CS 161 Recitation
Worksheet: Week 2

Terms to define
pre and post increment/decrement, %

Design and Testing (Program #2)
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 time 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 with Program #2 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.

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 the example design document distributed in recitation.
A digital copy is available online at:
https://web.engr.oregonstate.edu/~goinsj/resources/CS161/Polya_template.pdf

Using Program #2, as a group, consider the following problem statement:

Problem Statement

The local middle school would like some text adventure games to keep their students occupied during down time. The school is leaving it up to your skill and good judgement to develop a game. It is up to you what the story and theme is but there are some requirements:

- There must be **at least five different "if" statements** involved in the adventure
- **At least one** of those "if" statements must contain a nested "if" statement
- There must be **an element of chance** that would change a user's chosen path
- You must **handle invalid input** from the user.

An example run of the program could look as follows:

```
Hello and welcome to the Wilderness Adventure Game!
To go right enter 1, to go left enter 2: 1

You chose to go right. You have now entered the grasslands and are being followed
by oxen. They look friendly.
Enter 1 to befriend the oxen, enter 2 to run from the oxen: 1

You attempted to befriend the oxen. You think you can ride one of them.
Enter 1 to attempt to ride the ox, enter 2 to walk away: 11

Oops! That wasn't a valid response. Please try again:  1

Unfortunately fate was not on your side. [due to a random number within C++]
You were thrown from the ox's back and are forced to walk away.
Enter 1 to walk right, enter 2 to walk left: 1

You chose to go right. You have now entered the swamplands and have spotted an
approaching alligator. Enter 1 to hide, enter 2 to run home: 2

You chose to go home. The adventure has ended.
```

The rest of the implementation is up to you, but try to make your game as clean and attractive to play as possible.

Based on the problem statement, answer the following questions:

Understanding the Problem – Do you understand everything in the problem? Do you understand what is meant by "**There must be at least five different "if" statements involved in the adventure**", "**At least one of those "if" statements must contain a nested "if" statement**", and "**There must be an element of chance that would change a user's chosen path**"?

Design – What are the steps to create this adventure game? What are the steps for handling bad input? Write the flow-chart and/or pseudo-code for these steps.

Testing – Create a test plan with the test cases (**bad, good, and edge cases**). What do you hope to be the expected results?