Programming Assignment 1:

Quine-McCluskey Method (Due: 08/09)

Problem Descriptions:

Implement a two-level logic optimizer based on Quine-McCluskey method. The first step is to generate all prime implicants for the given function. The second step is to choose a minimum-cost cover and generate a sum-of-products expression of this function with **minimum number of prime implicant** (If there exist different solutions with the same minimum number of prime implicant, please choose the one which has minimum number of literals).

Input format:

You should allow input from a file.

f(A, B, C, D) = ∑ m(4, 5, 6, 8, 9, 10, 13) + ∑ d(0, 7, 15).

The following is an example:

```
.i 4                /* input variables(A,B,C,D) */
.m                  /* on set   */
4 5 6 8 9 10 13
.d                  /* don't care set */
0 7 15
```

The number of input variables is limited to 8.

Output format:

Generate all prime implicants and the minimum sum-of-products expression to a file.

Prime implicants are separated by a line and expressed in uppercase. They should be sorted according to **literal number** and **alphabetical order**, please notice that **{X > X'}.** For example: `A > A' > B >….`

If there are more than 15 primary implicants, report **only the first 15 primary implicants**. However, you still have to report the correct number of total primary implicants at ".p" field.

The following is an example:

```
.p 7                /* there are 7 prime implicants */
AB'C'
AB'D'
AC'D
A'C'D'
B'C'D'
A'B
BD

.mc 3               /* 3 prime implicants in minimum covering */
AB'D'
AC'D
A'B
literal=8
```

Note that literal is the sum of chars in each prime implicants. For example: A'B literal = 2; AB'D' literal = 3.

Compile & Execute:

You should write a makefile to generate an executable binary <student_id>.o and your program must be able to receive commands in following format.

```
$ ./<student_id>.o        <input file name>.txt     <output file name>.txt
```

For example:

```
$ ./311510141.o           input1.txt                output1.txt
```

Note that input and output file should be the arguments of program. Please make sure your code can be compiled and executed on workstation.

Grading:

There are six testcases in total: Two open cases and four hidden cases.
Your grade depends on the correctness and runtime of your program:

1. Correctness (80%) : **10** points for each open case, **15** points for each hidden case
2. Performance (20%) : Determined based on the ascending order of runtime.

For each case, the run time **limit is up to 300 seconds**. It will be regarded as failed if you use more than 300 seconds.
The running time ranking is based on the sum of execution time for each case. Please note that only those who pass all testcases will be included in the runtime ranking.

Program Submission:

1. Please use C++ language to write your programs.
2. Submission file naming rules:

   Please Create a "<student_id> folder " with the following files in it.

   - Source code (.cpp, .h, .c) (ex: <student_id>.cpp)
   - Makefile (ex: Makefile)
   - Report (ex: <student_id>.pdf)

   Use the command below to compress "<student_id> folder " in the Linux environment:

   ```
   $ tar  cvf   <student_id>.tar   student_id
   ```

   Upload this “<student_id>.tar” into E3.
3. Anyone who violates submission file naming rules will get 10 deduct points.
4. Anyone who doesn't follow the requirements of output format will get 20 deduct points.
5. Do not print any words on terminal, otherwise you will get 20 deduct points.
6. **Plagiarism is forbidden, otherwise you will get 0 point!!!**
7. **Delay-submission rule: Please contact TA to submit the HW1 files within three days overdue. The score will be discounted 30% off for each day. If you exceed three days, you will get 0 point.**
8. If you have any questions, please feel free to ask them on the discussion forum HW1. TA won't reply to any homework-related questions sent directly through email. If you still got any further questions, please contact shchang.ee09@nycu.edu.tw