# 

Three-Variable K-Map Logic Minimization

Project 1

By: Laila Nasser ElKoussy & Yousef Mohab Koura

900160812   900160083

# Design:

## Classes & Structs:

For the program, we created one class, *KMap*, which contains all the arrays, vectors, structs, and functions necessary for the generation of the k-map and its equivalent reduced Boolean expression. We also created two structs. The first is a minterm struct, which contains variables that store its value (1 or 0), its decimal index (0-7), and the number of times it’s been included in an implicant. The second is an implicant struct, which contains a vector that holds the indices of the minterms in the implicant and variables that hold the implicant’s size and expression.

## Input & Validation:

The input for the program is a single string that consists of integer numbers—which range from 0-7 to denote minterms—separated by commas and terminated by a period. If the input string violates any of the aforementioned requirements, the program asks the user to reenter it. After the input string is validated, it’s passed on to the k-map generation function (a member of the k-map class) to be processed and used to generate a visual k-map and its reduced Boolean expression.

## Visual K-Map Generation:

The *generate* function first extracts each integer value (minterm) and pushes it into a storage vector; after each insertion in the vector, an integer variable called ones is updated to reflect the number minterms in the k-map. The function then checks the number of ones to see whether or not the function is a tautology (always true). If it is always true, another function—called *tautology()*—is called; this function prints a k-map filled with ones and states that the function is always true. However, if the function is not found to be a tautology, another function—called *process()*—is called. The process function then handles calling the function for each subsequent task. First, it calls the *fill()* function which fills a 2D array with 1’s and 0’s equivalent to the input minterms and calls the *printmap()* function to print the 2D array (location in the array is based on an index given by the *getindex* function).

## Reduced Boolean Expression Generation:

The *extract()* function is then called. This function extracts all valid implicants that exist in this k-map. It consists of a for loop that passes by each cell in the map and checks whether it is an included minterm. If it is an included minterm, it first adds an implicant which contains it alone and then checks around it—to its sides, below it (if it it’s in the top row), if it’s a line of 4 minterms, and if it’s a square of minterms— to see if larger implicants can be generated. Whenever an implicant is found, it is pushed into the vector of implicants and the number of includes of each minterm in the map is updated. *Process()* then calls the *order()* function, which simply orders the implicant vector from biggest to smallest in size. Afterwards the *simplify()* function is called; this function takes the ordered implicant vector and checks whether each of its implicants is useless or not (this is done using a Boolean function called *useless*, which checks if the implicant includes at least one minterm that isn’t included in any other implicant). If the implicant is useless, it’s deleted and the number of inclusions for each minterm is updated. If it’s found to be essential, it is pushed into another vector that includes all essential prime implicants. The function then lists the number of essential prime implicants and displays the minterms in each implicant. Then, the *translate* function is called. This function receives each essential prime implicant and translates it into its Boolean expression equivalent by converting each minterm into bit form and comparing them all to determine which bits changed and which didn’t. Finally, the program prints all the converted Boolean expressions to form the reduced Boolean expression is SOP form.

# Build & Usage:

The program source consists of the main .cpp file and the KMap class’ .h & .cpp files; including these files in a C++ project is sufficient to build the program. The usage of the program is very simple, as it only requires a single input (the minterm string). The program then uses this minterm string to perform all subsequent tasks. Therefore, the program can be used to easily get the reduced Boolean expression for any 3-variable Boolean function.