1. **Goal**

The goal of this assignment is to build an expression tree and then use the tree to compute Bikini Bottom Residents taxes.

2. **Problem Statement [client's statement of their need]**

The Bikini Bottom Civic Association (BBCA) needs an expression evaluator so they can correctly compute Sponge Bob’s taxes for him. Also the program needs to output the expression in three different formats to satisfy various accounting standards.

3. **Requirements Analysis [What is the client’s problem?]**

Q: What kinds of expressions?
A: Expression involves integers, and the operators + (addition), − (subtraction), * (multiplication), / (division), and neg (negation).

Q: What needs to be done with each expression?
A: 1) for Sponge Bob, evaluate the expression
   2) for Patrick’s accountant print out the expression in prefix notation
   3) for Squidward’s accountant print out the expression in postfix notation
   4) for Mr. Krabs’s accountant print out the expression in infix notation

For example given an expression that adds 1 and 2, the outputs would be

1) “3”
2) “+ 1 2”
3) “1 2 +”
4) “(1 + 2)”

4. **Design [How]**

[This I’ll leave up to you. I’m happy to look at your design and corresponding plateau schedule if you like.]

I will also encourage some upfront OO thought, which will pay off in the end. To this end, I’m happy to look at your UML class diagram.

**What to hand in**

1. A well-formatted (no methods spanning page breaks!) 2-up printout of your source code.

2. A GitHub repo that includes your analysis and design in README.md and your Java code such that

   cd ~/cs312/<your asn6 repo>
   javac asn6/TestExpressions.java
   java TestExpressions

will run your program using jUnit.
Assignment Requirements  [ part of being a course rather than software development ]

- You must implement an expression tree and include appropriate node kinds.
- You must use jUnit to test your code (see the repo).
  (In Eclipse you need to include the jUnit 4 libraries. File > New > jUnit Test Case, will automatically add them.)
- To get full credit output only required parenthesis. For example, \((2 + 3) \times 4\) is more naturally written \((2 + 3) \times 4\) and \(((4 \times 5) + 6)\) is more naturally written \(4 \times 5 + 6\).
- Create your code to be part of a package named \texttt{asn6}.
- You can forgo header comments for short methods where the name makes it obvious, such as \texttt{public void add(Item it) \{ stuff.add(it); \} } from the last assignment.
- Use clear documentation and careful formatting. Be consistent in indentation and alignment of braces. Each open brace \“\{\” \textbf{must} be on its own line. Intent 2 or 4 spaces as the most.
- Each source code file must start with
  // This is my code
  // <Your Name>
  // CS312

Notes

- In Eclipse adding a jUnit test will ask about adding the paths to the necessary libraries ... add them :)  
- Mantra: rather than a “clever” encoding, let the data structures (e.g., the classes) do the work. (While perhaps not overly clever “char operation” is an encoding :) )
- The ability to hard code test expression using jUnit is meant as a gift. It is ok if you want to parse string input into expression trees, but you need not undertake that piece of joy.
- The following code suggests the need for a rethink:
  ```java
  else
  {
      //BIG ERROR
      return -999999999;
  }
  ```
- In class we may write code to find a specific operator using either a DFS or a BFS. We may also write an iterator that iterates through the integers contained in a tree.