ECE120: Introduction to Computing Optional C Exercise for Week 4

Printing Canonical SOP / POS

Overview:

In this exercise, you will write a program to print out the canonical SOP or POS (in minterms or Maxterms) of a Boolean function F of 3 or 4 variables plus X and Y, where X and Y are input from the console. The function F is different for each student, so please check the specifics of your unique function F in “canonicalForm_spec.txt”, a file in your working directory. To check out the files, please use the "svn update" command in your own ECE120 working directory, and all of the files for this exercise will appear in the folder named “OPTIONAL4” in your working directory. The details of your Boolean function F and which form your code should print out are specified in “canonicalForm_spec.txt”.

Function F:

Function F is defined by 3 or 4 variables plus X and Y. **This spec uses variables A, B, C, and D as an example to illustrate the function F of 4 variables in SOP form, but you must use the variables and form (SOP or POS) specified for you in canonicalForm_spec.txt** Plugging in the user-provided values of X and Y to function F results in a Boolean function that depends only on A, B, C, and D. For example, you might be given a Boolean function:
$F_{x,y}(A,B,C,D) = (A'+B+D+X')(B+C'+D'+Y)$ and asked to print the canonical SOP of function F. Plugging X = 0 and Y = 0 in to function $F_{x,y}$ gives $F_{0,0}(A,B,C,D) = B + C' + D'$, so your program should print out the canonical SOP of function $F_{0,0}(A,B,C,D) = (B + C' + D')$.

Specification:

You should implement the main function in the file “print_boolFunc.c” to do the following:

1. Use printf to print a prompt: “Please input X and Y values\n”.
2. Use scanf to read input values for the parameters X and Y.
3. Use printf to print out the function name, parameters and the operation (Pi or Sigma). You should print the parameters in alphabetic order. “Pi” represents the product operation. It should be used to print the canonical POS. As for printing the canonical SOP, “Sigma” represents the summation. Using the above example, this printf should print out “F(A,B,C,D)=Sigma(”
4. Use printf to print all of the minterms or Maxterms. The final output of your code should look like "F(A,B,C,D)=Pi(M0,M3,)\n" for the canonical POS, or "F(A,B,C,D)=Sigma(m1,m2,m6,)\n" for the canonical SOP. You must produce all commas, including the comma after the last term. **Fig 1** on the next page is an example of how the output of your code should look like.

```
F(A,M,W)=Pi(M6,M7,)
```

Figure 1: A, M, and W should be your own variables' names. To print the canonical POS, use "Pi" to represent the product operator and use the capital character "M" to denote maxterm. As for printing the canonical SOP, "Sigma" represents the summation operator and the character "m" denotes minterm.

5. See this Wikipedia page for better understanding of minterms and Maxterms https://en.wikipedia.org/wiki/Canonical_normal_form. Maxterms should be represented by "M" followed by the term number. And minterms are represented by "m". Notice that the term numbers can be different because of the ordering of variables, therefore, please make sure you use variables in alphabetic order when calculating the term numbers. For example, m10 for the variables A, B, C, and D means AB'CD' = 1 because 10_{10} is 1010 in binary, and m12 for the same variables means ABC'D' = 1. Bit indexing for maxterms has the opposite meaning; for example, M7 for variables A, B, and C means A' + B' + C' = 0 because 7_{10} is 111 in binary, whereas M4 for the same variables means A' + B + C = 0.

Compilation:

Use the command "gcc -Wall -g print_boolFunc.c -o opt4" to compile your code.
Use "./opt4" to run your code.
Please note that if the code that you submit cannot be compiled, you will receive no feedback.

Checkout and feedback:

Checkout:

Use the "svn update" in your working directory to get the folder "OPTIONAL4" which has all the files. You can check out a new ECE120 working directory using "svn checkout https://subversion.ews.illinois.edu/svn/fa16-ece120/netid ece120", where netid is your netid.

Commit:

After you commit your code to your svn repository (using the "svn commit" command), our tool will begin to grade your code. The grading should take a couple of minutes.

Feedback:

Please use the "svn update" command to get the feedback after the grading is completed. The feedback will be in separate files named "*.test", for example, "0.test". The feedback files are usually one or two sentences indicating your error. Test cases are also available for some errors. For this exercise, we will first test your printf formats, so please make sure that your printfs follow the requirements. For any questions regarding the test cases and this exercise, please email me at beipang2@illinois.edu or to Prof. Lumetta at lumetta@illinois.edu.
If the feedback has only one file "0.test" which says you have passed all the tests, then congratulations, you have completed this exercise.