

**Putting the basics together:****Calculating Paycheck**

It's finally time to write a complete program. This project will use many of the C++ features that were illustrated in the source code you were given for the first and second programming assignments. The resulting program will read simple data that includes the gross pay for an employee and produce the net pay based on some deductions. This will require reading a data from an input file, doing simple calculations, and writing a simple output file. In addition, your program will have to use a C++ selection mechanism to make decisions regarding several possible charges.

**Sample input data:**

Here is a sample input file, named `Salary.txt`, for the program. The first line specifies the name of the employee, the second specifies the month, and the third line specifies the gross pay.

Each data value is preceded by a descriptive label that ends with a colon character ': '.

Name:	Wile E. Coyote
Month:	February, 2004
Gross Pay:	4000.00

The input values are guaranteed to be syntactically and logically correct. The name, month and gross pay are just string data. For formatting purposes, you may assume that the name will be no more than 40 characters long.

**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
Federal Income Tax	20%
State Income Tax	6%
Social Security tax	5.75%
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. Given the data in the input file, you must create a paycheck for the employee that lists the actual amounts for all these deductions, in addition to the net pay. The required format is shown in the sample output file given next.

**Sample output:**

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

```
Programmer: Mir Farooq Ali
CS 1044 Spring 2004 Project 3

Acme Corporation
Paycheck details
Name: Wile E. Coyote
Month: February, 2004

Gross Pay:                $4000.00

Federal Income Tax:       $800.00
State Income Tax:        $240.00
Social Security Tax:     $228.00 230.00
Medicare Tax:            $120.00
Pension Plan:            $160.00
Health Insurance:        $150.00

Net Pay:                  $2302.00 2300.00
```

As usual the first two lines just identify the programmer and project. Next, there is a blank line followed by four lines of details about the company and employee. Next there is a blank line, followed by details of each deduction and finally, the 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 as shown.

**Suggested implementation plan:**

You should design your solution piece by piece. Begin by identifying the major tasks that have to be done, and then add detail for each of those tasks. Your implementation should be developed in the same manner. Here's a suggested order of implementation.

First, implement the input code to read the employee name, month and gross pay, and write the output code to print them back out in the specified format. This will require declaring the appropriate stream variables and setting up the file streams, and declaring and using variables to store the input values. Test this code and make sure it produces correct results. Once you know this part works you should never have to worry about it again.

Second, implement the code to perform the deductions. It is recommended to use variables for each individual deduction and the net pay. The net pay is the result of subtracting all the deductions from the gross pay. Finally, implement the code to write out each individual deduction and the net pay to the output file.

By implementing the program feature by feature you let yourself concentrate on solving one small part of the problem at a time. You also should only have to test one part of it at a time as well.

## Evaluation:

Everything that was said in the specification for Projects 1 and 2 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 (the administration page) 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://www.cs.vt.edu/curator>

The submission link is at: <http://eags01.cs.vt.edu:8080/CuratorS04/index.jsp>

**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.**