National Yang Ming Chiao Tung University
Department of Electrical Engineering
Data Structures, Fall 2022

October 26, 2022
Instructor: Mark P.-H. Lin
TA: Yu-Wei Chang

Programming Assignment #3

Linked Lists

1 Problem Description

Simplification of Boolean expressions is an important step while designing any digital system. The Karnaugh map, or K-map, is a method of simplifying Boolean algebra expressions by taking advantage of humans' pattern-recognition capability. However, with more input variables, pattern recognition in K-maps can be tedious or even infeasible.

Instead of simplifying Boolean expressions with K-maps, the Quine–McCluskey method has an advantage over K-maps when a large number of inputs are present. It consists of two steps: (1) finding all prime implicants of a Boolean function, and (2) selecting a minimal set of prime implicants for the function. Such method is functionally identical to K-map, which is much more efficient for use in computer programs.

In this programming assignment, you are asked to find all prime implicants of a Boolean function based on the Quine–McCluskey method, which consists of the following steps:

1. Classify all the minterms (implicants) with the value "1" into different groups according to the number of 1's in their binary values.
2. Combine any two implicants if they differ in exactly one bit in the binary values, replace the bit with "–", and cross out the combined implicants.
3. Repeat Steps 1 and 2 until the implicants cannot be further combined.
4. Output the remaining implicants, or prime implicants.

An example of finding the prime implicants of a Boolean function is demonstrated below. Given the following four-variable Boolean function containing seven minterms with the value "1", $F(a,b,c,d) = \sum m(0,5,7,10,11,13,15)$, the Quine–McCluskey method can be demonstrated with the tabular form.

Group	Initial	Combination 1	Combination 2
Zero 1	0000 ($m0$)		
One 1			
Two 1's	~~0101 ($m5$)~~ ~~1010 ($m10$)~~	~~01–1 ($m5, m7$)~~ ~~–101 ($m5, m13$)~~ 101– ($m10, m11$)	–1–1 ($m5, m7, m13, m15$)
Three 1's	~~0111 ($m7$)~~ ~~1011 ($m11$)~~ ~~1101 ($m13$)~~	~~–111 ($m7, m15$)~~ 1–11 ($m11, m15$) ~~11–1 ($m13, m15$)~~	
Four 1's	~~1111 ($m15$)~~		

2 Input Format

The input file consists of two lines. The first line gives the number of variables, N, of the Boolean function, F, where $3 \leq N \leq 16$, while the second line gives all minterms with the value "1", ranging from 0 to $2^N - 1$. We ignore all "don't cares" minterms, or the minterms with the value "X", in this programming assignment. The sample input below indicates the Boolean function, $F(a, b, c, d) = \sum m(0, 5, 7, 10, 11, 13, 15)$.

Sample Input	Comments
4	// the number of variables, N, of the Boolean function, F
0,5,7,10,11,13,15	// minterms with value "1"

3 Output Format

The output file generated by your program must include the following information: (1) the minterms after initial grouping, (2) the implicants after each combination iteration, and (3) all prime implicants. The output file format is illustrated by the sample output below, which results from the aforementioned sample input, where the key words, "Initial Grouping", "Combination #", and "Prime Implicants" must be included together with your results.

Sample Output	Comments
Initial Grouping	// the initial minterm grouping results
0: 0000	// according to the number of 1's
1:	// in the binary value of each minterm,
2: 0101, 1010	// where the number of 1's ranges from 0 to N,
3: 0111, 1011, 1101	// and $N = 4$ given in the sample input
4: 1111	
Combination 1	// the resulting implicants
0:	// after the first iteration of combinations
1:	
2: 01–1, –101, 101–	
3: –111, 1–11, 11–1	
4:	
Combination 2	// the resulting implicants
0:	// after the second iteration of combinations
1:	
2: –1–1	
3:	
4:	
Prime Implicants	// the resulting prime implicants, which cannot
0000, 101–, 1–11, –1–1	// be further combined with any other implicant

You may interchange the order of the resulting implicants in each group, as well as the order of the resulting prime inplicants in the output file.

4 Command-line Parameter

In order to test your program, you are asked to add the following command-line parameters to your program:

[executable file name] [input file name] [output file name]

5 Submission Information

1. Your program must be written in the C/C++ language and can be compiled on the Linux platform.
2. The source files of your program must be named with "[your student ID].h" and "[your student ID].cpp".
3. To submit your program, please archive all source files of your program into a single zip file, named "[your student ID].zip", and upload it to E3.

6 Due Date

Be sure to upload the zip file by "Wednesday, November 9, 2022". There will be a 25% penalty per day for late submissions.

7 Grading Policy

The programming assignment will be graded based on the following rules:

- Pass all sample inputs on E3 with compilable source code (50%)
- Pass five hidden test cases (50%)

Be sure to design your own "linked lists" and/or "arrays" in this programming assignment. You are not allowed to use any container in the Standard Template Library (STL), or any other open-source library, such as vector, list, map, set, and all the other container classes. The submitted source codes will NOT be graded if those open-source container classes are found in your source codes.

The submitted source codes, which are either copied from or copied by others, will NOT be graded.