Programming Assignment #1

Binary Decision Diagram (BDD)

Objective

1. To understand how to use CUDD package.
2. To exercise the concept of binary decision diagram.
3. To learn how to visualize BDDs.

Problem 1

Check whether two Boolean equations are equivalent.

Input

Input file includes two functions specified in the following format:

Boolean equation 1.
Boolean equation 2.

The first line describes the first Boolean equation, while the second line describes the second Boolean equation. Each equation ends up with a period and every variable is represented by exactly one character (i.e., 26 variables at most). The Boolean equation is given in sum-of-product (SOP) form: lowercase character represents a plain variable, whereas its uppercase counterpart is for its complement.

Input file example

```
ABcD+ABCD+aBcD+aBCD.
BD.
```

Output

Output 1 if the given two Boolean equations are equivalent; otherwise output 0.

```
1           // if the given two equations are equivalent.
```

Problem 2

Construct a BDD with given variable ordering.

Input

Input file is a node list of following format:

```
Boolean equation.
Variable ordering 1.
Variable ordering 2.
..
Variable ordering n.
```

The first line specifies the Boolean equation, while the following lines give the various variable orderings. Each equation ends up with a period and every variable is represented by exactly one character (i.e., 26 variables at most). The Boolean equation is given in sum-of-product (SOP) form: lowercase character represents a plain variable, whereas its uppercase counterpart is for its complement.

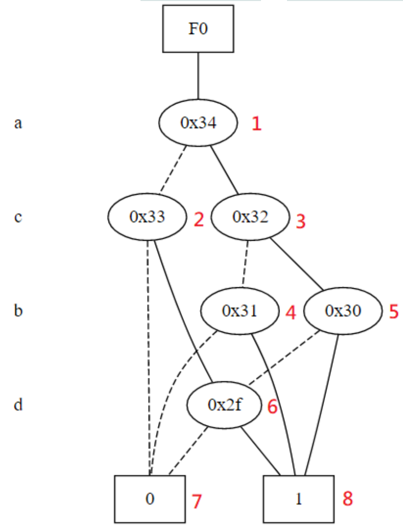
Input example

```
ab+cd.
acbd.      // First is variable 'a', then 'c' …
```

Output

Output: Number of nodes required to represent the given BDD.

```
8          // 8 nodes are required in this BDD, as the following figure shows.
```



Problem 3

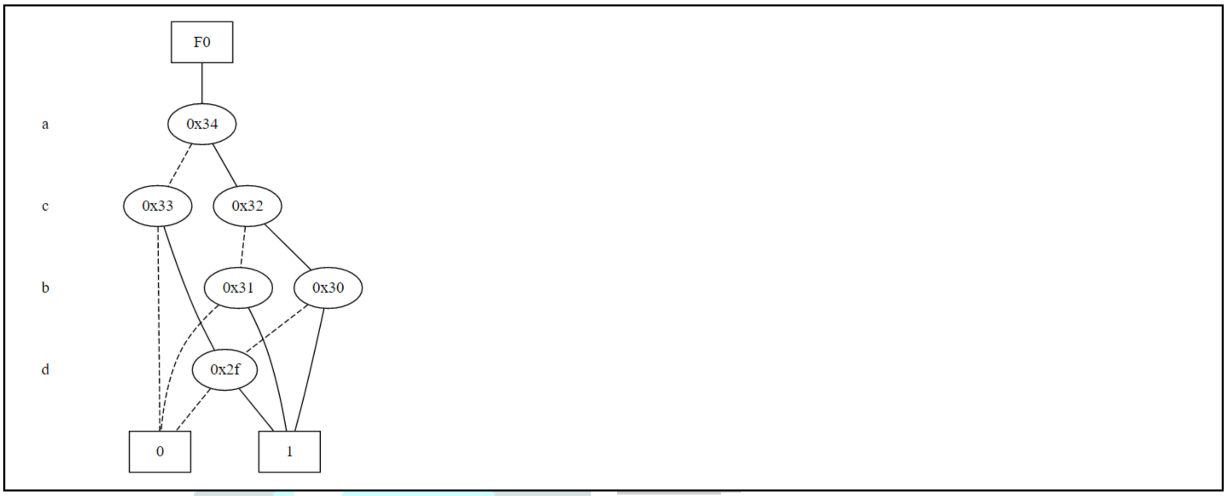
Visualize the result of your CUDD program. You need to use function ‘Cudd_DumpDot’ to produce an associated .dot file, then use the tool ‘Graphviz’ to generate and visualize the graph.

Input

Input file is same as Problem 2.

Output

Output: A graph of given function and ordering.



Compile & Execute

Compile command : **$ show in installation PDF.**
Execute command : **$ show in installation PDF.**
Note that input and output file should be the arguments of program. Please make sure your code can be compiled and executed.

Program Submission

1. Please use the C language and your program must be written in only four source files.
2. Your source file must be named as
   "Student_ID_number_hw1_1.c",
   "Student_ID_number_hw1_2.c",
   "Student_ID_number_hw1_3.c"
   and please make sure that all characters of the filename are in lower case. For example, if your student number is 9711592, the name of your program file should be
   "9711592_hw1_1.c",
   "9711592_hw1_2.c",
   "9711592_hw1_3.c".
3. Upload your report and program to the E3 platform by the deadline.
4. Don't print any words on the terminal.

Report

1. Your report must contain:

   a. The graph of problem 3.

   b. Difference between simulated annealing and genetic algorithm.

   The report file name must be "Student_ID_number_hw1.doc(x)" or "Student_ID_number_hw1.pdf" and please make sure that all characters of the filename are in lower case. For example, if your student number is 9711592, the name of your program file should be "9711592_hw1.pdf".

Grading

• **Report**	**20%**
• **Problem 1**	**30%**
■ **Case1**	**5%**
■ **Case2**	**5%**
■ **Case3 (hidden)**	**5%**
■ **Case4 (hidden)**	**5%**
■ **Case5 (hidden)**	**5%**
■ **Case6 (hidden)**	**5%**
• **Problem 2**	**30%**
■ **Case1**	**5%**
■ **Case2**	**5%**
■ **Case3 (hidden)**	**10%**
■ **Case4 (hidden)**	**10%**
• **Problem 3**	**20%**
■ **Case1**	**5%**
■ **Case2**	**5%**
■ **Case3**	**5%**
■ **Case4**	**5%**

* Time limit is 300s.

Notices

- **Due Date : 2020/04/05, 23:55:00**
- **You'll get 0 grade if failing to hand in on time.**
- **Plagiarism is strictly forbidden. 0 grade guarantee!**