Reading to Input Failure, Simple File I/O, Arithmetic Calculations, Using Standard Functions

This programming assignment uses many of the ideas presented in sections 3 and 4 of the course notes, so you are advised to read these notes and the following program specification carefully.

The Program Specification: Exponential Growth and Decay

If you put $5000 in the bank and earn 6% annual interest, how much money do you have after 30 years if you don't make any withdrawals? If a radioactive substance has a half-life of 27.7 years, how long does it take for there to be only 1/1,000,000 of the original amount of the substance? Problems such as these involve exponential growth or decay and are common to many different sciences[1]. For this assignment, you will write a simple program that will apply one of the standard mathematical models to an exponential growth problem.

Say we have a certain amount of a resource, M_0. This resource is being consumed at a rate of R_0 units per year. Additionally, this rate of consumption is increasing at an annual growth rate, k. Then we can compute the number of years until we run out of the resource using the formula for T_e shown below. We can compute the number of years until we run out of half of the resource using the formula for T_h shown below.

$$T_e = \frac{1}{k}\ln\left(\frac{kM_0}{R_0}+1\right) \qquad T_h = \frac{1}{k}\ln\left(\frac{kM_0}{2R_0}+1\right)$$

where:

- T_e is the number of years until we run out of the resource
- T_h is the number of years until we run out of half of the resource
- k is the annual growth rate of the consumption (remember that 5% = 0.05)
- ln is the symbol for the natural logarithm (log base e)
- M_0 is a constant value that is the initial amount of the resource
- R_0 is a constant that is the annual rate of consumption

One application of this formula is to compute the amount of time before national or worldwide reserves of oil or coal run out. For example, suppose the current estimated amount of coal reserves is $M_0 = 379.0 \times 10^9$ tonnes (1 tonne = 10^3 kg) and the annual rate of consumption is $R_0 = 0.6 \times 10^9$ tonnes. Additionally, assume the consumption rate is increasing by 3% every year (k = 0.03). The total number of years the coal reserve will last is:

$$\begin{aligned} T_e &= \frac{1}{0.03}\ln\left(\frac{0.03\times(379\times10^9)}{0.6\times10^9}+1\right) \\ &= \frac{1}{0.03}\ln(0.03(631.67)+1) = \frac{1}{0.03}\ln(19.95) = \frac{1}{0.03}(2.9932) = 99.77 \end{aligned}$$

So for this example, the supply will last 99.77 years. A similar calculation using the formula for T_h will show that we will run out of half of the reserves in about 78 years. This illustrates an interesting property of exponential growth; that it would take 78 years to use the first half of the resource, but that in about 20 more years, all of the resource would be exhausted if the consumption continued to increase at the annual growth rate.

[1] Much of the material for this assignment is based on information found in the introductory physics textbook, "Physics, 3rd Edition," by Tipler, pp.586-588.

Input file description and sample:

Your program **must** read its input from a file named `expgrowth.in` — use of another input file name will result in a score of zero. The first line of the input file contains column labels which should be ignored. The second line contains the initial reserve amount (M_0), followed by the initial consumption rate (R_0). The third line contains a column label that should be ignored. Each remaining line of the input file will contain a single annual growth rate value in percent, followed by a single newline character.

You may assume that all the input values will be logically correct (no negatives, for instance). You may also assume that the annual growth rate will always be greater than zero and that the units of measurement/size for the initial reserve and consumption rates are the same.

For instance:

```
Initial Reserve        Initial Consumption Rate
  379.0                  0.6
Annual Growth Rate (in percentages)
1.0
2.3
3.0
5.0
12.5
```

The input values will all be given as real numbers and should be read in as type `double`.
Note that you must **also not make any assumptions** about the number of lines of data in the input file. Your program must be written so that it will detect when it's out of input and terminate correctly. A technique for this will be discussed in class; see also slides 4.20 – 4.21 in the course notes.

What to Calculate:

You will write a program to read the input file and compute the number of years until the reserve is totally consumed for each of the annual growth rates. That is, for each annual growth rate, you will need to use the initial reserve amount and consumption rate in the formula given above to compute the number of years the reserve will last at that annual growth rate. You will also compute the number of years until the reserve is half (50%) consumed for each of the annual growth rates.

To perform these calculations, you will need to use the formulas given above. The formulas require using the natural logarithm function, ln. Fortunately, this function may be found among the standard library functions in C++. The natural logarithm function in the `math.h` library is called **`log( )`**; it takes one parameter of type double and returns the value of the natural logarithm of that parameter. So, the statement:

```
double avalue = log(2.7183);
```

would assign the value 1.0000 to the variable `avalue`. To use this function you must include the standard header file `math.h`.

Output description and sample:

A sample output file produced from the sample input file above is shown below. The first line of your output should identify you by name, as shown. The second line should include the title "Exponential Growth and Decay" only. The third line should be blank. The fourth line should include the heading "Initial Data" only. The fifth lines should display the initial reserve amount, labeled exactly as shown. The sixth line should display the initial rate of consumption, labeled exactly as shown. The seventh line should be blank.

Next your output file will contain a table, with one line of output for each annual growth rate value given in the input file. Each line of the table should list the annual growth rate as a percentage, the number of years until the reserve is completely exhausted at that growth rate (time to use 100%), and the number of years until half of the reserve is exhausted at that growth rate (time to use 50%). The table columns should have labels, exactly as shown on the eighth line. There should be a line of delimiters immediately after the column labels (on the ninth line), and another to mark the end of the table. Each line of output (including the last) should be followed by a newline character.

The initial reserve amount and initial consumption rate should be printed with precision 2. The annual growth rate, time to use 100%, and time to use 50% values should be printed with precision 1.

Your program must write its output data to a file named `expgrowth.out` — use of any other output file name will result in a score of zero.

```
Programmer:  Donald Allison
Exponential Growth and Decay

    Initial Data
Reserve:    379.00
Rate:         0.60

   AGR     Time100   Time50
-----------------------------
   1.0     199.0     142.5
   2.3     119.2      91.8
   3.0      99.8      78.3
   5.0      69.7      56.4
  12.5      35.1      29.6
-----------------------------
```

You are not required to use this exact horizontal spacing, but your output must satisfy the following requirements:

- You must use variables of type `double` for all the real numbers. If you use `float` variables, your answers may not be as accurate.
- All decimal values must be printed to show the same number of decimal places as in this sample.
- You must use the specified header and column labels, and include your name in the first line as shown.
- You must arrange your output in neatly aligned columns, with a label identifying the contents of each column. Use spaces, not tabs to align your output.
- You must use the same ordering of the columns as shown here.
- You must print a newline at the end of each line, including the line of hyphens marking the end of the table.

Programming Standards:

You'll be expected to observe good programming/documentation standards. All the discussions in class about formatting, structure, and commenting your code should be followed. A copy of *Elements of Programming Style* is included with the course notes — if you don't have a copy we strongly suggest you read the on-line edition (available from the course web page). Some specifics:

- Your header comment must describe what your program does.
- You must include a comment explaining the purpose of every variable or named constant you use in your program.
- You must use meaningful identifier names that suggest the meaning or purpose of the constant, variable, function, etc.
- Use named constants instead of variables where appropriate.
- Precede every major block of your code with a comment explaining its purpose.
- You must use indentation and blank lines to make control structures (like loops and if-else statements) more readable.

Hints:

This program requires that you know how to manage file-oriented input/output operations — the slides and the text provide good examples and guidelines. Your program must read lines of input data until there is no more data to be processed.

You'll have to use **manipulators** to manage the formatting of your output. Read the discussion on slides 4.16 – 4.19 carefully. There are important clues there.

You also have to do some mathematical calculations that go beyond mere arithmetic. The C++ language includes most of the standard mathematical functions, including trigonometric, logarithmic and exponential functions. To use them, you need to add another #include at the beginning of your program: `#include <math.h>`

Testing:

Obviously, you should be certain that your program produces the output given above when you use the given input file. However, verifying that your program produces correct results on a single test case does not constitute a satisfactory testing regimen. You should test your program on all the posted input/output examples given along with this specification. Those were generated by the same program that will be used to test your solution.