Broome Community College
CST 113

Lab 6
Conditional Testing – If, If/Else Nested Ifs, Switch
Repetition – for, while, repeat
Introduction to Output Functions

Lab Objectives:

- Use of nested ifs & switch conditionals to make decisions in your program
- Use of loop processing for repetitive tasks
- Use of input and output files
- Use of string functions
- Managing input by using the **`ignore()`** function

Materials:

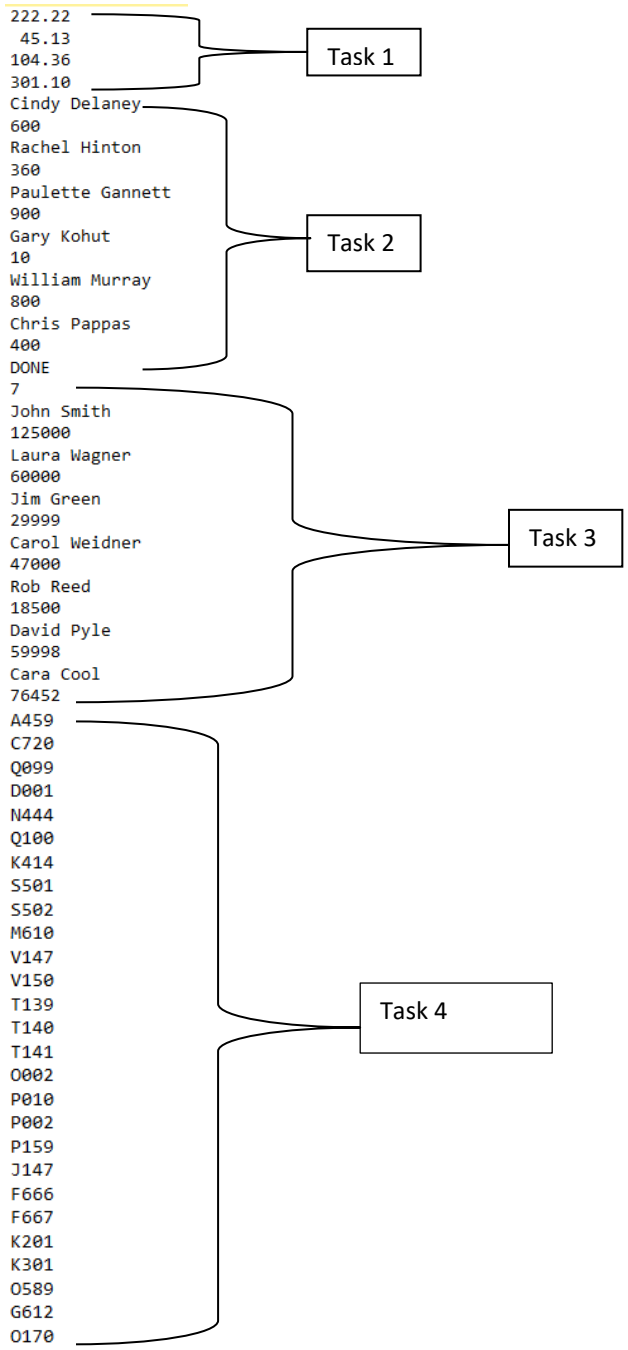
1. Mod4 notes and example
   A. These files can be found in Module 4
   B. The ***CondTestExampl***e will be very helpful for this lab
   C. The ***LoopExample*** will also be very helpful for this lab
2. **Mixed File Input Example**
   A. This example can be found in Module 3
3. **Mod2UserInput Example**
   A. This example can be found in the Module 2
4. Grading and Documentation
   B. The files below are found in the same location in Blackboard where you found these instructions Content/Spring Labs/Lab 6
      i. Grading Block & Doc Block file – **Lab6GrdBlkAndIPOStart.txt**
      ii. The three input files

Introduction:

Problem: In an input file there is data for distinct tasks. Unlike previous lab Labs, the tasks (except Task 0), are not related or dependent on each other. Each task 's solution will be limited to particular types of basic conditional statements. For this lab, you can use the conditionals expressions that you have learned BUT it some cases you are directed to solve the problem with a specific type of conditional. This restriction is designed to provide students will an opportunity to gain a clear understanding of that type conditional.

Files

1. The three input files are: **`Lab6In-1.txt`**, **`Lab6In-2.txt & Lab6In-3.txt`** are provided and can be downloaded from Blackboard
2. Make sure you copy the screen output and paste it into a file named: **`Lab6ScreenOut.txt`**
   A. This should just be the output from running the program and selecting 1 for the file number
3. The physical output file should be named to match the number of the input file: **`Lab6Out-1.txt`** or **`Lab6Out-2.txt`**
4. Name the Solution **and** the CPP file: **`YourLastNameFirstInitialLab6`**
   A. If your name is John Smith, the two files should both be named: **SmithJLab6**
5. Make sure ALL the files listed above are correctly attached to the solution.
   A. Three input files are provided to test the program. You should run your program with ALL files to assure that it works completely correct. Explanation of the tasks associated with the input file data in **`Lab6In-1.txt`** is shown below:



Instructions:

General Notes

1. For this lab, you are going to solve several small tasks. Each task will be added to ONE project and submitted as ONE zip file. You will get partial credit for each task that you solve correctly. You must solve task 0 correctly to be eligible for a grade on other tasks.
   A. **All** output (except for the file selection task) will go to one of two output files which should be named: **Lab6Out-1.txt, Lab6Out-2, & Lab6Out-2**
      i. **Lab6Out-1.txt** will be opened when **Lab6In-1.txt** is selected
      ii. **Lab6Out-2.txt** will be opened when **Lab6In-2.txt** is selected
      iii. **Lab6Out-3.txt** will be opened when **Lab6In-3.txt** is selected
   B. All output should be clearly labeled

2. Create a new project as usual. Copy the Grading Block and doc block to the top of your empty .CPP file

3. Review the grading block
   A. Read the grading block file carefully so you understand the criteria that will be used to determine your grade on the Lab
   B. For example: Find out what mistakes will result in an automatic zero grade being recorded for the Lab

Preprocessor Directives

4. Create all necessary preprocessor directives in the correct location
   A. You will need the usual preprocessor directives for input/output, format manipulators, and strings
   B. You will need a special preprocessor directive for the use of data files
      i. See the examples in the **Mod3B** notes

Constants

5. Declare constants for all constant values
   A. Group constants by the task, putting //Task N above the blocked group of constants.
      i. Where N is the task number
   B. Use the sample screen shot to help you understand how to group the constants, this is just a partial list and does not include all the constants for Task 2

```
//Task 1 - Grocery Total
const int WEEKS_IN_MONTH = 4;

//Task 2 - Professor Ranking
const string HIGHEST_RANK = "Professor";
```

Variables

6. Make sure the variables have meaningful names for each task.
   A. **Do not re-use variables** from one task to the next.
   B. Unlike other labs, we are going to group variables by the task first AND then by TYPE instead of grouping by type only.
      i. This should help avoid confusion as to what variables belong to which task in the declarations
      ii. Put a comment above the variable block for each task stating the task number
      iii. See the example below:

```
//Task 0 & Task Messages
ifstream fin;
ofstream fout;

int fileNumber;

string inputFileString;
string outputFileString;
string taskHeading;

int stringLength;
```

Program organization

7. Above each task in the **`main`**, include a comment, listing the task # and a brief description of the task.
   A. Example:

```
//-----------------------------------------------------------------------------------------------
//Task 1 - Grocery Total & Points & Gas Discount Awarded
```

   B. Solve one task at a time, compile and run the program and make sure it works correctly for ALL input files before moving on to the next task.
   C. If you cannot get a task to work correctly, you should edit the input file and remove ONLY the lines of data for the task you cannot get to work correctly

Conditional Formats for Nested If and Switch

(Nested if – any number of conditionals but the last is typically the default)
(See Mod notes and examples for more details)

```
if (conditional)
{
     //Statements
}
else if (conditional)
{
     //Statements
}
else if(conditional)
(
     //Statements
}
else
{
     //Statements
}
```

Any number of else if (conditional)

Default – catches all cases that are not met by the conditionals above

Switch – REMEMBER to adjust the indentation to meet this requirement – VS will indented the code differently

```
switch (expression)
{
     case constant-expression:
          //Statements
          break;
     case constant-expression:
     case constant-expression:
          //Statements
          break;
     case constant-expression:
     case constant-expression:
     case constant-expression:
          //Statements
          break;
     default:
          //Statements
}
```

Any number of cases, each ends with a break

The default should "catch" all constant-expression values that do not meet one of the above cases

Output

8. Output considerations
   A. Remember to desk check your program's calculated output with a calculator
      i. Programs that produce incorrect output will be returned with a grade of zero
9. Output the course heading CENTERED to an output file
   A. This is the same heading that was output for the previous Labs, centered on the length of the divider above and below the heading data to the output file
   B. This includes the 4 string constants created for the header
      i. You should have this section of the code mastered by this time
10. Output a divider between each task in your output file & output two blank lines between each task to block them
11. Include an output statement which includes the task number and a brief one line identifying description
    A. Each Task has an example of an appropriate output statement for the task number
    B. DO NOT put this output statement inside any conditionals or loop section of the code, it should be printed immediately AFTER any input and before any calculations or decisions.
12. All floating-point output should be formatted to 2 decimal places unless otherwise specified
13. Make sure the output is formatted as shown in the sample outputs for each task
    A. Center ALL headings
    B. Align data in columns when possible
    C. Make sure the output divider spans the width of the output as shown in the samples
14. Use the sample output provided in each task to verify your results using the data from the first input file
15. REMEMBER to use **`setw`**, **`left`** and **`right`** justification to format the output correctly!

Output Functions

16. In programming, functions are used to reduce the duplication of code and to "declutter the program"
    A. The first function that we are going to include in this program is called **PrintDivider**
        i. This function does exactly what you would expect and we use it repeatedly
    B. Carefully follow these directions to add this function to your program
        i. Step 1: Add the prototype
            1) This introduces the function to your program and should be placed ABOVE the main and BELOW the constants
            2) This prototype identifies the name and the parameters – arguments for the function
                A) parameters are what is passed into the function
            3) The prototype for PrintDivider is:

```
//Prototype for PrintDivider
void PrintDivider(ofstream& fout, char symbol, int numberOfSymbols);
```

                A) The first parameter is the output file name, this can be whatever logical name you created for your program
                    i) The data type is listed first and for the first parameter it is ofstream&
                        (1) The & is important for files when using them in functions
                B) The second parameter is the character that you want to output
                    i) The data type is listed first and for the second parameter it is char
                C) The third parameter is the number of symbols that you want to print
                    i) The data type is listed first and for the third parameter it is int
        ii. Step 2: The actual function code, the function definition
            1) This is the code that will execute9 when the function is called
            2) The code belongs BELOW the main after return 0 and the ending brace
            3) The function doc block and the code is:

```
//-----------------------------------------------------------------------------------------------
//PrintDivider - Output a divider to the specified file, # of symbols passed
//-----------------------------------------------------------------------------------------------
void PrintDivider(ofstream& fout, char symbol, int numberOfSymbols)
{
    fout << setfill(symbol) << setw(numberOfSymbols + 1) << ' ' << setfill(' ') << endl;
}
```

        iii. Step 3: Calling the function in the main where needed
            1) You will use this function to output a divider to either the screen or the output file
                A) A call to output the divider to the screen could look like this:

```
//Output the divider to the screen
PrintDivider(static_cast<ofstream&>(cout), '-', 55);
```

                    i) The first parameter in the call is `static_cast<ofstream&>(cout)`
                        (1) This will cast the screen output to `ofstream&` so it can be passed into the function
                    ii) The second parameter in the call is `'-'` which is the character that will be printed across for the divider
                    iii) The third parameter in the call is 55 which is the number of symbols that you want printed
                B) In the call the function name must be the same name as in the prototype and the definition
                C) AND the parameter values in the call must match in number and type by position
                    i) There are 3 parameters in the prototype and definition, there must be 3 parameters in the call
                    ii) The first parameter is an output file, the second a char and the third an int
                D) A call to output the divider to the opened logical output file, **`fout`** could look like this:

```
//Output the divider to the output file
PrintDivider(fout, '-', 60);
```

        E) Each time you want to output a divider, call this function with any character and any number of characters

TASKS:

1. **TASK 0 - Opening the input & Output Files**
   A. Display on the screen three options for the name of the possible input file. The user will enter a 1, 2 or 3. If the user enters a 1, assign the input file string, **Lab6In-1.txt** and the output file string is assigned **Lab6Out-1.txt** and if the user enters a 2 the input file string is assigned **Lab6In-2.txt** and the output file string is assigned **Lab6Out-2.txt** otherwise the input file string is assigned the third input and output file names

   B. This procedure is like what you did in Lab 5, except this time we want you to assign the name of the name of input file and the name of the output file to a string variables inside the conditional statement, and then open the files and print the messages **outside** of the conditional

   C. After the user selects a number, display the name of the file along with a message that states the file was opened successfully using the output divider
   D. Sample Screen Input:

```
Select the input file:
1.  Lab6-1.txt
2.  Lab6-2.txt
3.  Lab6-3.txt

Enter the input file choice: 1
--------------------------------------------------
The input file: Lab6In-1.txt is successfully opened.
The output file: Lab6Out-1.txt is successfully opened.
--------------------------------------------------
Press any key to continue . . .
```

   E. Make sure you copy and paste the screen output to a txt file named **Lab6ScreenOutput.txt** and add this file to the Solution Explorer Window
   F. The purpose of having multiple input files is to allow you to test your program with multiple sets of input data
   G. YOU MUST GET this task working correctly!

1. **TASK 1 - Grocery Total, Points Awarded & Gas Discount Set**
   A. A **FOR** loop and a **NESTED IF** should be used to accomplish this task
   B. Input the four weekly totals (four represents the number of weeks in the month) from an input file, one at a time in the for loop, accumulating each weekly total in the monthly grocery total & outputting the week number and the weekly total
   C. From the monthly grocery total, determine and set the points and the gas discount percent that the grocery store awards based on the following table
      i. REMEMBER – you should not have a compound conditional test in the nested if
         1) Example: else if (number >100 && number < 500)
         2) Review the notes on nested ifs if this is not clear

Monthly Total Greater Than	Points Awarded	Gas Discount per gallon
1500.00	500	0.30
900.00	430	0.25
675.00	200	0.15
480.00	180	0.10
300.00	150	0.04
0.00	100	0.02
(No Purchases)	0	0.00

   D. Output the monthly total, the points awarded, and the gas discount clearly labeled and formatted
   E. Do NOT create constants for the values in this task
      i. It would be an excessive number of constants that would be challenging to name them appropriately!

   F. Sample Output:
      i. (Notice the use of the output divider and the output of the gas discount with the cent sign- ¢ displayed after the gas discount)

```
------------------------------------------------------------
             Task 1 - Grocery Points & Discount
------------------------------------------------------------
Week #                                          Weekly Total
------------------------------------------------------------
     1                                              222.22
     2                                               45.13
     3                                              104.36
     4                                              301.10
------------------------------------------------------------
The total groceries spent for the month are:        672.81
------------------------------------------------------------
The total points awarded are: 180
Gas Discount per gallon awarded is : 10¢
------------------------------------------------------------
```

      ii. Notice the output of the task heading, the column heading, the week numbers and the weekly totals
         1) The Week number should be the LCV of the for loop
         2) The Weekly totals are the input value formatted to display 2 decimals
         3) The Monthly total is an ACCUMULATED value and formatted to display 2 decimals
         4) The Total points is an integer determined by testing the Monthly total
         5) The gas discount is a double determined by the same testing the Monthly total
            A) This value is output to display with 0 decimals and the cent sign- ¢
               i) To accomplish this:
                  (1) Change the display of floating point values to 0 for the output file
                  (2) Multiple the set gas discount by 100
                  (3) Output this calculated value
                  (4) Output `static_cast<char>(unsigned(162))`
                  (A) 162 is the ASCII value of the cent sign, casting it to a char will output the character at that ASCII value

2. **TASK 2 - William & Chris University – Professional Ranking**
   A. Use a **SENTINEL WHILE** loop & a **SWITCH** statement to accomplish this task
   B. Input the professor's name and total professional points earned while the professor's name is not DONE
   C. From the professional points awarded, determine and set the ranking for each professor

Total Professional Points Earned Greater or Equal to	Ranking
800	Professor
600	Associate
400	Assistant
Less than 400	Lecturer

   D. Create appropriate constants
   E. 1000 is the maximum points ever awarded at the University
      i. Set the ranking in the **`switch`** statement
         1) HINT: 300/100 = 3, 310/100 = 3, 399/100 = 3
   F. MAKE sure you format the **SWITCH** statement as required for this course (it is different than the default indenting produced by the Visual Studio editor)
   G. Output a heading for the University, the instructor's name, total professional points completed, and the ranking
      i. Do NOT put the output in the **`switch`** statement
   H. Sample Output:

```
------------------------------------------------------------
               Task 2 - Professional Ranking
------------------------------------------------------------
                William and Chris University
------------------------------------------------------------
Professor's Name                  Points             Ranking
------------------------------------------------------------
Cindy Delaney                        600           Associate
Rachel Hinton                        360            Lecturer
Paulette Gannett                     900           Professor
Gary Kohut                            10            Lecturer
William Murray                       800           Professor
Chris Pappas                         400           Assistant
------------------------------------------------------------
```

3. **Task 3 - Employee Pay**
   A. Use a FOR loop & a **NESTED IF** statement
   B. In an input file, there is the number of employees, followed by the names and yearly salary of the number of employees of the C++ Programming Group Input the number of employees and the employee's name and the gross pay of each employee. Determine & set the tax bracket & the tax rate based on gross pay, using the table below:
      i. Create appropriate constants

Salary	Tax Bracket	Tax Rate
Below 30,000	1	15%
30,000 – 59,999	2	22%
60,000 – 99,999	3	33%
100,000 & above	4	40%

      ii. It may be easier to start the nested if using the highest range and working through the conditionals from highest to lowest
      iii. REMEMBER – you should <u>not</u> have a compound conditional test in the nested if
         1) Example: else if (number >100 && number < 500)
         2) Review the notes on nested ifs if this is not clear
   C. Calculate the taxes
      i. The tax rate times the salary
   D. Calculate the net pay
      i. The gross pay minus the taxes
   E. Output the company name (centered), the employee name, the tax bracket, the gross pay, the tax rate, the taxes, and the net pay aligned as shown in the sample output:

```
-------------------------------------------------------------------------------------------
                                Task 3 - Employee Pay
-------------------------------------------------------------------------------------------
                                C++ Programming Group
-------------------------------------------------------------------------------------------
Name                    Tax Bracket      Gross Pay       Tax Rate          Taxes        Net Pay
-------------------------------------------------------------------------------------------
John Smith                        4      125000.00            40%       50000.00       75000.00
Laura Wagner                      3       60000.00            40%       24000.00       36000.00
Jim Green                         1       29999.00            15%        4499.85       25499.15
Carol Weidner                     2       47000.00            22%       10340.00       36660.00
Rob Reed                          1       18500.00            15%        2775.00       15725.00
David Pyle                        2       59998.00            22%       13199.56       46798.44
Cara Cool                         3       76452.00            22%       16819.44       59632.56
-------------------------------------------------------------------------------------------
```

4. **TASK 4 - BCC Sports Stadium Entrance**
   A. Use a **WHILE** loop & **SWITCH** statement along with the table to determine and set the Gate number
   B. While there are string tickets in the input file, determine the gate number where the fan should enter the stadium.
      i. Example: A459
      ii. Remember to use a primer and a changer for file processing
   C. As you enter the stadium, your ticket is scanned and you are directed to a gate based on the ticket number's first character
      i. The first character of the ticket indicates the gate
      ii. The last three characters indicate the seat number
   D. The gate number is an alphabet capital letter and the gate is determined from the following table:

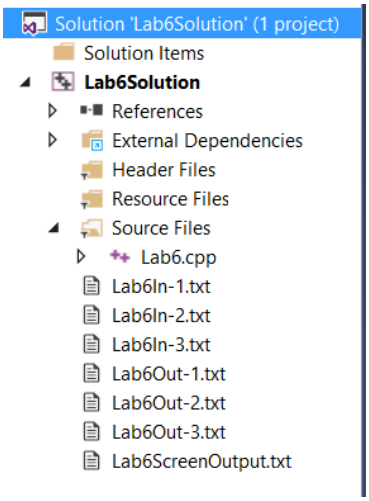
First character of the ticket number	Gate
A,E,I,M,Q,U,Y	1
B,F,J,N,R,V,Z	2
C,G,K,O,S,W	3
D,H,L,P,T,X	4

   E. REMEMBER - you have been introduced to the string method **`at()`** which allows you to access a character in a string
      i. For example: **`ticket.at(0)`**
         1) Accesses the first character in the string, **`ticket`**
   F. To access the seat number, you should look up in the text book the string method, **`substr`** which is one way to access the three-character seat number
   G. Count the number of fans that entered each gate and output this total after all tickets have been read
   H. Output the stadium name, the count of the ticket read, the ticket, the gate number, and the seat number. Followed by the counts of the number of fans that entered each gate
   I. Sample output:

```
------------------------------------------------------------
             Task 4 - Sports Stadium Seating
------------------------------------------------------------
                     BCC Sports Stadium
------------------------------------------------------------
 Fan Count         Ticket           Gate         Seat Number
------------------------------------------------------------
         1           A459              1                459
         2           C720              3                720
         3           Q099              1                099
         4           D001              4                001
         5           N444              2                444
         6           Q100              1                100
         7           K414              3                414
         8           S501              3                501
         9           S502              3                502
        10           M610              1                610
        11           V147              2                147
        12           V150              2                150
        13           T139              4                139
        14           T140              4                140
        15           T141              4                141
        16           O002              3                002
        17           P010              4                010
        18           P002              4                002
        19           P159              4                159
        20           J147              2                147
        21           F666              2                666
        22           F667              2                667
        23           K201              3                201
        24           K301              3                301
        25           O589              3                589
        26           G612              3                612
        27           O170              3                170
------------------------------------------------------------
         Gate Number           Fan Count
------------------------------------------------------------
                  1                   4
                  2                   6
                  3                  10
                  4                   7
------------------------------------------------------------
```

Advice and WARNINGS

1. IPO and Comments
   A. Use the partially completed IPO to help you organize the task solution
      i. Make sure you complete the IPO
      ii. The order of the statements in the IPO should be the order of the comments and code in the main program
   B. The IPO should be used for the commenting in the code
   C. Output of the divider is included in the IPO to clarify where it should be used
   D. For Conditionals, the commenting should be Determine WHAT based on WHAT
      i. Look at the examples of conditional commenting
   E. Loops – the commenting should start with For each… or While…
2. CAREFULLY read each task
   A. Use a nested if or switch as stated in the problem
   B. Make sure you fix the indentation of the switch from what VS does to what is required for this course
      i. See the examples
3. CONSTANTS
   A. If the problem statement contains phrases with numbers, these numbers should likely be constants
      i. For example: if the total is greater than 1000… then 1000 should be defined as an appropriately named constant
   B. Block the constants by task and within each task by type
4. VARIABLES
   A. Block the variables by task and within each task by type
5. INPUT
   A. Use a getline to input strings of more than one word
   B. Use an ignore if the previous input was standard input
6. BLOCK the code and the tasks
   A. Put a blank line between each comment and code section
   B. Put a divider comment between each task, identifying the task
      i. See the example above in these instructions
7. FORMATTING the output
   A. Use left and right to help you with formatting
      i. This is useful for right justifying the values to align with the end of the output divider
      ii. This is useful to left align the data in the left column and then right align in the right column
   B. Output two blank lines between each task's output
8. FILES
   A. Don't forget to add the input files, the output file, and the txt file with a copy of the screen output to the Solution Explorer
   B. As shown in the screen shot below:



COMPLETION

1. IPO
   A. Add the grading and main doc block above any of your code
   B. Put your name in the grading block where required
   C. Complete the IPO for each task as provided with the grading block by completing the statements
      i. You should not have to add any statements or change any statements provided
      ii. Try to organize your solution as the IPO is presented
         1) Before making any changes ASK your professor
   D. Again, The IPO statements can be used as the comments in the code with little or no editing
2. Grading Block
   E. The grading block GIVES you many hints on how to approach the solution
   F. Use this to your advantage
3. Discussion
   A. Don't forget to complete the post first and to actively participate in the discussion
4. In the project folder, delete any of these files found:
   A. Debug Folder
   B. .VC file -The Database File
   C. .vs Folder
5. Zip the project folder using the Windows Compression Utility
6. Submit the ZIPPED folder to the Assignment submission link **LAB 6** in Blackboard by the due date and time shown in the course calendar