

**Input Failure Control and Decisions:**

**Detailed Paycheck for employees**

For this project you will modify and extend your implementation for Project 3 to employ an input-failure loop and more sophisticated decisions. The input file will be radically different, but the output file will have precisely the same format.

**Sample input data:**

Here is a sample input file, named `Salaries.txt`, for the program. The first two lines are for descriptive purposes and do not contain any input data. Each of the following lines in the input file contains exactly two fields. The first field is the name of an employee, and the second field is his/her gross pay. The two fields are separated by a single tab (`\t`) character.

Name	Gross Pay
=====	=====
Bugs Bunny	4000.00
Elmer Fudd	3200.00
RoadRunner	5000.00
Yosemite Sam	6000.00

There can be a varying number of employees specified in the input file. You should not design your program to contain only four lines. There is guaranteed to be at least one employee in the input file.

The input values are guaranteed to be syntactically and logically correct. This means that there will a tab character between the two input fields.

**Calculations:**

There are a lot of deductions made from this employee’s paycheck. These include the Federal income tax, state income tax, social security tax, medicare tax and deductions for health insurance and pension plan. The deductions are outlined below.

Deduction type	Deduction Percentage or amount
State Income Tax	6%
Social Security tax	5%
Medicare tax	3%
Pension plan	4%
Health Insurance	\$150.00

The deductions are either in the form of a percentage of the gross pay or a direct dollar amount as indicated for the health insurance. The federal income tax is based on the monthly gross pay of the employee. The deduction percentage is indicated in the table below

Salary	Federal Income Tax
< 2000	12%
2000 – 3999	15%
≥ 4000	20%

Given the data in the input file, you must list the pay details for each employee. The required format is shown in the sample output file given next.

**Sample output:**

Here is a sample output file, named `PaycheckDetails.txt`, which corresponds to the input data given above:

```

Programmer: Mir Farooq Ali
CS 1044 Spring 2004 Project 4

Acme Corporation
Paycheck details

=====
Employee Name      Gross Pay   Deductions   Net Pay
=====
Bugs Bunny         4000.00    1670.00     2330.00
Elmer Fudd         3200.00    1206.00     1994.00
RoadRunner         5000.00    2050.00     2950.00
Yosemite Sam       6000.00    2430.00     3570.00
=====

```

The first two lines identify the programmer and project, as usual. Then there is a blank line, followed by the company name and descriptive header. This is followed by yet another blank line and a few more descriptive headers. Then, there is a line for each employee that details the name, gross pay, total deductions and net pay.

If you have read the description of how the Curator scores your program in the *Student Guide*, you know that is important that you use the same spelling and capitalization for all the labels shown above. The horizontal spacing does not effect scoring unless you combine things that should be separate or separate things that should be combined. You are free to experiment with the horizontal layout but you should try to align the columns neatly.

Note that you must insert blank lines and divider lines as shown.

**Evaluation:**

Everything that was said in the specification for the earlier about testing still applies here. Do not waste submissions to the Curator in testing your program! There is no point in submitting your program until you have verified that it produces correct results on the sample data files that are provided. If you waste all of your submissions because you have not tested your program adequately then you will receive a low score on this assignment. You will not be given extra submissions.

Your submitted program will be assigned a score, out of 100, based upon the runtime testing performed by the Curator System. We will also be evaluating your submission of this program for documentation style and a few good coding practices. This will result in a deduction (ideally zero) that will be applied to your score from the Curator to yield your final score for this project.

Read the *Programming Standards* page on the CS 1044 website for general guidelines. You should comment your code in the same manner as the code given for the first two programming assignments. In particular:

- You should have a header comment identifying yourself, and describing what the program does.
- Every constant and variable you declare should have a comment explaining its logical significance in the program.
- Every major block of code should have a comment describing its purpose.
- Adopt a consistent indentation style and stick to it.

Your implementation must also meet the following requirements:

- Choose descriptive identifiers when you declare a variable or constant. Avoid choosing identifiers that are entirely lower-case.
- Use named constants instead of literal constants when the constant has logical significance.
- Use C++ streams for input and output, not C-style constructs.
- Use C++ string variables to hold character data, not C-style character pointers or arrays.

- **Note: you are explicitly forbidden to write any user-defined functions for this program. This will make the program somewhat repetitive, and physically longer than the alternative. To some extent, that's the reason for this restriction.**

Understand that the list of requirements here is not a complete repetition of the *Programming Standards* page on the course website. It is possible that requirements listed there will be applied, even if they are not listed here.

### 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 five 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* can be found at: <http://ei.cs.vt.edu/~eags/Curator.html>

The submission client can be found at: <http://eags.cs.vt.edu:8080/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.**