CSCE 2301-01
Project 1:
Karnaugh Map Calculation Program

Sara Mohamed
900203032

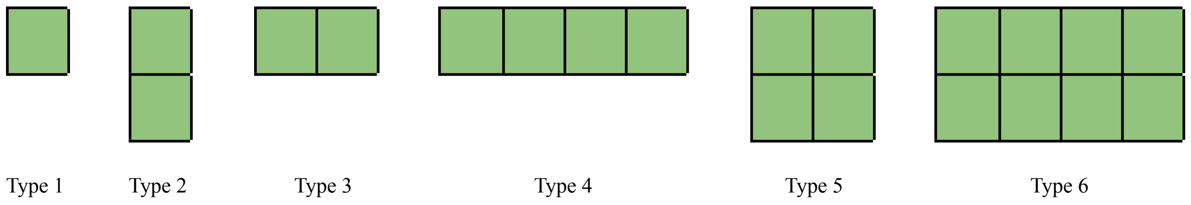
<u>Instructions for Use:</u>

Upon running the program, the user will be prompted to enter the desired number of variables to be used in the K-Map, and can choose either one, two or three. After making the choice, the user will then be prompted to enter the number of minterms for the K-Map function generation. (This number depends on the number of variables selected, so it ranges from 1-2 for one variable, from 1-4 for two variables, and for 1-8 for three variables). The program will then prompt the user to enter the minterms in their decimal representations, and will check each one, informing the user of an error if one of the minterm values is not within the acceptable range. If an error is detected in one of the minterms, the program will ask the user to re-enter a valid minterm instead. Afterwards, the program will generate the K-Map based on the appropriate minterms entered, as well as output the function in its simplest form.

<u>Program Structures</u>

This program operates through two structs: *cell* and *mterm*, as well as a globally initialized vector of vectors of cells called *KMap,* a vector of mterms *Minterms,* and another vector of mterms *FilteredTerms*. The struct *cell* represents a cell in the KMap, and contains the attributes *binrepresentation*, which is a string of the binary representation of the cell in the K-Map, *decrepresentation*, which is an integer of the decimal representation of the cell in the K-Map, and *value*, which is an integer of the value of the cell in the K-Map, either 0 if the cell is not within the selected minterms for the function, or 1 if it is. In the *cell* struct, there is also an overloaded function ==, which compares two cells and declares them equal if they have matching *decrepresentation*, and unequal if they do not. This function will help in filtering out redundant minterms later.

The struct *mterm* is a struct representing the minterms of the function that will be calculated in the program. *Mterm* contains two attributes: *minterm*, which is a vector of *cell*s, as well as *type* which is an integer representing the type of the minterm selected. This *type* is the main component of calculating the most simplified minterms and the most simplified function for the K-Map. The type of minterm is represented in the diagram below.



This numbering is also going to come in handy when prioritizing which redundant minterms to keep, such that, in general, the larger the type number of the minterm is, the more useful it is in getting the most simplified function.

Kmap is initialized globally, and is a vector of vectors of cells. This structure is empty, and is filled with cells depending on the number of variables chosen by the user.

<u>Program Flow</u>

Depending on the number of variables the user enters, the program calls on an inputMap function, a displayMap function, a printFunction function, and a calcAllMinterms function. There is one function called filterDuplicates(), that is shared among all the variable numbers.

In the main function, as aforementioned, the user enters the number of variables wanted. Based on the number of variables, the program calls either inputMap1(), inputMap2(), or inputMap3(). Each of these functions push_back vectors in KMap, by initializing the cells based on the number of variables required e.g. KMap has two rows and two columns for two variables, but two rows and one column for one variable; and for two variables, KMap[0][1] has binrepresentation “01”, while for three variables, KMap[0][1] has binrepresentation “001”. The functions then ask for the number of minterms, with different ranges for each function. Then, the functions accept the minterms from the users, also within different ranges. Finally, each function changes the value of the cells in the KMap from 0 to 1 for the minterms chosen by the user.

Each of the inputMap() functions call on a displayMap() function, which simply displays the value of each cell in the KMap for the user using simple for loops and cout statements.

For one variable KMaps, the function displayMap1() then calls on the function printFunction1(). This function checks if the first cell only is of value 1, for which it prints “A” ”. If the first cell is of value 0 and the second cell of value 1, it prints “A”. If both are of value 1, it prints out “1”. This is due to the simplicity of one-variable K-Maps. For the two variable and three variable K-Maps, it is slightly more complex, so displayMap2() calls on calcAllMinterms2(), and displayMap3() calls on calcAllMinterms3().

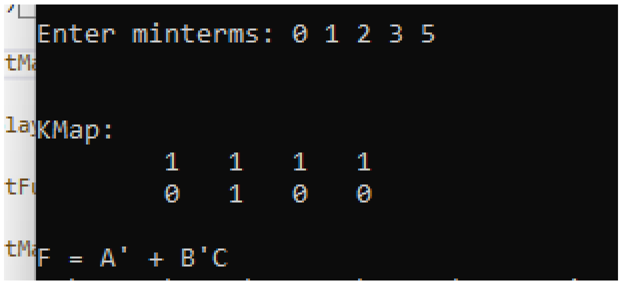
calcAllMinterms3() and calcAllMinterms2() have the same algorithm, with minor differences. calcAllMinterms3() loops over all minterms with value 1, and tries to find the largest grouping of minterms it can achieve with the selected minterm. First, it checks if all 8 cells are 1s. If so, it creates a mterm of type 6 with a vector containing all the cells in the K-Map and pushes it into the vector Minterms. If not, it checks to see if the cell to the right, the cell to the bottom, and the cell to the bottom right are all 1s. If so, it creates a mterm containing a vector with all those cells with type 5, and pushes that into Minterms. If not, it checks to see if the row containing the cell is a row of all 1s. If so, it creates a mterm with the vector containing all those cells of type 4, and pushes the mterm into MinTerms. If a minterm of type 5 or of type 4 wasn't found for the selected cell, the function checks the cell to the right to see if it is of value 1. If it is, it creates a mterm with the vector containing both cells, of type 3, and pushes the mterm into Minterms. If no minterm of type 5 was found for the selected cell, the function checks the cell underneath to see if it is of value 1. If it is, it creates a mterm with the vector containing both cells, and of type 2, and pushes the mterm into Minterms. If no minterm of any type was found, the function creates a mterm with a vector containing only the selected cell, of type 1, and pushes it into Minterms. The aim of this algorithm is to limit the number of duplicate minterms added into the vector Minterms, by first checking for larger groups before checking for smaller groups, and limiting the checking of smaller groups if larger groups were already found. calcAllMinterms2() contains the same algorithm, but starts by checking for minterms of type 5, and ignores those of type 4, since type 4 and type 6 minterms don't exist in K-Maps of two variables.

What is left is to filter the minterms: removing minterms that are repeated twice, and minterms whose cells are already included in other minterms. To do so, both calcAllMinterms2() and calcAllMinterms3() call on the same function, filterDuplicates(), and then on printFunction2(), and printFunction3(), respectively. filterDuplicates() creates a vector of bools called exclude, initialized all to false, with size equal to the size of Minterms, i.e the amount of unfiltered Minterms. The values in exclude correspond to whether the corresponding index in Minterms should be excluded in comparisons or not. It then loops over each minterm, call it temp, in Minterms and further loops over each cell in temp. It checks each minterm of type <= temp.type (except for type 2 and 3, because 3 is not necessarily more simplified than 2) whose corresponding value in exclude is false, to see if the other minterm contains the cells in temp. If, after looping through the minterms, it finds that at least one of the cells in temp is not present in other minterms, it pushes temp into the vector FilteredTerms. If it finds that every cell in temp is present in other minterms that were checked, it does not push temp into FilteredTerms, and it turns the corresponding value of temp in exclude to true, so it is not checked in upcoming loops.

The functions printFunction2() and printFunction3() then loop over FilteredTerms, and manually check the type of each minterm and the cells in it to print out the appropriate expression for the minterm in FilteredTerms. This is manually done through the use of if statements. For example, if printFunction3() finds a minterm of type 5 that contains cells KMap[0][1] and KMap[0][2], it prints "C".

Note: This program considers A the most significant bit, and C the least significant bit. This means that functions of only one variable will contain only A, functions of two variables will contain A and B, and functions of three variables will contain A, B, and C.

Note: The complexity of this program is of O(n2), where n is the number of cells in the K-Map, however, since the number of cells ranges from 2 to 8 in this particular program, this is not of significant time inefficiency.



```
Enter minterms: 0 1 2 3 5


KMap:
        1   1   1   1
        0   1   0   0

F = A' + B'C
```

~ sample test from project document