Lab 2 - Modeling a Real-World Problem

Due: end of class

DESCRIPTION OF PROBLEM

Part 1

The problem of modeling populations can be expressed using the Verhulst Formula.

For this lab, you will be implementing the Verhulst formula stated on that page, inputting all parameters from the user using cin, and outputting a table of years and populations for years 1 to k using cout (k is another input parameter). Your output table format should look something like this (you don't need to lineup columns):

```
Year  Population
  1        57
  2        74
  3        82
  ...
  22      114
```

To attack the problem, you should do what was suggested in your 135 lecture (step 0 is reading the linked page to get a general idea of the problem):

1. Identify inputs and outputs of the problem (including datatypes)
2. Understand the problem well. Read the linked page again, concentrating on the problem you will be solving (in this case, the Verhulst formula given in the middle of the page.
3. Write pseudocode for the problem, as C/C++ comments.
4. Flesh out the pseudocode with real code.
5. Run, test, debug, repeat until complete!

You should implement the pseudocode using the appropriate C iteration constructs (for those who know about recursion, please don't use that for this lab). In addition, your code should have appropriate error checking to ensure that all parameters are legal, and reprompt if the user enters an invalid input. For example, you might want to ensure that the h input is <1 (among other constraints).

You can test your code on the 3 parameter sets given at the bottom of the linked page. In addition, you should also test that your program does appropriate error checking for inputs. Make sure you mention the test cases you tried in comments.

Part 2

You wish to identify critical values of g leading to oscillating and chaotic behavior. Using the values in the linked page, play with different values of g to see when it starts oscillating. Repeat for chaotic behavior. Write your answers in comments in your submission.

HAND IN

Your 136 instructor will tell you what to hand in and how.

GENERAL COMMENTS FOR ALL PROGRAMS THIS SEMESTER

You should have the following header on all programs:

```
/*
  Author: <name>
  Course: {135,136}
  Instructor: <name>
  Assignment: <title, e.g., "Lab 1">

  This program does ...
*/
```

GRADING

All 135 and 136 programs this semester will be graded on:

- Correctness: Does your program work?
- Testing: Have you generated sufficient and good test data to give reasonable confidence that your program works?
- Structure: Have you structured your code to follow proper software engineering guidelines? This includes readability and maintainability.
- Documentation: How well documented is your code? Good documentation does not repeat the code in English, but explains the point of each code block, highlighting any design decisions and/or tricky implementation details.