Clicker Questions

Question 1: When I put the keyword **const** in front of a reference parameter, it means that:

- It will seg-fault if you try to change it.
- The procedure or method will not modify it.
- It is a constant, like Pi.
- The parameter is a fixed size data type, like **int** or **char**.
- The compiler will emit lots of errors and warnings.

Question 2: When I put the keyword **public** in front of a class' variable, it means that:

- A procedure that has an instance of the class may access or modify the variable.
- It is **const** by default.
- It is a global variable.
- The variable can only be modified in one of the class' methods.
- The compiler will emit lots of errors and warnings.

Questions 3-8: In all of the following, the answers are either "Like", "Dislike" or "Sometimes Like / Sometimes Dislike".

- **Question 3:** What does Dr. Plank think of declaring variables in the middle of procedures.
- **Question 4:** What does Dr. Plank think of tertiary expressions?
- **Question 5:** What does Dr. Plank think of Python?
- **Question 6:** What does Dr. Plank think of putting executable code in header files?
- **Question 7:** What does Dr. Plank think of reference parameters?
- **Question 8:** What does Dr. Plank think of The **const** keyword?

Answers Clicker Questions

Question 1: When I put the keyword **const** in front of a reference parameter, it means that the procedure or method will not modify it. If it does, then the compiler will flag an error. The keyword is nice, because whoever calls such a procedure or method knows that its parameter won't change. Please see https://web.eecs.utk.edu/~jplank/plank/classes/cs202/Notes/Procedures/index.html for more discussion.

Question 2: When I put the keyword **public** in front of a class' variable, it means that a procedure that has an instance of the class may access or modify the variable. If it's **private** or **protected**, then it may only be accessed or modified within implementations of the class (there's a little more to it than this because of inheritance, but just go with that for now).

Question 3: What does Dr. Plank think of declaring variables in the middle of procedures? Dislike. I like my variable declarations at the top of a procedure or method. That way, you know where to find them.

Question 4: What does Dr. Plank think of tertiary expressions? Sometimes like, sometimes dislike. I like them when they are simple and readable. When they are convoluted, or embedded in convoluted expressions, I dislike them.

Question 5: What does Dr. Plank think of Python? Dislike. Maybe that will change someday, but I find the language to be too much of a free-for-all. I like structure and explicit types. This doesn't mean that you have to dislike Python -- I think it is an important language that has expanded the reach of machine learning, and that if you learn it, it will only help your careers. I simply don't like it.

Question 6: What does Dr. Plank think of putting executable code in header files? Dislike. In my opinion, header files should have procedure prototypes and class definitions, and no running code. Headers should be static, and implementations should be dynamic. It's a matter of readability, and being able to trace through what your code is doing.

Question 7: What does Dr. Plank think of reference parameters? Like. They avoid copying and read better than pointers, and sometimes you want your procedure to be able to modify its arguments.

Question 8: What does Dr. Plank think of The **const** keyword? Like. It's really nice to know when things are being modified and when they aren't, and const lets you know.