University of Pittsburgh
Department of Electrical and Computer Engineering
ECE 0301 – Fall 2020

In-Class Assignment #5

Programming concepts:

- Functions, parameters, function prototypes, calling functions
- Overloading functions

ECE concepts:

- Boolean algebra, functions of three variables
- Decimal-to-binary conversion
- Canonical sum-of-products and product-of-sums forms
- Truth table evaluation

Background: For this assignment, you will write software for analyzing and realizing Boolean functions of three variables: $f(x,y,z)$. Recall that any such function can be specified by a truth table with $2^3 = 8$ rows. For example,

x	y	z	$f(x,y,z)$
0	0	0	0
0	0	1	1
0	1	0	1
0	1	1	0
1	0	0	1
1	0	1	0
1	1	0	0
1	1	1	1

x	y	z	$f(x,y,z)$
0	0	0	0
0	0	1	0
0	1	0	0
0	1	1	1
1	0	0	0
1	0	1	1
1	1	0	1
1	1	1	1

The eight binary values in the output column of the truth table uniquely specify the Boolean function, and therefore we can also specify the function by taking these eight bits and computing the equivalent number in the range 0-255, with the MSB in the first row and the LSB in the last row. We will refer to this number as the *function index*. For the two example functions specified above, the corresponding function indices are

Left:	01101001 = 64 + 32 + 8 + 1 = 105
Right:	00010111 = 16 + 4 + 2 + 1 = 23.

Every Boolean function can be realized in two canonical forms. The canonical sum-of-products form is obtained by finding all of the ones in the truth table, taking the minterm corresponding to each row where the function is 1, and forming the sum of the selected minterms. For the two example functions, the CSOP forms are

Left: $f_{105}(x,y,z) = \sum m(1,2,4,7) = x'y'z + x'yz' + xy'z' + xyz$

Right: $f_{23}(x,y,z) = \sum m(3,5,6,7) = x'yz + xy'z + xyz' + xyz$.

The canonical product-of-sums form is obtained by finding all of the zeros in the truth table, taking the maxterm corresponding to each row where the function is 0, and forming the product of the selected maxterms. For the example functions, we have

Left: $f_{105}(x,y,z) = \prod M(0,3,5,6) = (x+y+z)(x+y'+z')(x'+y+z')(x'+y'+z)$

Right: $f_{23}(x,y,z) = \prod M(0,1,2,4) = (x+y+z)(x+y+z')(x+y'+z)(x'+y+z)$.

Instructions: Develop software according to the following specifications and submit whatever portion you have completed to the class repository before midnight on the due date.

1. Write a computer program that will display the following introductory message to standard output:
```
ECE 0301: Boolean Functions of 3 Variables.
Realization in Canonical Forms.
```

2. Write a function that implements a three-input `OR` gate. This function takes three `bool` variables as inputs, representing *x*, *y* and *z*, and returns a `bool` value as output that is `false` if and only if all three inputs are `false`.

   In your `main` function, define an `ofstream` object, and use it to open a text file for output named "`Bool_func_3var_CSOP_CPOS.txt`". For the remainder of this assignment, all required output must be written to this file. Output that you use for temporary debugging purposes can be written to standard output if you choose.

   Create another function the writes a truth table for the three-input OR gate to the output file. This function takes an `ofstream` object for the output file as its only parameter (passed by reference), and returns nothing. The output must be formatted as a table with four columns, separate by a TAB, with headers for the inputs and output, and a line of dashes (32 dashes), as shown below:
```
X       y       z       f(x,y,z)
--------------------------------
0       0       0       0
0       0       1       1
```
   etc.

   The steps to be performed by the truth-table function are as follows:
   - Write the table headers and line of dashes to the output file.
   - Loop over the integers 0-7. For each integer:
     - Define `bool` variables for `x`, `y` and `z`. Assign values to these variables so that `xyz` is the three-bit binary representation for the loop count.

- Call your `OR`-gate function to compute the output for the current values of `x`, `y` and `z`.
- Write `x`, `y`, `z`, and the `OR` gate output to the file, separated by a TAB, so that they output values appear in columns.

Modify your `main` function so that it performs the following actions:

- Define an `ofstream` object and open the output file.
- Write the introductory message to the output file.
- Call the function to write the truth table for the `OR` gate to the output file.

3. Write a function that implements a three-input `AND` gate. This function takes three `bool` variables as inputs, representing *x*, *y* and *z*, and returns a `bool` value as output that is `true` if and only if all three inputs are `true`.

   Modify your truth-table-writing function so that it takes a `bool` variable as an additional parameter. When this parameter is false, the function must write the truth table for an `OR` gate to the output file. When the parameter is true, the function must write the truth table for an `AND` gate to the output file. Change the name of the function to reflect what it now does.

   Modify your `main` function so that it writes truth tables for the `OR` gate and the `AND` gate to the output file. The required format is summarized below, but some truth table rows have been omitted.

```
ECE 0301: Boolean Functions of 3 Variables.
Realization in Canonical Forms.

Truth table for OR gate.

x       y       z       f(x,y,z)
--------------------------------
0       0       0       0
0       0       1       1
(etc.)
1       1       1       1

Truth table for AND gate.

x       y       z       f(x,y,z)
--------------------------------
0       0       0       0
0       0       1       0
(etc.)
1       1       1       1
```

   When you are certain your program is correct, save it in a file named:

   `ece0301_ICA05_step03.cpp`

   Submit this file to the class repository.

4. Write a function that returns the expression for any minterm. This function takes an integer parameter in the range 0-7 as input, and returns a string object containing the corresponding minterm expression. For example, if this function is called with the parameter set to 1, the function will return the string `x’y’z`, and if it is called with the parameter set to 5, it will return `xy’z`. Note that, in order to determine which inputs to complement, you must determine the binary representation for the integer parameter.

   At the beginning of your `main` function, define an `ifstream` object, and use it to open a text file for input named “`Bool_func_3var.txt`”. For the remainder of this assignment, all required input must be read from this file. Use a text editor to create a file with this name, and type the number 0 as the only character on the only line in the file.

   After opening the input file, your `main` function should perform the following actions
   - Read the number from the input file and store it in an integer variable.
   - Define an `ofstream` object and open the output file.
   - Write the introductory message to the output file.
   - Call the function to return the selected minterm expression.
   - Write a line of test to report the selected minterm expression, e.g.
     ```
     m1 = x'y'z
     ```

   Test your function with values 0-7 in the input file, and confirm that the correct minterm is written to the output file each time.

   Add input validation to your function so that, if the input parameter is outside the range 0-7, it prints the error message:
   ```
   ERROR! Invalid minterm index.
   ```
   to standard output and exits the program with the failure termination code. Modify the input file, and run your program, to test the input validation.

5. Write a function that returns the expression for any maxterm. This function takes an integer parameter in the range 0-7 as input, and returns a string object containing the corresponding maxterm expression. For example, if this function is called with the parameter set to 1, the function will return the string `x+y+z’`, and if it is called with the parameter set to 5, it will return `x’+y+z’`.

   Modify your `main` function so that, after writing the minterm to the output file, it calls your maxterm function, and writes the returned expression to the output file. At this point, for any integer 0-7, your program will generate the corresponding minterm and maxterm. Test your functions as in step 4.

   Add input validation to your maxterm function, and test it, as in step 4.

   The required format for the output file at this step is summarized below, for the case when the input file contains the number 2:
   ```
   ECE 0301: Boolean Functions of 3 Variables.
   ```

```
Realization in Canonical Forms.

m2 = x'yz'
M2 = x + y' + z
```

When you are certain your program is correct, save it in a file named:
`ece0301_ICA05_step05.cpp`
Submit this file to the class repository.

6. Write a function that has the same name as your function from step 4 (i.e., you will be *overloading* this function) and returns the truth value for any minterm, given the truth values for the inputs. The inputs to this function are the `bool` variables specifying *x*, *y* and *z*, and an integer variable in the range 0-7 specifying a minterm. Have your function call the `AND`-gate function from step 3, with the appropriate inputs complemented, to compute the return value.

   For example, if this function is called with the `bool` parameters set to `x = false`, `y = true`, `z = true`, and the integer parameter set to 1, then it will return the truth value for
   $$m_1 = x'y'z = (\texttt{!false})(\texttt{!true})(\texttt{true}) = \texttt{false}.$$
   In other words, when the integer parameter is set to 1, your function should call the and-gate function with the inputs set to `!x`, `!y`, and `z`, respectively.

   Modify your truth-table-writing function so that it writes the truth table for the selected minterm. This function takes two parameters: an `ofstream` object for the output file, passed by reference, and an integer variable specifying the selected minterm, and returns nothing. The steps to be performed by this function are:
   - Write a line of text to indicate which minterm has been selected, e.g. if the integer parameter is 4:
     `Truth table for minterm 4.`
   - Write the truth table headers to the output file.
   - Use the loop from step 3 to write the rows of the truth table to the output file.

   Change the name of this function to reflect what it now does.

   Modify your `main` function so that, after writing the minterm expression to the output file, it calls the function that writes a truth table for the selected minterm to the output file. If you have done this correctly, the truth table in the output file will be 0 in all rows except for the row corresponding to the selected minterm. Test your program to ensure that the truth table is correct with different values for the integer in the input file.

7. Overload your maxterm function from step 5 by writing another function with the same name that returns the truth value for any maxterm, given the truth values for the inputs. The inputs to this function are the `bool` variables specifying *x*, *y* and *z*, and an integer variable in the range 0-7 specifying a maxterm. Have your function call the or-gate function from step 2, with the appropriate inputs complemented, to compute the return value.

   For example, if this function is called with the `bool` parameters set to `x = false`, `y = true`, `z = true`, and the integer parameter set to 1, then it will return the truth value for

   $$M_1 = x + y + z' = (\texttt{false}) + (\texttt{true}) + (\texttt{!true}) = \texttt{true}.$$

   In other words, when the integer parameter is set to 1, your function should call the or-gate function with the inputs set to `x`, `y`, and `!z`, respectively.

   Modify your truth-table-writing function so that it can also write the truth table for a selected maxterm. This function now takes three parameters: an `ofstream` object for the output file, passed by reference, an integer variable specifying the selected minterm or maxterm, and a `bool` parameter to select a minterm or maxterm, and returns nothing. If the `bool` parameter is `false`, this function must write the truth table for the minterm selected by the integer parameter, otherwise is must write the truth table for the selected maxterm. Change the name of this function to reflect what it now does.

   Modify your `main` function so that, for any integer in the range 0-7, the program writes the following to the output file:

   - the introductory message,
   - the Boolean expression for the corresponding minterm,
   - a truth table for the minterm,
   - the Boolean expression for the corresponding maxterm, and
   - a truth table for the maxterm.

   Test your program with each of the numbers 0-7 in the input file, and ensure that the output file is correct each time.

   The required format for the output file at this step is summarized on the next page, for the case when the input file contains the number 2.

```
ECE 0301: Boolean Functions of 3 Variables.
Realization in Canonical Forms.

m2 = x'yz'

Truth table for minterm 2.

x       y       z       f(x,y,z)
--------------------------------
0       0       0       0
0       0       1       0
0       1       0       1
0       1       1       0
(etc.)
1       1       1       0

M2 = x + y' + z

Truth table for maxterm 2.

x       y       z       f(x,y,z)
--------------------------------
0       0       0       1
0       0       1       1
0       1       0       0
0       1       1       1
(etc.)
1       1       1       1
```

When you are certain your program is correct, save it in a file named:
`ece0301_ICA05_step07.cpp`
Submit this file to the class repository.

8. Write a function that returns the canonical sum-of-products (CSOP) expression for any Boolean function of three variables. This function takes an integer parameter in the range 0-255 representing the function index as input, and returns a string object containing the CSOP expression. The general approach to follow is:
   - If the integer parameter is outside the range 0-255, print the following error message to standard output, and exit with the failure code.
     ```
     ERROR! Function index out of range.
     ```
   - Declare a string object to store the CSOP expression.
   - Write a loop to determine the binary representation for the value of the integer parameter.
   - Depending on the value of each bit, the corresponding minterm should either be included or not included in the CSOP expression. For each minterm that should be included, call your minterm expression function from step 4 and append what it

returns to the CSOP expression. You will need a variable to keep track of which minterm corresponds to each bit.

- After you append each minterm, subtract the bit value from the function index.
    - If the function index is now 0, then you have just appended the last required minterm to the CSOP expression, and the loop should terminate.
    - If the function index is not zero, then there are more minterms to find. Append the string literal "` + `" to the CSOP expression, and return to the top of the loop.

Modify your `main` function so that it reads a function index in the range 0-255 from the input file, and writes the corresponding CSOP expression to the output file. Test your program with a variety of values for the function index, and ensure that it produces the correct expressions.

9. Overload the function from step 8 by writing another function with the same name that returns the truth value for any CSOP expression, given the truth values for the inputs. The inputs to this function are the `bool` variables specifying *x*, *y* and *z*, and an integer variable in the range 0-255 for the function index.

   Inside your function, perform input validation on the function index, as in step 8. Next, declare a `bool` variable to store the return value, and initialize it to `false`. The remainder of your function will follow a similar control structure to that described in step 8, except that, for each minterm that should be included in the function:
   - Call your minterm function from step 6.
   - Compute the logical OR of what is returned from the minterm function with the `bool` variable that stores the return value, and store the result in the same variable.
   - At the end of the loop return the value stored in the `bool` variable.

   Modify your truth-table-writing function so that it writes the truth table for a CSOP expression to the output file. This function now takes two parameters: an `ofstream` object for the output file, passed by reference, and an integer variable specifying function index, and returns nothing. The steps to be performed by this function are:
   - Write a line of text to indicate the function index, e.g. for function 23:
     ```
     Truth table for CSOP form of function 23.
     ```
   - Write the truth table headers to the output file.
   - Use the loop from step 3 to write the rows of the truth table to the output file.

   Change the name of this function to reflect what it now does.

   Modify your `main` function so that, after writing the CSOP expression to the output file, the truth-table writing function is called. Test your program with a variety of values for the function index, and ensure that it produces the correct expressions and truth tables.

   When you are certain your program is correct, save it in a file named:
   ```
   ece0301_ICA05_step09.cpp
   ```
   Submit this file to the class repository.

10. Write two more functions that are analogous to those from steps 8 and 9, but for the canonical product-of-sums (CPOS) expression for the selected function. Test your new functions to ensure correct functionality. A correct program will, for any function index in the range 0-255:
    - write the introductory message to the output file,
    - write the CPOS expression for the selected function,
    - write a truth table for the selected function by evaluating the CSOP expression,
    - write the CPOS expression for the selected function, and
    - write a truth table for the selected function by evaluating the CPOS expression.

    Modify your program so that the input is read from a more informative file. The input file should have only two lines as follows:
    ```
    Boolean function of three variables
    function index = 23
    ```
    Use the `getline()` function to read the first two lines and confirm they are correct, and if so, convert the numeric characters to an integer for the function index, by using the `stoi()` function. You may also make use of the string class member functions in Gaddis, Section 10.7. If the function index is valid, write the items listed above to the output file.

The required format for the output file is summarized below, for the case when the function index is 23.

```
ECE 0301: Boolean Functions of 3 Variables.
Realization in Canonical Forms.

CSOP: f23 = x'yz + xy'z + xyz' + xyz

Truth table for CSOP form of function 23.

X       y       z       f(x,y,z)
--------------------------------
0       0       0       0
0       0       1       0
0       1       0       0
0       1       1       1
1       0       0       0
1       0       1       1
1       1       0       1
1       1       1       1

CPOS: f23 = (x + y + z)(x + y + z')(x + y' + z)(x' + y + z)

Truth table for CPOS form of function 23.

x       y       z       f(x,y,z)
--------------------------------
0       0       0       0
0       0       1       0
0       1       0       0
0       1       1       1
1       0       0       0
1       0       1       1
1       1       0       1
1       1       1       1
```

When you are certain your program is correct, save it in a file named:

`ece0301_ICA05_step10.cpp`

Submit this file to the class repository.

Don’t forget to include comments! You will lose credit if you leave them out, even if your code functions as directed!