

C++ Mathematical Calculations:**Information Retention**

For this project, you will complete a partial implementation of a simple C++ program, by writing the necessary calculations.

Intuitively, it is clear that when a human learns a body of facts the amount of that information that is still retained will decrease over time. In psychology, Ebbinghaus has developed a mathematical model for this phenomenon:

$$P(T) = (100 - A)e^{-BT} + A$$

where P is the percentage of the information that is retained after T weeks, and A and B are constants that depend on the individual being studied. In essence, A is the percentage of the information that will be permanently retained by the individual, and B is the rate at which information is lost.

In this project, you will complete a program that reads the data needed to apply the given formula to predict the information retention for an individual over a given number of weeks. The partial implementation given below manages reading the input data and writing some of the output data. Your task is to implement the necessary calculations and complete the output. Be sure to follow the specification below carefully. As was the case with the previous project, the given code uses some C++ features that have not been covered yet; the comments should make the logic reasonably clear.

Sample input and corresponding output:

Here is a sample input file for the program (named "EbbinghausData.txt"). The first line contains the name of the subject of the study. The second and third lines specify the two constants for the Ebbinghaus Model, and the fourth specifies the number of weeks for which an estimate is to be given.

```
Danforth Quail
50
0.8
5
```

Here is a sample output file for the program, which must be named "Ebbinghaus.txt". It begins with two lines identifying the programmer (you) and the specific project, followed by a blank line. We will always require that information at the beginning of your output file. The remainder of the output file reports the results computed by the program.

There will be five lines of computed output. The first specifies the name of the subject, the second is blank, the third is purely a row of column labels corresponding to the number of weeks to be reported, the fourth is simply a separator, and the fifth contains the percentage of information loss for each week. If you have read the *Student Guide to the Curator*, you already know that all the fixed text must be precisely as shown in the sample output. The output should be aligned for easy readability.

```
Programmer: <your name here>
CS 1044:  Information Loss

Subject:  Danforth Quail

Week:    0    1    2    3    4    5
-----
Loss:    0   28   40   46   48   50
```

Additional samples of input and correct output will be available on the course website.

Note: the numbers in the last line were obtained by truncating each value from the Ebbinghaus formula to an integer before any other calculations are done with that value.

The program source code:

The following program shell is provided as a starting point for the project. You are not required to use the given code; you may modify it as much or as little as you wish. Regardless, be sure that your final implementation meets the requirements given on the *Programming Standards* page on the course website, especially those for comments.

```
// Programmer:      Bill McQuain and <your name here>
// Last modified:   January 20, 2007
// Compiler:        Visual C++ 2003
//
// This program applies the Ebbinghaus Model of information retention
// to data about a given individual.  The Ebbinghaus Model says that
// if a subject initially knows 100% of the information about a subject,
// then after T weeks the subject will retain approximately P% of that
// information, where P is given by the formula:
//
// 
$$P = (100 - A)e^{(-Bt)} + A$$

//
// where e is the Euler number and A and B are constants that depend
// on the individual in question.
//
#include <fstream>          // for file streams
#include <iomanip>          // for output format manipulators
#include <string>          // for string variables
#include <cmath>           // for pow() function
using namespace std;      // put all of the above in scope

const double EULER = 2.718281828459045; // approximation of Euler's number

int main() {

    // Attach an input stream to the data file:
    ifstream Data("EbbinghausData.txt");

    string NameOfSubject = "Anon";    // name of the subject
    int A = 0;                        // Ebbinghaus constants
    double B = 0.0;
    int NumberOfWeeks = 1;            // time interval

    // Read the name of the subject:
    getline(Data, NameOfSubject);

    // Read the Ebbinghaus constants and time interval:
    Data >> A >> B >> NumberOfWeeks;

    // Close the data file:
    Data.close();

    // Attach an output stream to the log file:
    ofstream Log("Ebbinghaus.txt");

    // Write an identification header:
    Log << "Programmer: <your name here>" << endl;
    Log << "CS 1044: Information Loss" << endl;
    Log << endl;

    // Write the subject data:
    Log << "Subject: " << NameOfSubject << endl;
    Log << endl;

    // Write the column header label:
```

```
Log << "Week:";
// Write the column headers:
int Week = 0;
while ( Week <= NumberOfWeeks ) {
    Log << setw(5) << Week;
    Week = Week + 1;
}
Log << endl;

// Write the separator:
Log << "-----";
Week = 0;
while ( Week <= NumberOfWeeks ) {
    Log << "-----";
    Week = Week + 1;
}

// Write the data label:
Log << endl << "Loss:";

// Write the information loss data:
Week = 0;
while ( Week <= NumberOfWeeks ) {

    // Apply Ebbinghaus' formula for the current week:

    // Write the information loss for the current week:

    // Step to the next week:
    Week = Week + 1;
}
Log << endl;

// Close the log file:
Log.close();
return 0;
}
```

Submitting your program:

You will submit this assignment to the Curator System (read the *Student Guide*), and it will be graded automatically. Instructions for submitting, and a description of how the grading is done, are contained in the *Student Guide*.

You will be allowed up to ten submissions for this assignment. Use them wisely. Test your program thoroughly before submitting it. Make sure that your program produces correct results for every sample input file posted on the course website. If you do not get a perfect score, analyze the problem carefully and test your fix with the input file returned as part of the Curator e-mail message, before submitting again. The highest score you achieve will be counted.

The *Student Guide* and other pertinent information, such as the link to the proper submit page, can be found at:

<http://www.cs.vt.edu/curator/>

Pledge:

Each of your program submissions must be pledged to conform to the Honor Code requirements for this course. Specifically, you **must** include the following pledge statement in the header comment for your program:

```
//      On my honor:
//
//      - I have not discussed the C++ language code in my program with
//        anyone other than my instructor or the teaching assistants
//        assigned to this course.
//
//      - I have not used C++ language code obtained from another student,
//        or any other unauthorized source, either modified or unmodified.
//
//      - If any C++ language code or documentation used in my program
//        was obtained from another source, such as a text book or course
//        notes, that has been clearly noted with a proper citation in
//        the comments of my program.
//
//      - I have not designed this program in such a way as to defeat or
//        interfere with the normal operation of the Curator System.
//
//      <Student Name>
```

Failure to include this pledge in a submission is a violation of the Honor Code.