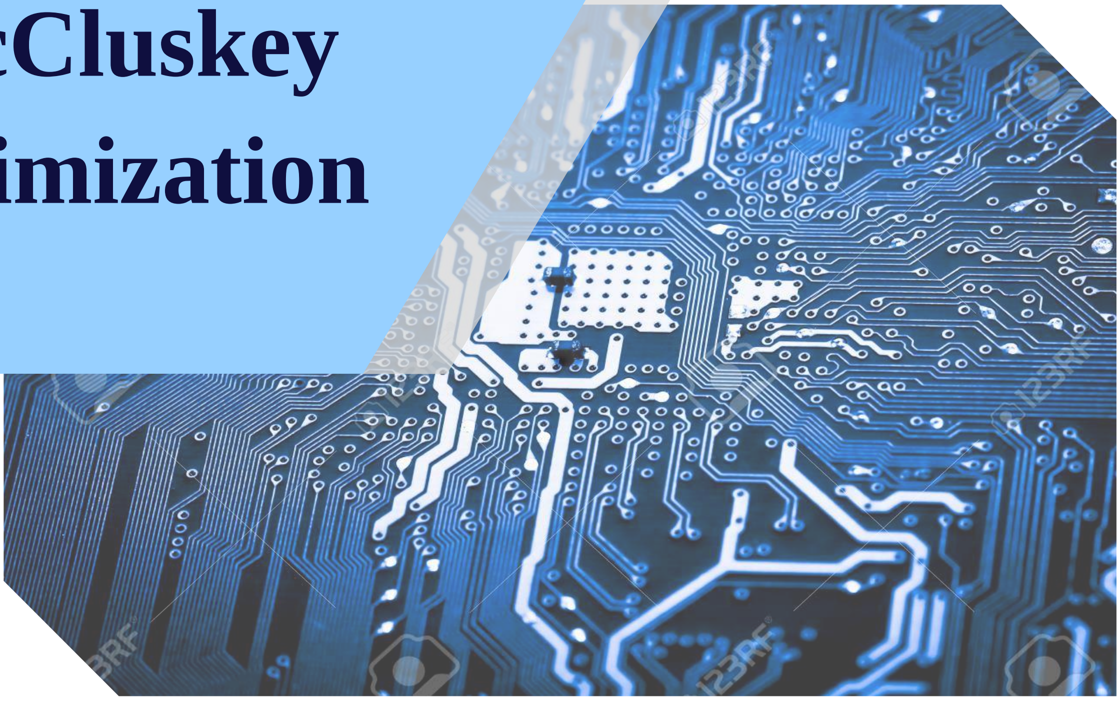
CSCE2301 – Digital Design I

<u>Project 1 Report</u>

Quine-McCluskey Logic Minimization



Group members :	Marwah Sulaiman	900172284
	Mohamed Abu Elwafa	900172603

Supervised by : Dr. Mohamed Shalan

Contents

Program Design

1. Classes :

• Binary_number class

- We constructed binary_number class which represents any binary number (used in program to represent covers) as a combination of :
  1. A decimal number (unsigned num)
  2. A number representing the locations of the dashes (unsigned dashes)
  3. Boolean expressions indicating whether is is used to get combined with another binary number (bool is_used)

  Also, every binary number has an associated vector (vector<int> covered_mins) that includes all the minterms and don't cares that are covered by this number as integers.

- The class includes 3 main functions :
  1. void print_with_dashes(int Bits_num) : prints the binary number to Bits_num decimal places with dashes (e.g. 01--)
  2. int count_ones(int n) : counts the number of ones in the binary number
  3. void print_number(int Bits_num) : prints the binary number to Bits_num decimal places. (e.g. 0111)

- the class is used in the program in the representing prime implicants and essential prime implicants as well as in the operations performed on the minterms and don't cares to reach
- them.

2. Used Data Structure : Vectors

We used the vector data structure to implement the tables of implicants (discussed in next part) because it is fully dynamic and much easier to use than dynamic arrays or maps in implementation. In other words, vectors facilitaed the proceses of searching, insertion and reading.

3. Algorithm and code flow

1- Reading from file

Firstly, we read the number of variables, then the minterms and the don't cares which we store in 2 vectors of type unsigned. During the reading, we check if there are any errors such as :

- Number of minterms+don't cares are greater than 2^number of variables
- A duplicate in both minterms or don't cares
- Inserting more than 3 lines in the text file
- A number existing in both minterms and don't cares.

2- Creating initial table

Secondly, we used the vector data structure to implement the Quine McCluskey Implication table. To clarify, we implemented a vector of vectors of type binary_number to create the initial table/first column of the Implication table. We created a function (create_MinTable) that uses the minterms and don't cares vectors (part 1) to fill the initial table. Every vector inside represents a group of binary numbers (minterms and don't cares) that have a specific number of ones in order to facilitate the comparison process in the next stage.

3- Generating prime implicnts and their covered minterms and don't cares

Thirdly, we used the initial table to generate the prime implicants through a loop that keeps generating combined implicants till it reaches implicants that can't be combined. To demonstrate, we created another vector of vectors (mid_table) of the same type. At the first time, the loop takes the initial table and generates the mid_table that contains the combined implicants from the initial table, then it considers this mid_table as the initial table and generates mid_table with new combined elements and keeps doing so till it produces a mid_table that has zero elements. At this point, we can say that we reached the prime implicants. Lastly, we push the prime implicants in another vector of vectors (final_table). Note: in every loop we check if there are any implicants that are not used in any combinations and push them in the final table.

Program Design (cont’d)

- Combining algorithm :

  We compare each implicant to all implicant in its next group to create combined implicant (of type binary_number too) following this algorithm :
  1- AND the number of each and get a new number
  2- XOR the numbers of each and get the dashes of the new number
  3- If the number of dashes is greater that 1, then the two implicants can’t be combined, because this means we have a change in more than one bit between the two numbers.
  4- In every combination, we push the covered minterms and don’t cares (covered_mins) of each number in the covered_mins vector of the new number

- Lastly, we print all the prime implicants and their covered_mins vectors that represent the minterms and don’t cares they cover.

4- Generating essential prime implicants

Fourthly, we loop on the prime implicants in the final_table, and for every prime implicant, we loop on its covered minterms and chack if it exists as a covered minterm by another prime implicant. Once we finish all comparisons with such existence, we push the prime implicant in the essential PIs vector.

5- Generating minterms that are not covered by EPIs

Lastly, we loop on the minterms vector (part 1) and check every minterms whether it is covered by EPI or not by comparing it to all minterms covered by essential prime implicants (covered_mins vector of the EPIs). If it doesn’t exist there, we print it.

Instructions

- Input instructios :

The inputs are provided by a text file structured as the following:

3 lines :

- the first line : number of variables (Supported upto 16 variables)
- The second line: the minterms separated by commas (‘,’)
- The third line : the don’t care terms separated by commas (‘,’)

for Example:

3

1, 3, 6, 7

0

- how to build and use the program :

1. put the 3 source codes in the project file
2. Put the text file with the correct format with the sources file :
3. Run the program

Test Cases

1.

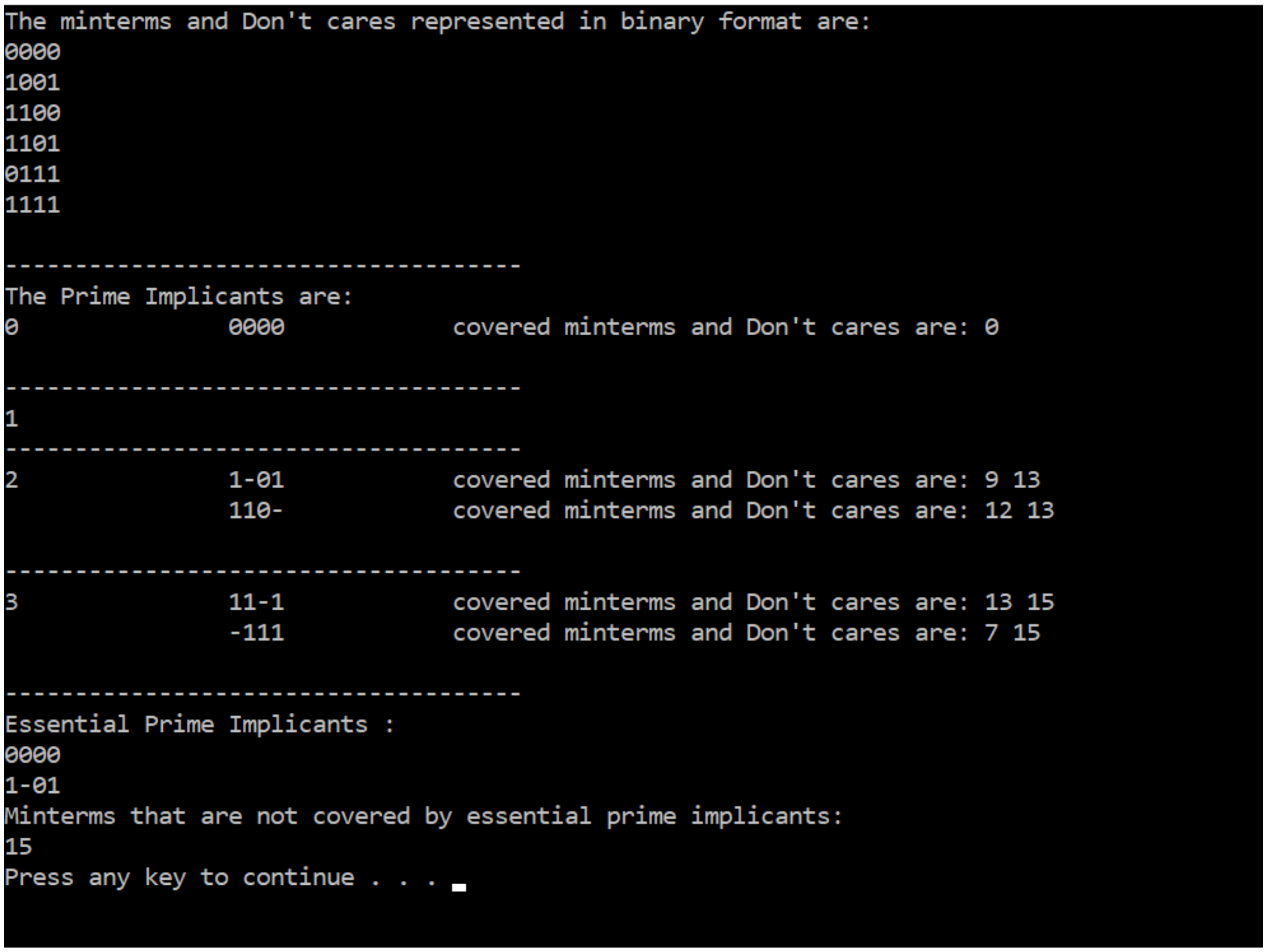
- text file (input) :

  4

  0, 9, 13, 15

  7, 12

- output :



```
The minterms and Don't cares represented in binary format are:
0000
1001
1100
1101
0111
1111

-------------------------------------
The Prime Implicants are:
0               0000            covered minterms and Don't cares are: 0

-------------------------------------
1
-------------------------------------
2               1-01            covered minterms and Don't cares are: 9 13
                110-            covered minterms and Don't cares are: 12 13

-------------------------------------
3               11-1            covered minterms and Don't cares are: 13 15
                -111            covered minterms and Don't cares are: 7 15

-------------------------------------
Essential Prime Implicants :
0000
1-01
Minterms that are not covered by essential prime implicants:
15
Press any key to continue . . .
```

Test Cases (cont’d)

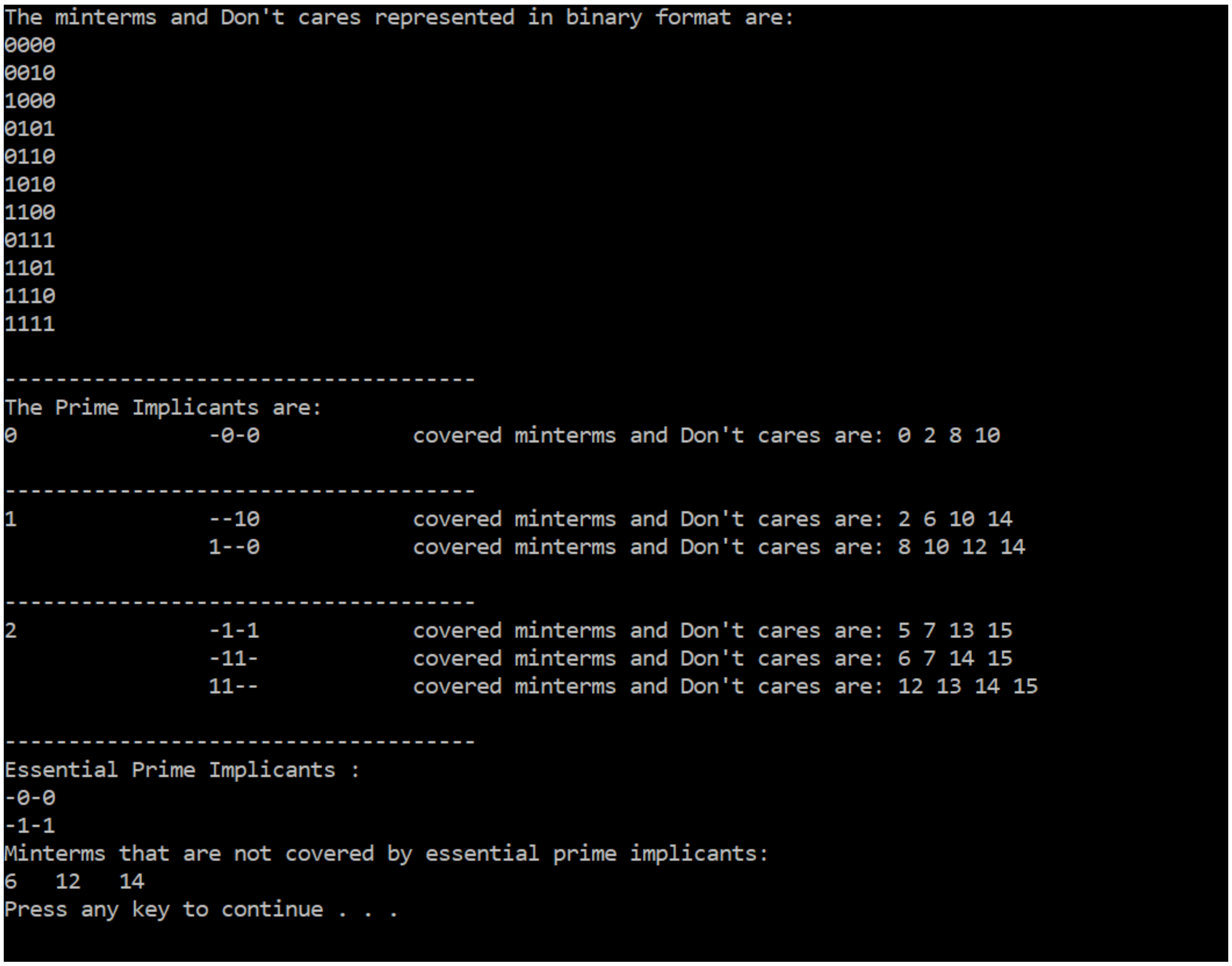
2.

- text file (input) :

  4

  0, 2, 5, 6, 7, 8, 10, 12, 13, 14, 15

- output:

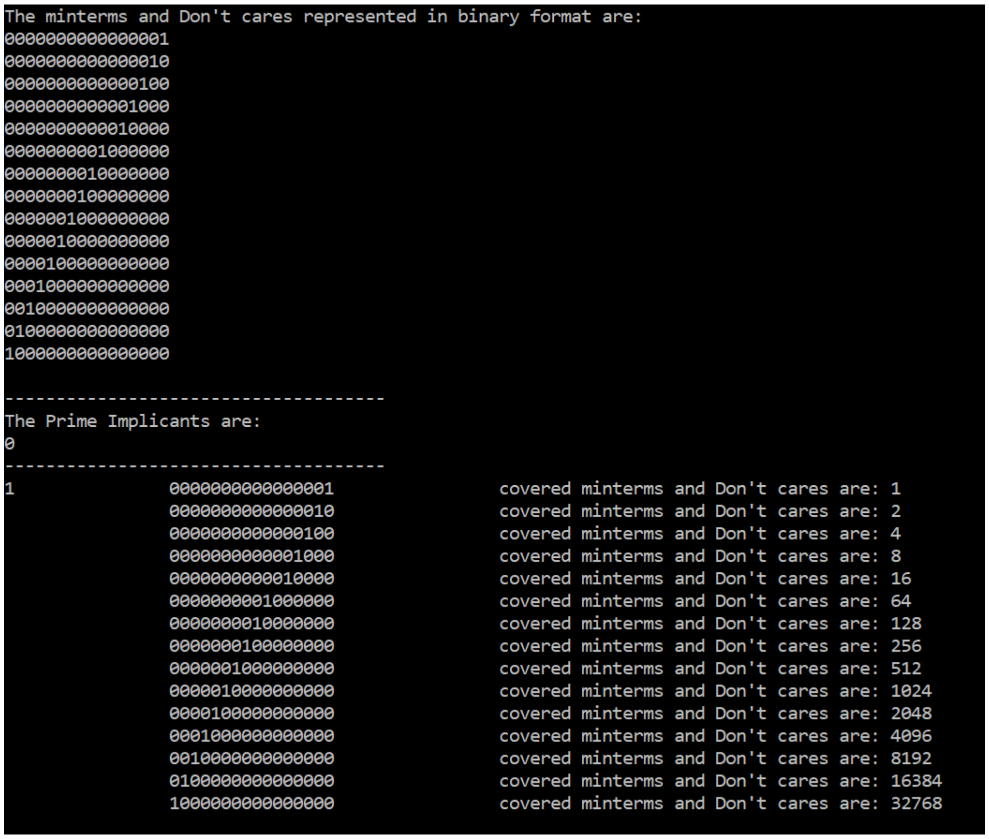


```
The minterms and Don't cares represented in binary format are:
0000
0010
1000
0101
0110
1010
1100
0111
1101
1110
1111

-------------------------------------
The Prime Implicants are:
0               -0-0           covered minterms and Don't cares are: 0 2 8 10

-------------------------------------
1               --10           covered minterms and Don't cares are: 2 6 10 14
                1--0           covered minterms and Don't cares are: 8 10 12 14

-------------------------------------
2               -1-1           covered minterms and Don't cares are: 5 7 13 15
                -11-           covered minterms and Don't cares are: 6 7 14 15
                11--           covered minterms and Don't cares are: 12 13 14 15

-------------------------------------
Essential Prime Implicants :
-0-0
-1-1
Minterms that are not covered by essential prime implicants:
6   12   14
Press any key to continue . . .
```

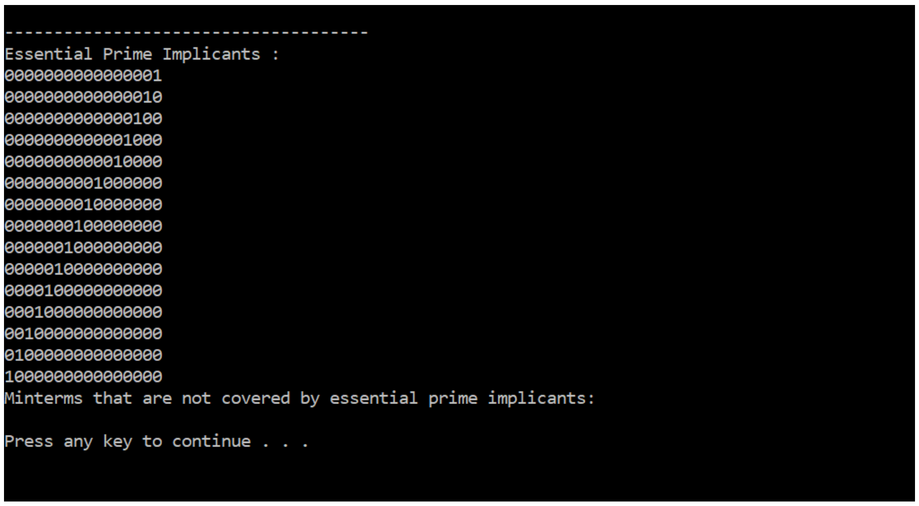
Test Cases (cont’d)

3.

- text file (input) :
  16
  1,2,4,8,16,64,128,256,512,1024,2048,4096,8192,16384, 32768
- output:



```
The minterms and Don't cares represented in binary format are:
0000000000000001
0000000000000010
0000000000000100
0000000000001000
0000000000010000
0000000001000000
0000000010000000
0000000100000000
0000001000000000
0000010000000000
0000100000000000
0001000000000000
0010000000000000
0100000000000000
1000000000000000

------------------------------------
The Prime Implicants are:
0
------------------------------------
1               0000000000000001                covered minterms and Don't cares are: 1
                0000000000000010                covered minterms and Don't cares are: 2
                0000000000000100                covered minterms and Don't cares are: 4
                0000000000001000                covered minterms and Don't cares are: 8
                0000000000010000                covered minterms and Don't cares are: 16
                0000000001000000                covered minterms and Don't cares are: 64
                0000000010000000                covered minterms and Don't cares are: 128
                0000000100000000                covered minterms and Don't cares are: 256
                0000001000000000                covered minterms and Don't cares are: 512
                0000010000000000                covered minterms and Don't cares are: 1024
                0000100000000000                covered minterms and Don't cares are: 2048
                0001000000000000                covered minterms and Don't cares are: 4096
                0010000000000000                covered minterms and Don't cares are: 8192
                0100000000000000                covered minterms and Don't cares are: 16384
                1000000000000000                covered minterms and Don't cares are: 32768
```



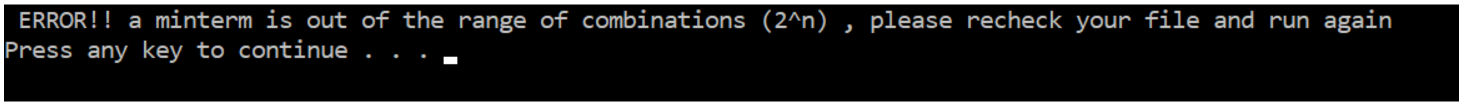
```
------------------------------------
Essential Prime Implicants :
0000000000000001
0000000000000010
0000000000000100
0000000000001000
0000000000010000
0000000001000000
0000000010000000
0000000100000000
0000001000000000
0000010000000000
0000100000000000
0001000000000000
0010000000000000
0100000000000000
1000000000000000
Minterms that are not covered by essential prime implicants:

Press any key to continue . . .
```

Test Cases (cont’d)

Error Checking:

1.

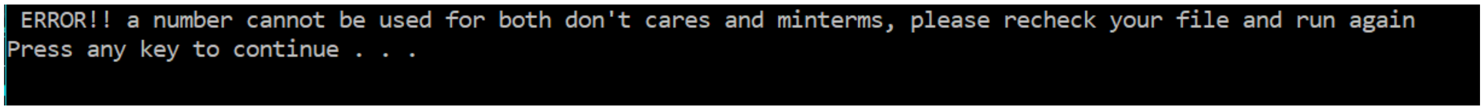
- text file :
  3
  0, 2, 5, 6, 15
- output :



```
 ERROR!! a minterm is out of the range of combinations (2^n) , please recheck your file and run again
Press any key to continue . . .
```

2.

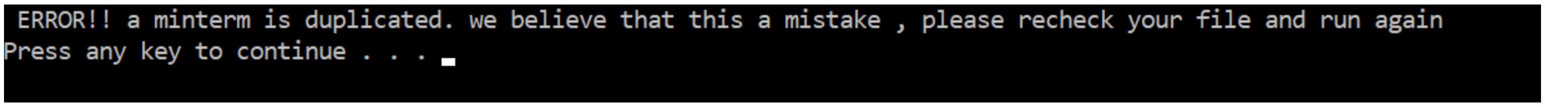
- text file :
  3
  0, 2, 5, 6
  3, 5
- output :



```
 ERROR!! a number cannot be used for both don't cares and minterms, please recheck your file and run again
Press any key to continue . . .
```

3.

- text file :
  3
  0, 2, 5, 6, 6
  3
- output :



```
 ERROR!! a minterm is duplicated. we believe that this a mistake , please recheck your file and run again
Press any key to continue . . .
```

Contributions

- We divided tasks on both of us to do, but we worked together in debugging and solving the problems we faced in coding.

Our main responsibilities:

Marwah Sulaiman	**Mohamed Abu Elwafa**
Reading from file	Error checking
Creating initial table (create_MinTable function)	Creating binary_number class
Creating the loop generating prime implicants	Creating create_combined function
Generating essential prime implicants and the minterms that are not covered by them	Generating the minterms and don’t cares covered by prime implicants
Project report	Readme file
Testing and debugging in all parts together	
Handling git hub and its issues together	