1. Go over Quiz #2 answers.

2. Reading input from the user:
   - How do you use cin?
   - What happens if you read input that doesn’t match the variable?
     ```cpp
     int x; cin >> x; // user enters t or 10.5?
     ```
   - What happens if you read input into two variables with one cin?
     ```cpp
     int x, y; cin >> x, y; // every variable needs its own cin
     ```
   - What happens if you read input into the variable before declaring it?
     ```cpp
     cin >> x; int x; // where does the program fail?
     ```

3. Describe the difference between syntax vs. semantics

4. Design and Testing (Assignment #3)

Design is very important when developing programs and there are a variety of ways to approach it. You may draw pictures, write it out in prose or structured text, use pseudo code, and more! The point of design is to give you a blueprint to follow while you are coding. This saves time debugging your program as you can catch mistakes early. **It is better to spend one hour of designing than it is to spend five hours debugging.**

George Polya developed a well-known model for problem solving in mathematics that is based on these 4 principles.

- Understanding the problem. (Recognizing what is asked.)
- Devising a plan. (Responding to what is asked.)
- Carrying out the plan. (Developing the result of the response.)
- Looking back. (Checking. What does the result tell me? Did I do it right?)

Polya’s steps 1, 2, and 4 do not directly deal with writing the solution (in programming that is the C++ code itself), but rather, the steps you need to make sure you write a correct solution/program that solves the given problem statement. With this said, make yourself familiar assignment #3 and Polya’s steps 1, 2, and 4.

**Understanding the Problem**

In your own words, explain what YOU think the problem is asking you to do. Document your uncertainties about the problem and anything else that you feel was unclear or vague. This is to ensure that YOUR understanding matches MY understanding of the problem.

**Devising a Plan/Design**

Provide an algorithm/pseudo code to help solve the problem. In addition, draw pictures/flow charts to help you devise your plan, as well as any other design decisions you make, such as how to manage your time, how to decompose the problem, where to start first, etc.

**Looking back/Testing**

This includes any checking/self-reflection you did while solving the problem, which includes using a calculator to make sure the output is correct, testing to make sure your code executes correctly and behaves the way you expect under specific circumstances,
using sources of information to make sense of the results, etc. However, you need to think about the input prior to implementation!!!

Please see an example of this document: Polya_template.pdf

In small groups, using Assignment #3 as an example, pick 2 functions and create a design for each including the following:

- How you understand the problem
- A plan for the functionality of the function as though it is in main (don’t worry about arguments/parameters and return values at this point)
- A testing table for each function you created a plan for.

If you have time…

Additional Material:
In small groups, answer the following question and define the following terms:

- What do you not understand from the material covered in class so far?
- Terms to define: input, conditional execution, logical vs. relational operator, short circuit

What is the difference in switch versus if/else? What are your limitations with a switch? What if you are missing a break in a switch? Can you nest switches and if/else?