ACE Coursework 2

1. Problem Specification

Every formula α of propositional logic can be converted into another formula α' such that α and α' are logically equivalent, and where α' is a disjunction of conjunctions of literals, where a literal is either an atom or its negation. We say that α' is in disjunctive normal form (DNF). For example, in the propositional case, a DNF formula looks like this:

$$(p \wedge \neg q) \vee (q \wedge r \wedge \neg s \wedge p) \vee (\neg r \wedge q)$$

Let A be a finite set of individual names. The syntax of the logic of direction (LD) is defined as

$$\phi, \psi \coloneqq E(a,b) \mid W(a,b) \mid I_{ew}(a,b) \mid N(a,b) \mid S(a,b) \mid I_{ns}(a,b) \mid \neg\phi \mid \phi \wedge \psi$$

Where $a, b \in A$, $\phi \vee \psi = \neg(\neg\phi \wedge \neg\psi)$, $\phi \rightarrow \psi = \neg(\phi \wedge \neg\psi), \phi \leftrightarrow \psi = (\phi \rightarrow \psi) \wedge (\psi \rightarrow \phi), \bot = \phi \wedge \neg\phi, \top = \phi \vee \neg\phi$.

Write a Java program that takes an LD formula α, which does not involves $\leftrightarrow$, as input, and outputs another formula α' such that α and α' are logically equivalent and α' is in DNF.

2. Input and Output

The inputs and outputs are both Java Strings. The following table shows how to represent each logical operator as a String in Java.

Standard logical operator	Corresponding symbol as a String in Java
$\neg$	~
$\wedge$	&
$\vee$	|
$\rightarrow$	->

We use “T” and “F” to represent $\top$ and $\bot$ respectively.

To simplify the problem, we define the input format of an LD formula as follows:

A formula is an atomic formula or a complex formula.

An atomic formula is of one of the forms:

$T, F, E(a,b), W(a,b), Iew(a,b), N(a,b), S(a,b), Ins(a,b)$.

A complex formula is of one of the forms:

- ~ [$formula$]
- [$formula$] & [$formula$]
- [$formula$] | [$formula$]
- [$formula$] -> [$formula$]

Sample inputs and outputs are provided as follows.

Input	Output
~ [[E(a,b)] & [W(b,c)]]	~ [E(a,b)] | ~ [W(b,c)]
[E(a,b)] -> [W(b,c)]	~ [E(a, b)] | [W(b,c)]
[E(a,b)] & [~ [E(a,b)]]	F
[E(a,b)] | [~ [E(a,b)]]	T

For more examples, see examples.txt.

3. A Standard Procedure

For your reference, a standard procedure to convert an LD formula, which does not involve $\leftrightarrow$, into DNF is as follows. You may follow this procedure, or follow any other procedure, as long as it works correctly.

(1) Eliminate $\rightarrow$, using the following equivalence: $\phi \rightarrow \psi \equiv \neg\phi \vee \psi$

(2) Move $\neg$ inward so that it appears only in front of an atom, using the following equivalences:

- $\neg\neg\phi \equiv \phi$
- $\neg(\phi \wedge \psi) \equiv (\neg\phi \vee \neg\psi)$
- $\neg(\phi \vee \psi) \equiv (\neg\phi \wedge \neg\psi)$

(3) Distribute $\vee$ over $\wedge$, using the following equivalences:

- $\big(\phi \wedge (\psi \vee \eta)\big) \equiv (\phi \wedge \psi) \vee (\phi \wedge \eta)$
- $\big((\phi \vee \psi) \wedge \eta\big) \equiv (\phi \wedge \eta) \vee (\psi \wedge \eta)$

(4) Collect terms, using the following equivalences:

- $(\phi \vee \phi) \equiv \phi$
- $(\phi \wedge \phi) \equiv \phi$
- $(\phi \vee \neg\phi) \equiv \top$
- $(\phi \wedge \neg\phi) \equiv \bot$
- $\neg(\top) \equiv \bot$
- $\neg(\bot) \equiv \top$
- $\top \vee \phi \equiv \phi \vee \top \equiv \top$
- $\top \wedge \phi \equiv \phi \wedge \top \equiv \phi$
- $\bot \vee \phi \equiv \phi \vee \bot \equiv \phi$
- $\bot \wedge \phi \equiv \phi \wedge \bot \equiv \bot$

4. Coursework Submission

Please submit a zip file which consists of:

- Java program files;
- a “read me” file describing the structure of your Java program and how to run your program;

- a report (in PDF format) *briefly* describing how you solved the given problem, including which data structure you used and why, the algorithms designed and implemented, and why your program works correctly, time complexity of the program, the testing of your program, etc. Keep the report short and brief, within 2-3 pages.

Keep your code and report brief and clear! Good luck! ☺