**Programming Assignment 2**

File IO

Control structures

Looping

Operator precedence

Error checking (bad data, file stream fail state, etc.)

Use of proper data types

Exact output

**Write a program that can be used to create a detailed billing report for a company that provides tax consultations for customers:**

1. The information will be in the provided data file **customers.txt**. The data file consists of several rows of data, as follows (commas are not in the file):

Customer First Initial, Customer Last Name, Customer ID, Yearly Income, Consulting Time in Minutes, Consulting Agent ID

1. Request the name of the input file from the user. Do not “hard-code” the file name in the program.
2. Read the file using a looping construct that checks for the end of file (not the number of rows in the data file).
3. The agent ID maps to an agent last name and hourly rate. Evaluate the agent ID using a SWITCH statement to retrieve the agent last name and hourly rate. The mapping values are as follows. You will use these as constants/literals in your program:

Agent ID 1 = Bortles; Hourly Rate = 55.00

Agent ID 2 = Cower; Hourly Rate = 55.00

Agent ID 3 = Watt; Hourly Rate = 72.00

Any other agent ID is an error and needs to be handled accordingly. See **Error Handling** below.

1. Determine if the customer is a low income customer and set a flag accordingly (Y or N). This is to be output in the report (see below) and **may** be used for calculating the customer’s amount due (bill). A customer is low income if the yearly income is less than or equal to $25,000.
2. Calculate the customer’s amount due (bill) as follows:

If the customer is a low income customer:

If the consulting time is less than or equal to 30 minutes, the amount due (bill) is 0.00

If the consulting time is greater than 30 minutes, the amount due (bill) is 35% of the hourly rate (billing rate) times (the consulting time minus 30 minutes) divided by 60 minutes (i.e., billed by the hour). For example, if the consulting time is 32 minutes the calculation would be:

Hourly rate \* 0.35 \* (32 - 30) / 60

Otherwise (the customer is **not** a low income customer):

If the consulting time is less than or equal to 20 minutes, the amount due (bill) is 0.00

If the consulting time is greater than 20 minutes, the amount due is 65% of the hourly rate (billing rate) times (the consulting time minus 20 minutes) divided by 60 minutes.

1. Request an output file name from the user. Do not “hard-code” the file name in the program.
2. For each error-free row of input in the data file, output the following to the **console** (screen) in a neat, readable format. Output each input line on one output line. Use manipulators to output values in readable columns. Use dollar signs before monetary values. All monetary values must have 2 decimal points. Use headings to indicate values displayed (you DO NOT need to use the headings below, you may use your own):

Customer Name (initial and last name)

Customer ID

Yearly Income

Consulting time (in minutes)

Amount Due (customer’s bill)

Low Income (Y or N indicator)

Agent Name (consulting agent)

Agent’s Hourly Rate

Output must be in the following form, i.e., a report header and one heading for all lines for output:

Billing Report

Customer Name Customer ID Yearly Income Consulting Time Amount Due Low Income Agent Name Rate

1. For each error-free row of input in the data file, output the following to an **output** **file,** name of your choice, entered as requested, (in addition to the console) in a neat, readable format. Output each input line on one output line. Use manipulators to output values in readable columns. Use dollar signs before monetary values. All monetary values must have 2 decimal points. Use headings to indicate values (you DO NOT need to use the headings below, you may use your own):

Customer Name (initial and last name)

Customer ID

Yearly Income

Consulting time (in minutes)

Amount Due (customer’s bill)

Low Income (Y or N indicator)

Agent Name (consulting agent)

Agent’s Hourly Rate

Output must be in the following form, i.e., a report header and one heading for all lines for output:

Billing Report

Customer Name Customer ID Yearly Income Consulting Time Amount Due Low Income Agent Name Rate

You MAY use abbreviations for headers to fit across the screen.

1. **Error Handling**: There will be **errors** in the data file (see below for possible errors). If an error in the data is encountered, write the nature of the error and the customer name (initial and last name) and customer id containing the error to a **second output file**, name of your choice (not the console). **Do Not Process the Data**; continue with the next input line. Only log the first error found in each line/row of data.

You MAY hardcode (use a literal/constant) the name of the output error file.

Possible errors:

* Valid agent IDs are 1, 2 and 3. Any other agent ID is an error.
* The yearly income and consulting minutes must be greater than zero. Check both the yearly income and consulting minutes in your program.

1. Open all necessary data files and check the state of **all** file opens. If any file does not open properly, display an appropriate message and exit the program.
2. CLOSE ALL files before exiting the program.
3. Extra credit opportunity

During lectures, I discussed checking file status as follows:

if (!infile) …

Using the C++ resource link provided in Canvas or other C++ resource, find and use an alternative method of testing to see if the file is open. This could be the use of a function, testing of a value, etc. Be prepared to discuss your alternate method in class. One (1) point extra credit for using an alternate method.

1. Include the following documentation. Points will be deducted if you do not include the following documentation:

The name of your C++ file

Your name

Some kind of date, either the due date or the date you finished

The type of input

The type of output

A brief description of the algorithm/program

**Test and evaluate calculations for accuracy. Points will be taken for inaccurate calculations, improper formatting, directions not followed, no looping construct, no switch, etc.**

Turn in file code \*.cpp; README file; input file (even though provided) and ALL output files. You DO NOT need to submit the executable (\*.exe). You MAY zip all the files and submit if you choose. You MUST include all the files indicated or points will be deducted.

README must contain instructions for location of input/output files

Each solution is to be uniquely your own; no student or outside collaboration allowed.

**Points WILL BE taken if MINIMUM requirements and submissions not included.**

**I’ll award 1 point extra credit if the submission files are zipped and the zip file is submitted.**

See Canvas and Syllabus for due date