Programming Project #1

Assignment Overview

This project focuses on some mathematical manipulation. It is worth 10 points (1% of your overall grade). It is due Monday, January 25th before midnight.

The Problem

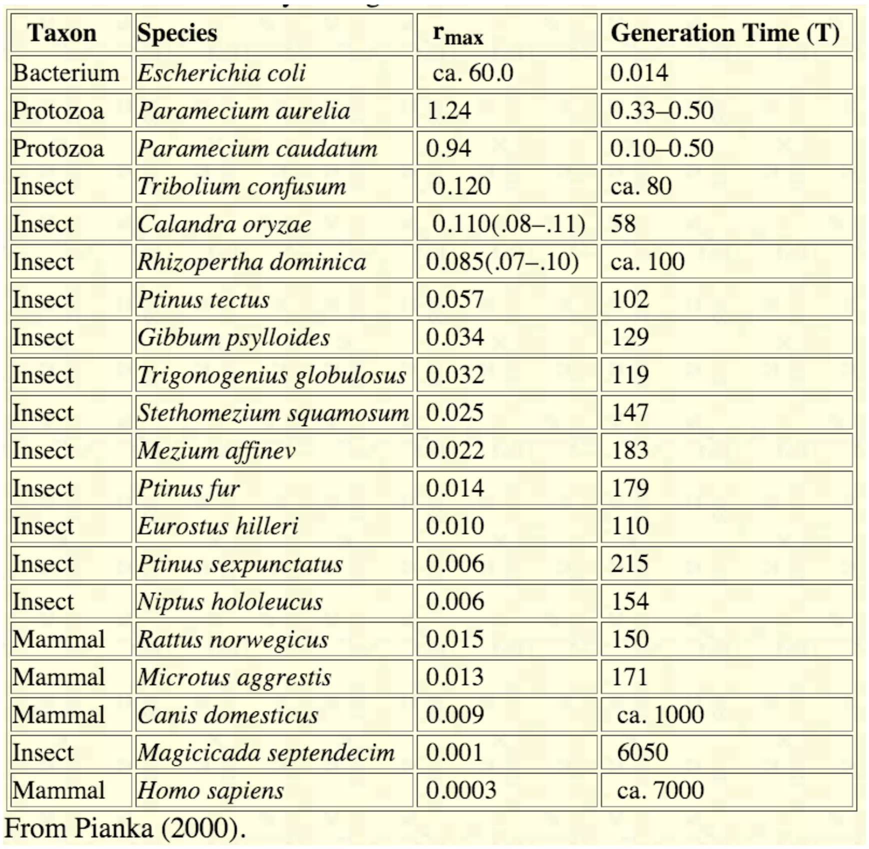
We are going to look at the problem of modeling population growth and predict a population size based on the initial population and some other parameters.

Some Background

The unrestricted growth of a population is based on a parameter (typically named *r*) called the intrinsic rate of natural increase, also sometimes called the Malthusian parameter after its creator Thomas Robert Malthus in 1798 (https://en.wikipedia.org/wiki/Malthusian_growth_model). Unrestricted growth means that an appropriate model of population growth is exponential. Malthus showed that unrestricted growth was of the form:

$$P_t = P_0 e^{rt} \quad \textbf{Equation1}$$

Where *t* is time and *r* is the Malthusian parameter. Common values for *r* below.



Taxon	Species	r_{max}	Generation Time (T)
Bacterium	*Escherichia coli*	ca. 60.0	0.014
Protozoa	*Paramecium aurelia*	1.24	0.33–0.50
Protozoa	*Paramecium caudatum*	0.94	0.10–0.50
Insect	*Tribolium confusum*	0.120	ca. 80
Insect	*Calandra oryzae*	0.110(.08–.11)	58
Insect	*Rhizopertha dominica*	0.085(.07–.10)	ca. 100
Insect	*Ptinus tectus*	0.057	102
Insect	*Gibbum psylloides*	0.034	129
Insect	*Trigonogenius globulosus*	0.032	119
Insect	*Stethomezium squamosum*	0.025	147
Insect	*Mezium affinev*	0.022	183
Insect	*Ptinus fur*	0.014	179
Insect	*Eurostus hilleri*	0.010	110
Insect	*Ptinus sexpunctatus*	0.006	215
Insect	*Niptus hololeucus*	0.006	154
Mammal	*Rattus norwegicus*	0.015	150
Mammal	*Microtus aggrestis*	0.013	171
Mammal	*Canis domesticus*	0.009	ca. 1000
Insect	*Magicicada septendecim*	0.001	6050
Mammal	*Homo sapiens*	0.0003	ca. 7000

From Pianka (2000).

However, except for early on in population growth, growth is rarely purely exponential. There are limitations (on resources, on space, etc.) that limit potential growth. A more realistic model is is the Verhulst model, introduced by Pierre-François Verhulst in 1838. Verhulst introduced a new variable, the carrying capacity (usually represented by the letter *K*) that represents the maximum population size allowable under the present conditions. His model allowed growth to be exponential when the population size is far away from *K* but also allow the growth to slow as it neared the size *K*. The equation, often called the Logistic Equation (https://en.wikipedia.org/wiki/Logistic_function#In_ecology:_modeling_population_growth), is shown below:

$$P_t = \frac{KP_0 e^{rt}}{K + P_0(e^{rt} - 1)}$$

$$s.t. \lim_{t\to\infty} P_t = K \qquad \textbf{Equation2}$$

Using this equation, you can predict the population at time *t* (P_t) based on:

- P_0 : the initial population size
- *K* : the carrying capacity (the max size that is allowable)
- *t* : the time (units unspecified)
- *r* : the Maltusian parameter

We are going to write a program implements the logistics equation.

Program Specifications

Your program will do the following:

1. Prompt for four floating point values listed above (P_0, *r, K, t)*
   a. You should be able to read all four variables off of a single, space separated line of input value.
2. Print the following results on a single line
   a. Float: the population size the unrestricted (equation 1) predicts
   b. Float: the population size the retricted (equation 2) predicts
   c. Use the Sample Interaction (next page) to set output format

Deliverables

proj01.cpp -- your source code solution (remember to include your section, the date, project number and comments).

1. Please be sure to use the specified file name, i.e. "proj01.cpp"
2. Save a copy of your file in your CSE account disk space (H drive on CSE computers).
3. You will electronically submit a copy of the file using the "handin" program: http://www.cse.msu.edu/handin/webclient

Assignment Notes:

1. The exponential function is represented by the function `exp` . It requires the `cmath` include. Look it up for details!
2. We might as well try to make the output somewhat readable. You can look this up in the text or on the internet, but you can use the following modifiers that affect how numbers print.

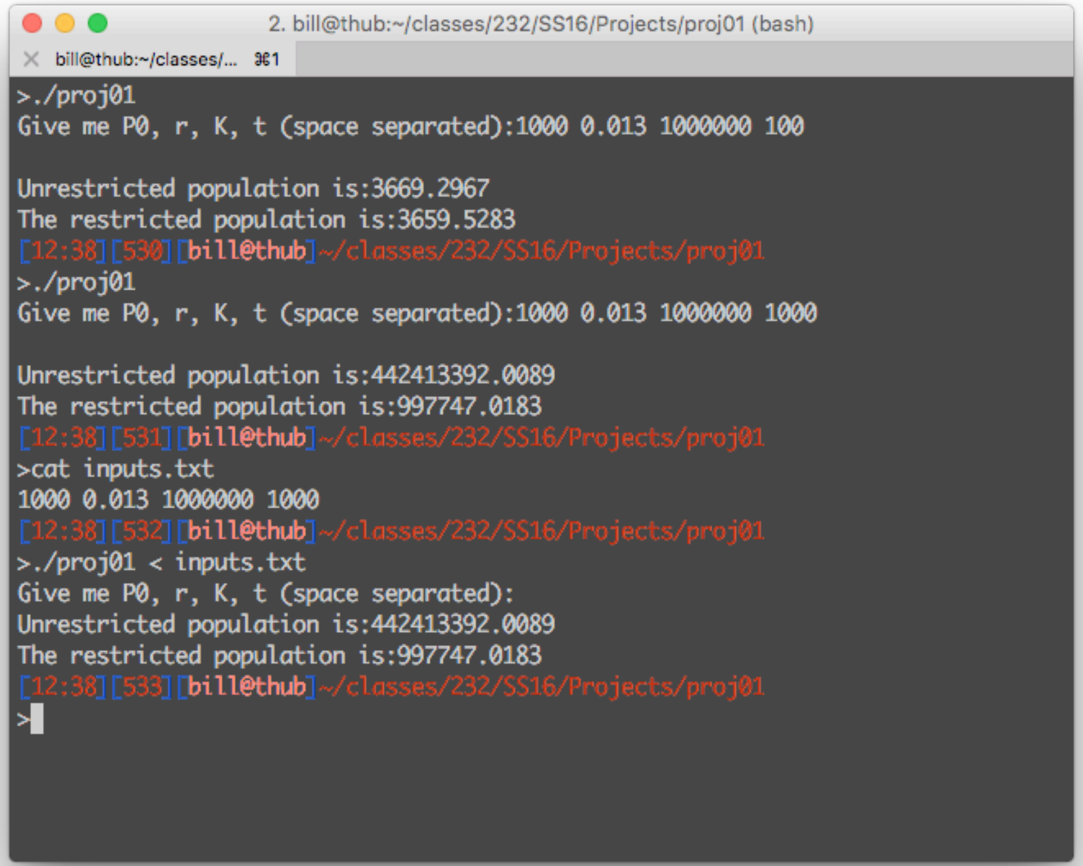
   a. `cout << fixed` Elements will be printed as floating point numbers (ex 123.456)
   b. `cout << scientific` Elements will be printed in scientific notation (ex 1.23456 x 10^2)
   c. `cout << setprecision(2)` Floating point numbers will have 2 values after the decimal point and will be rounded, (123.46)
   d. Thus `cout << fixed << setprecision(2) << 123.4567 << endl;` will print 123.46
3. The following statement will read two variables off of the same, space separated line. It is an example of chaining input:
   ```
   …
   double d1, d2;
   cin >> d1 >> d2;
   …
   ```
4. Look at the example output for the proper way to read and write values.
5. You do not have to check for bad input values. In general, we will explicitly indicate the errors we are looking for, but for now we are not checking for input errors.
6. If you read off of a single line for the input, you can do a handy trick off the command line. You can redirect a file (with that single line of input) to your main program. Thus in the terminal, in the directory where your program proj01 resides, you can do:

   ```
   ./proj01 < fileOfInput.txt
   ```

   This will automatically feed the input to the `cin` statement and produce the output. Makes it easier to test things (and to grade them too!!).
7. You can check your calculations on http://www.wolframalpha.com/ or using python or some other approach that you are comfortable with.

Getting Started
1. Using Geany create a new program.
2. If you are in a CSE lab or on x2go, select the H: drive as the location to store your file.
3. Save the name of the project: proj01.cpp
4. Prompt for some of the values and print them back out, just to check yourself.
5. When you do a calculation, use the provided table as a way to check yourself.
6. Use the handin web site to hand in the program (incomplete as this point, but you should continually hand things in).
7. Now you can calculate the individual elements of the project. After writing code for each of the below aspects, compile and print out values to make sure you are getting the correct values. Do it ***INCREMENTALLY***, that is write and check various calculations
8. Now you enter a cycle of edit-run to incrementally develop your program.
9. Use handin to hand in your final version.
10. ***Remember***: the project must compile on the lab machines or the x2go environment for project credit. Check it before you hand it in!

Sample Interactions


2. bill@thub:~/classes/232/SS16/Projects/proj01 (bash)
bill@thub:~/classes/... ⌘1

```
>./proj01
Give me P0, r, K, t (space separated):1000 0.013 1000000 100

Unrestricted population is:3669.2967
The restricted population is:3659.5283
[12:38][530][bill@thub]~/classes/232/SS16/Projects/proj01
>./proj01
Give me P0, r, K, t (space separated):1000 0.013 1000000 1000

Unrestricted population is:442413392.0089
The restricted population is:997747.0183
[12:38][531][bill@thub]~/classes/232/SS16/Projects/proj01
>cat inputs.txt
1000 0.013 1000000 1000
[12:38][532][bill@thub]~/classes/232/SS16/Projects/proj01
>./proj01 < inputs.txt
Give me P0, r, K, t (space separated):
Unrestricted population is:442413392.0089
The restricted population is:997747.0183
[12:38][533][bill@thub]~/classes/232/SS16/Projects/proj01
>
```