operators in Java.
   a. How is the expression

      p ∨ q ∧ r

      expressed in Java (as a function of p, q, and r)?

      p || q && r   which is the same as
      p || (q && r)
   b. If you did not add parentheses, but simply replaced the operators ∧ and ∨ with their equivalents in Java, then the result is equivalent either to

      (p ∨ q) ∧ r

      Or

      p ∨ (q ∧ r)

      Which one? Why?
      p v (q ∧ r), because in Java, equality operators will be evaluated first, then &&, then ||. Parenthesis can be added to change the order.
6. You should have seen propositional logic in a prerequisite course such as MATH 271 or 273 (or PHIL 279 or 377). However, some of the notation is probably different! For your

own reference, summarize the differences in notation that you find when comparing the presentation of propositional logic in these courses.

This depends on what courses you took, but you can find a decent summary here if required: http://cas2.umkc.edu/philosophy/vade-mecum/3-1.htm

Proof Rules: Simple Applications

1. Recall that the proof rules introduced in class can be used to reduce the problem of proving the correctness of algorithms to that of proving claims that do not have anything to do with algorithms at all (and that can be proved using the kind of proof techniques introduced in MATH 271 or 273).
2. List the claims that should be established to conclude each of the following.
   a. { true } i := 0 { 0 ≤ i ≤ n }

      Trivial, as the precondition is always true, meaning i = 0, and 0 ≤ i.
      Assuming n is a positive integer, 0 ≤ n, therefore i ≤ n and 0 ≤ i ≤ n.

   b. { 0 ≤ i ≤ n } i := i + 1 { 1 ≤ i ≤ n+1 }

      Precondition P implies $i_{old} \leq n$, ∴ $i = i_{old}+1$, and $i_{old} = i-1$.
      Since $n \geq i_{old}$, then $n \geq i-1$, and $n+1 \geq i$.
      Then we have $1 \leq i_{old} < i \leq n + 1$ ∴ $1 \leq i \leq n+1$.

   c. { i ≥ a } if (b ≥ a) then i := b else continue end if { (i ≥ a) ∧ (i ≥ b) }

      Case 1: b ≥ a
         i := b
         i = b ≥ a ∴ i ≥ a
         i = b ∴ i ≥ b
         Then (i ≥ a) ∧ (i ≥ b).
      Case 2: b < a
         i ≥ a > b ∴ i ≥ b
         Then (i ≥ a) ∧ (i ≥ b).

   d. { x = 0 } x := x + 1; x := x + 1 { x is an even integer }

      $x_0 = 0$
      $x_1 = x_0 + 1 = 0 +1 = 1$
      $x = x_1 + 1 = 1 + 1 = 2$
      Since x%2 = 2%2 = 0, x is an even integer.

3. Which of the programs listed in the previous question are partially correct?

   For all of the above, algorithm/program *S* halts (single statements only). Then given inputs that satisfy the precondition *P*, and the subsequent claims, we can see that postcondition *Q* is always satisfied. This satisfies the definition of partial correctness.

Linear Search solution: There is already a link on the course website (schedule), for the tutorial #2 supplement. I expect the file will be made public once all tutorial sections have been completed.