ECE120: Introduction to Computing Optional C Exercise for Week 5

Printing a K-map

Overview:

In this exercise, you will write a program to print out the K-map for a Boolean function F of 3 or 4 variables plus X and Y, where X and Y are variables that you should read from the console. The function F is different for each student, so please check the specifics of your unique function in “kmap_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 “OPTIONAL5”. Your function should print out a 3- or 4-variable K-map, depending on the number of variables. The size of the k-map and the Boolean function are specified in “kmap_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, but you must use the variables specified for you “kmap_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: $F_{x,y}(A,B,C,D) = (A'+B+D+X')(B+C'+D'+Y)$. 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 $F_{0,0}(A,B,C,D) = (B + C' + D')$.

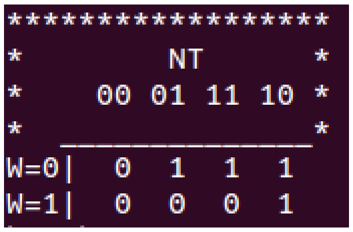
Specification:

You should implement the main function in the file “print_kmap.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. Please use "%d%d" or "%d %d" as the format. You should type in “0 0”, for example, to assign values to the parameters.
3. The first two variables should be used to label the columns of the K-map. You need to replace the “AB” in the code given to you with your own first two variables. If your variables are B, C, D, and E, you should replace the “AB” with “BC”.
4. The last 1 or 2 variables should be used to name the rows. Write code to print the names and values of variables for each row. For the example given above, you should use the format “DE=%d%d|” to print the variables’ names and values for each row. The two integers are the values for D and E, just like the K-map that you learned in class.
5. Please use a printf to print out the value of each cell in the K-map. Please print it as a decimal number using format "%d". This printf should print out just whitespaces and the

content of cells. You may use as much extra whitespace as you like to make your K-map looks nice. **Figure 1** on the next page is an example of how the K-map should appear.

6. Do not forget to print a newline character at the end of each row.



```
******************
*        NT      *
*    00 01 11 10 *
*  ______________*
W=0|  0  1  1  1
W=1|  0  0  0  1
```

Figure 1

The second line prints name of variables at the columns: N and T. At the start of each row, you need to print the variable name W and its value.

Hint:

- Remember that rows and columns follow a Gray code order in a K-map, so only one variable changes value between any two adjacent rows (or any two adjacent columns).

Compilation:

Use the command “gcc -Wall -g print_kmap.c -o opt5” to compile your code.
Use “./opt5” 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 ECE120 svn working direcotry to get the folder “printKmap” which has all the files. You can check out a new ECE120 svn 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 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 beipang2@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.