Programming Assignment 2
K-map-based logic minimization

Introduction

- Objective
  - Implement the k-map-based logic minimization approach
- Description
  - The program reads a Boolean function (with 4 variables) in the k-map format
  - Then, it optimizes the Boolean function with the k-map-based logic minimization approach
  - Finally, it outputs the minimized Boolean function in the SOP form

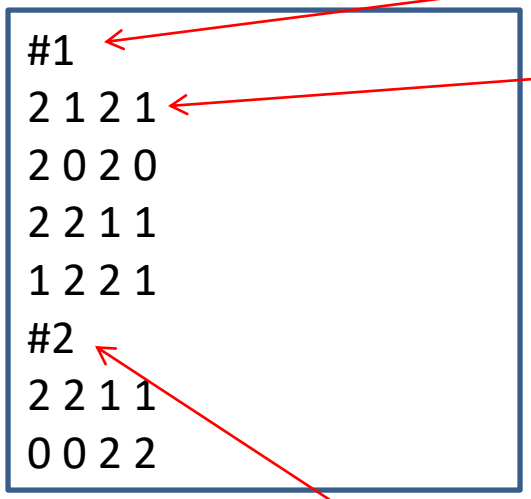
Input format

- A Boolean function has four input variables, w, x, y, z in order
- All the Boolean functions to be minimized are stored in an input text file
- In the input file
  - The first integer indicates the number of Boolean functions
  - Each line represents a Boolean function **in the k-map format**
    - 16 integers separated with a space: 0 for 0, 1 for 1, and 2 for X
    - These 16 integers correspond to the squares in the k map from top to down and left to right in order

```
2
1 0 1 1 0 1 1 0 0 1 1 0 1 1 1 1
2 1 1 2 0 2 1 0 0 0 1 0 0 0 1 0
```

Output format

- The minimized Boolean functions are stored in order in an output file
- A Boolean function is represented in the SOP form



The 1st Boolean function

- **Each line represents a minterm**
- **The input variables are w, x, y, and z in order**
- **0: the input variable is in the complement form**
- **1: the input variable is in the normal form**
- **2: the input variable is a don't care (does not appear in the term)**
- **E. g.: 2 1 2 1 presents xz**

The 2nd Boolean function

Requirements

- Your program should works correctly
- Your program should be executable and compiled by legally licensed compliers

Delivery

- Due date
  - 11/18 (Wed.)
  - Fixed deadline, no late delivery is allowed
- Deliveries
  - Your source code
  - A readme describing how to run your program
  - Pictures show your execution results for the given testbench by PrintScr
- Notice
  - **YOU WILL GET A VERY LOW SCORE, IF YOUR SOURCE CODE IS SIMILAR TO OTHERS**

Bonus

- **You will get a bonus score, if your program can deal with a function with more than 4 variables**
- **Reference: Quine-McCluskey algorithm**