```
/*
   File: prog6_a_l523.cpp

   Author: Aaron Luna
   C.S.1428.007
   Lab Section: L17
   Program: #6
   Due Date: 11/23/20

   This program prints to an output file a title and column headers
for a
   payroll report. It then reads an employee's work record from an
input file.
   The number of overtime hours worked, gross pay, state tax, federal
tax,
   and net pay are calculated for each employee. The author's personal
   identification information followed by the company payroll report
is
   printed to the output file. Monetary values are displayed to two
decimal
   places.

   An attempt to avoid repetitive code is made.

   An appropriate message is displayed to the console screen if either
the
   input or the output file fails to open.

   An appropriate message is written to the console screen informing
the
   user of the output file name to which the results have been
written.

   The client file (main) calls the following void functions to
process
   the data:

      - printIdInfo prints the author's personal information - name,
        class/section number, lab section number, due date - on the
first,
        second, third and fourth lines. Two blank lines are left after
the
        due date. The integer lecture section number is displayed in a
        three-digit field with leading zeros shown. The calling
routine
        determines the output stream to which the information is
directed.
        (Refer to sample output below.)
      - printReportHeadings prints to the output file the title &
column
```

headings for the payroll report. One blank line is left after the centered report title. Column headers are displayed on two lines. (Refer to sample output below.)

- dataIn reads the employee ID#, hours worked and pay rate from an input file storing the values read into parallel arrays. The employee ID# is stored in a 1D array of integers. The hours worked and the pay rate are stored in the first and second columns of a 2D array of real numbers.
- overTime calculates the number of overtime hours worked by the employee based on a 40 hour work week.
- grossPay calculates the employee's gross pay for the week. If the employee worked 40 hours or less, gross pay is the hourly pay rate multiplied by the number of hours worked; otherwise, the employee worked more than 40 hours, and they are paid the regular hourly rate for the first 40 hours plus time and a half for any hours over 40.
- stateTax calculates state taxes owed by the employee, which is calculated at a straight 6% of the employee's weekly gross pay. (6% is a sample tax rate. The tax rate will vary with each state.)
- federalTax calculates federal taxes owed by the employee. If weekly gross pay is $400.00 or less, the tax rate is 20%; otherwise, the employee's weekly gross pay is more than $400.00, and the tax rate is calculated at 20% for the first $400.00 and 31% for any amount over $400.00. (These rates will vary based on current federal tax regulations.)
- netPay calculates the employee's net pay for the week. (gross pay minus calculated taxes both state & federal).
- printReportData prints to the output file the information for each employee in tabular form. Monetary values are displayed to two digits of precision. (Refer to sample output below.)
- writeFileLocation prints an appropriate message to the console screen indicating to the user the name of the output file to which the results have been written. The output file name is

```
provided by
        the calling routine.(Refer to sample output below.)

   The following named constants are declared globally:
      - the number of hours in a normal work week (40.00)
      - the state tax rate (0.06)
      - the federal tax rates (0.20; 0.31)
      - the cut off at which federal taxes increase (400.00)
      - parallel array dimensions
      - names used to reference individual columns in the payroll
array

   Locally declared named constants include:
      - a string to hold the author's name
      - a string to hold the author's two-digit lab section number
      - a string to hold the project's due date
      - an integer to hold the three-digit lecture section number
      - an integer representing the maximum string length allowed for
input and
        output file names which are stored in character arrays of that
length


================================================================================

   Layout and content of the output are shown in the samples below.

   Input (file - prog6_?inp.txt) // '?' represents three-digit lecture
section #
          one record per employee / each record containing three
numbers

          ID#(integer)  hours worked (real)  pay rate (real)
          ...

   Constants: globally declared:
                integer: ROWS
                         COLS
                         {2D array column indices)
                             HRS_WRKD = 0,
                             PAYRATE = 1,
                             OVRTIME = 2,
                             GROSS = 3,
                             ST_TAX = 4,
                             FED_TAX = 5,
                             NETPAY = 6;

                double: CUT_OFF (hours in a work week)
                        STATE_TX_RATE
```

```
                          TAX_CUT_OFF (division @ which net pay is
taxed higher)
                          LOW_TAX_RATE
                          HI_TAX_RATE

   Constants: locally declared:
                 string:  AUTHOR
                          LAB_SECTION
                          DUE_DATE

                 integer: LECTURE_SECTION
                          MAX_LENGTH_FN = ?  // filename's maximum
length

   Output (console) - Sample Console Output:

   Author's Name
   C.S.1428.?        // '?' represents three-digit lecture section #
   Lab Section: L?   // '?' represents two-digit lab section number
   --/--/--          // dashes represent due date, month/day/year
        <blank line>
        <blank line>
   Program results have been written to prog6_?out.txt.

   Output (file: prog6_?out.txt): // '?' represents three-digit
lecture sec #
   Sample File Output

   Author's Name
   C.S.1428.?        // '?' represents three-digit lecture section #
   Lab Section: L?   // '?' represents two-digit lab section number
   --/--/--          // dashes represent due date, month/day/year
        <blank line>
        <blank line>
                        Monthly Payroll Report
        <blank line>
    ID#     Hours    Hourly    Overtime   Gross    State   Federal
Net
          Worked     Rate       Hours     Pay      Tax      Tax
Pay
   1000     51.00      6.55      11.00   370.07    22.20    74.02
273.86
   ...
   1002     26.00     15.00       0.00   390.00    23.40    78.00
288.60
   ...


================================================================
=
```

```
   <Output will vary based on actual input values.>
*/

#include <iostream>
#include <fstream>
#include <iomanip>
#include <cstdlib>     // 4 Code::Blocks

using namespace std;

const int ROWS = 10,    // number of employees
          COLS = 7;

          // 2D array (payroll) column indices
const int HRS_WRKD = 0,
          PAYRATE = 1,
          OVRTIME = 2,
          GROSS = 3,
          ST_TAX = 4,
          FED_TAX = 5,
          NETPAY = 6;

const double CUT_OFF = 40.00,      // work week
             STATE_TX_RATE = 0.06,
             TAX_CUT_OFF = 400.00, // earnings after which taxed at
higher rate
             LOW_TAX_RATE = 0.20,
             HI_TAX_RATE = 0.31;

void printIdInfo( ostream &out, const string AUTHOR, const int
LECTURE_SECTION,
                  const string LAB_SECTION, const string DUE_DATE ),
     printReportHeadings ( ofstream &fout ),
     dataIn ( ifstream &fin, int employee[], double payroll[][COLS] ),
     overTime ( double payroll[][COLS] ),
     grossPay ( double payroll[][COLS] ),
     stateTax ( double payroll[][COLS] ),
     federalTax ( double payroll[][COLS] ),
     netPay ( double payroll[][COLS] ),
     printReportData ( ofstream &fout, int employee[], double
payroll[][COLS] ),
     writeFileLocation ( char [] );

int main ( )
{
    const string AUTHOR = "Aaron Luna",
                 LAB_SECTION = "L17",
                 DUE_DATE = "11/23/20";
    const int LECTURE_SECTION = 007,
```

```
          MAX_LENGTH_FN = 20;

    char input_filename[MAX_LENGTH_FN + 1] =  "prog6_007inp.txt" ,
         output_filename[MAX_LENGTH_FN + 1] =  "prog6_007out.txt" ;

    int employee[ROWS];          // employee ID#s

    double payroll[ROWS][COLS];  // payroll data

    ifstream fin;
    fin.open( input_filename );

    if ( !fin )
    {
        cout << endl << endl
             << "***Program Terminated.***" << endl << endl
             << "Input file failed to open." << endl;

        system("PAUSE>NUL");

        return 1;
    }

    ofstream fout;
    fout.open( output_filename );

    if ( !fout )
    {
        cout << endl << endl
             << "***Program Terminated.***" << endl << endl
             << "Output file failed to open." << endl;

        fin.close();

        system("PAUSE>NUL");

        return 2;
    }

    printIdInfo ( fout, AUTHOR, LECTURE_SECTION, LAB_SECTION,
DUE_DATE );
    printReportHeadings ( fout );
    dataIn ( fin, employee, payroll );
    overTime ( payroll );
    grossPay ( payroll );
    stateTax ( payroll );
    federalTax ( payroll );
    netPay ( payroll );
    printReportData ( fout, employee, payroll );
    printIdInfo( cout, AUTHOR, LECTURE_SECTION, LAB_SECTION,
```

```
DUE_DATE );
    writeFileLocation ( output_filename );

    fin.close();
    fout.close();

    system("PAUSE>NUL");

    return 0;
}

/*
    Function: printIdInfo

    The void function printIdInfo prints the author's name, class &
lecture
    section number, lab section number plus program due date on
separate
    lines. Two blank lines are left after the due date. setfill and
setw are
    used to assure the lecture section is displayed in a three-digit
field with
    leading zeros shown. The calling routine determines the output
stream to
    which the information is directed.

    Sample Output:
    Author's Name
    C.S.1428.?          // '?' represents author's three-digit lecture
sec #
    Lab Section: L?     // '?' represents author's two-digit lab sec #
    --/--/--            // dashes represent due date, month/day/year
          <blank line>
          <blank line>

    Receives: output file variable,
              const string containing author's name,
              const int containing a three-digit lecture section
number,
              const string containing a two-digit lab section number,
              const string containing the due date;
                    (in this order)
    Constants: AUTHOR (string)
               LECTURE_SECTION (integer)
               LAB_SECTION (string)
               DUE_DATE (string)
    Returns: nothing; prints author's personal information
*/

void printIdInfo ( ostream &out, const string AUTHOR, const int
```

```
LECTURE_SECTION,
                  const string LAB_SECTION, const string DUE_DATE )
{
    out << AUTHOR << endl << "C.S.1428." << setw(3) << setfill('0')
        << LECTURE_SECTION << endl << "Lab Section: L" << LAB_SECTION
        << endl << DUE_DATE << endl << endl << endl;
}

/*
    Function: printReportHeadings

    The void function printReportHeadings prints the title and column
headers
    for the monthly payroll report to an output file. One blank line
is left
    after the centered report title. Column headers are displayed on
two lines
    as shown in the sample below.

                         Monthly Payroll Report
                             <blank line>
     ID#      Hours   Hourly   Overtime    Gross     State    Federal
Net
             Worked    Rate      Hours      Pay       Tax       Tax
Pay

    Receives: output file variable
    Constants: none
    Returns: nothing; prints payroll report title/column headers for
the
                      monthly payroll report
*/

void printReportHeadings ( ofstream &fout )
{
    fout << "           Monthly Payroll Report " << endl << endl;
    fout << "ID#     Hours   Hourly   Overtime    Gross    State
Federal    Net"
         << endl;
    fout << "       Worked    Rate     Hours      Pay      Tax
Tax      Pay"
         << endl;
}

/*
    Function: dataIn

    The void function dataIn reads the employee ID#, hours worked, and
pay
    rate from an input file storing the values read into parallel
```

```
arrays.
    The employee ID# will be stored in a 1D array. The hours worked
and the
    pay rate will be stored in the first and second columns of a 2D
array.

    Receives:  input file variable
               1D array of integer ID#s
               2D array of reals (payroll information) / COLS
               (in this order)
    Constants: globally declared integers:
                  ROWS - parallel arrays row dimension
                  COLS - 2D array column dimension
                  HRS_WRKD - column designation in 2D array
                  PAYRATE - column designation in 2D array
    Returns:   fills the 1D array of employee ID#s plus the hours
worked and
               payrate columns of the 2D array are filled with data
read from
               the input file
*/

void dataIn ( ifstream &fin, int employee[], double payroll[][COLS] )
{
    for ( int i=0; i < ROWS; i++)
    {
        fin >> employee[i];
        fin >> payroll[i][HRS_WRKD];
        fin >> payroll[i][PAYRATE];
    }
}

/*
    Function: overTime

    The void function overTime calculates the number of overtime hours
worked
    by the employee based on a 40 hour work week.

    If the employee worked 40 hours or less,
           overtime = 0.0
    otherwise  // employee worked > 40 hours earning overtime
           overtime = hours worked - 40

    Receives: 2D array of reals (payroll information) / COLS
    Constants: globally declared integers:
                  ROWS - parallel arrays row dimension
                  COLS - 2D array column dimension
                  HRS_WRKD - column designation in 2D array
                  OVRTIME - column designation in 2D array
```

```
               globally declared reals:
                  CUT_OFF (hours in a work week)
    Returns: fills the overtime column in the 2D array with calculated
data
*/

void overTime ( double payroll[][COLS] )
{
    for ( int i=0; i < ROWS; i++ )
    {
        if ( payroll[i][HRS_WRKD] > CUT_OFF )

            payroll[i][OVRTIME] = payroll[i][HRS_WRKD] - CUT_OFF;

        else

            payroll[i][OVRTIME] = 0;
    }
}

/*
    Function: grossPay

    The void function grossPay calculates the employees gross pay for
the week.

    If the employee worked 40 hours or less,
           gross pay is the hourly pay rate multiplied by the number
of hours
           worked
    otherwise     // employee worked > 40 hours
           employee is paid the regular hourly rate for the first 40
hours plus
           time and a half for any hours over 40.

    Receives: 2D array of reals (payroll information) / COLS
    Constants: globally declared integers:
                  ROWS - parallel arrays row dimension
                  COLS - 2D array column dimension
                  column designations in 2D array
                       GROSS, HRS_WRKD, OVRTIME, PAYRATE
               globally declared reals:
                  CUT_OFF (hours in a work week)
    Returns: fills the gross pay column in the 2D array with
calculated data
*/

void grossPay ( double payroll[][COLS] )
{
    for ( int i=0; i < ROWS; i++ )
```

```
    {
        if ( payroll[i][OVRTIME] == 0 )

            payroll[i][GROSS] = payroll[i][PAYRATE] * payroll[i]
[HRS_WRKD];

        else

            payroll[i][GROSS] = ( payroll[i][PAYRATE] * 1.5 *
payroll[i][OVRTIME] )
            + ( payroll[i][PAYRATE] * CUT_OFF );
    }
}

/*
    Function: stateTax

    The void function stateTax calculates the state taxes owed by the
employee
    calculated at a straight percentage of the employee's weekly gross
pay.

    Receives: 2D array of reals (payroll information) / COLS
    Constants: globally declared integers:
                  ROWS - parallel arrays row dimension
                  COLS - 2D array column dimension
                  column designations in 2D array
                     GROSS, ST_TAX
               globally declared reals:
                  STATE_TX_RATE, real (globally declared)
     Returns: fills the state tax column in the 2D array with
calculated data
*/

void stateTax ( double payroll[][COLS] )
{
    for ( int i=0; i < ROWS; i++ )
    {
        payroll[i][ST_TAX] = STATE_TX_RATE * payroll[i][GROSS];
    }
}

/*
    Function: federalTax

    The void function federalTax calculates federal taxes owed by the
employee.

    Note: The sample below assumes a tax cut-off of 400.00 and a tax
rate for
```

```
          20% for low wage earners and a tax rate of 31% for high wage
earners.
          Actual values may differ.

    If weekly gross pay is $400.00 or less,
           tax rate is 20%.
    otherwise   // employee's weekly gross pay is > $400.00
           tax rate is calculated at 20% for the first $400.00
           and 31% for any amount over $400.00.

    Receives: 2D array of reals (payroll information) / COLS
    Constants: globally declared integers:
                  ROWS - parallel arrays row dimension
                  COLS - 2D array column dimension
                  column designations in 2D array
                     GROSS, FED_TAX
               globally declared reals:
                  TAX_CUT_OFF - income level at which taxes are
increased
                  LOW_TAX_RATE
                  HI_TAX_RATE
    Returns:   fills the federal tax column with calculated data
*/

void federalTax ( double payroll[][COLS] )
{
    for ( int i=0; i < ROWS; i++ )
    {
        if ( TAX_CUT_OFF >= payroll[i][GROSS] )

            payroll[i][FED_TAX] = payroll[i][GROSS] * LOW_TAX_RATE;

        else
            payroll[i][FED_TAX] = ( payroll[i][GROSS] - TAX_CUT_OFF )
* HI_TAX_RATE
            + TAX_CUT_OFF * LOW_TAX_RATE;
    }
}

/*
    Function: netPay

    The void function netPay calculates the employee's net pay for the
week
    - gross pay minus calculated taxes (state and federal).

    Receives: 2D array of reals (payroll information) / COLS
    Constants: globally declared integers:
                  ROWS - parallel arrays row dimension
                  COLS - 2D array column dimension
```

```
                   column designations in 2D array
                      GROSS, ST_TAX, FED_TAX
    Returns:   fills the net pay column with calculated data
*/

void netPay ( double payroll[][COLS] )
{
    for ( int i=0; i < ROWS; i++ )
    {
        payroll[i][NETPAY] = payroll[i][GROSS] - ( payroll[i][FED_TAX]
                                                   + payroll[i]
[ST_TAX]);
    }
}

/*
    Function: printReportData

    The void function printReportData prints to an output file the
payroll
    information for each employee in tabular form under the
appropriate column
    headers. The employee records are single spaced. Monetary values
are
    displayed with two digits of precision.

    Sample Tabular Data:

    Note: The report title and column headers are printed by a
previously
          called function.

                         Monthly Payroll Report

     ID#      Hours   Hourly   Overtime    Gross     State    Federal
Net
             Worked    Rate      Hours      Pay       Tax       Tax
Pay
    1000      51.00     6.55     11.00    370.07     22.20     74.02
273.86
    1001      40.50     6.50      0.50    264.88     15.89     52.98
196.01
    ...

    Receives:  output file variable
               protected 1D array of integer ID#s
               protected 2D array of reals (payroll information) /
COLS
               (in this order)
    Constants: globally declared integers:
```

```
                    ROWS - parallel arrays row dimension
                    COLS - 2D array column dimension
    Returns: nothing; prints out report data under appropriate report
headings
             previously printed
*/

void printReportData ( ofstream &fout, int employee[], double
payroll[][COLS] )
{
    fout << fixed << setprecision(2) << setfill(' ');

    for ( int i=0; i < ROWS; i++ )
    {
        fout << employee[i];

        for ( int k=0; k < COLS; k++)
        fout << setw(10) << payroll[i][k] << endl;
    }
}

/*
    Function: writeFileLocation

    The void function writeFileLocation writes the message below to
the screen
    indicating to the user the name of the output file to which the
results
    have been written. The output filename is provided by the calling
routine.

    Output:
    "Program results have been written to <filename here>."

    Receives: protected 1D array of characters - output file name
    Constants: 1D array of characters representing the output file
name
    Returns: nothing; writes a message containing the output file name
to the
             screen
*/

void writeFileLocation ( char output_filename[] )
{
    cout << " Program Results have been written to " <<
output_filename
         << ". " << endl;

}
```