Middle East Technical University
Department of Computer Engineering

CENG 140

Spring '22

Take Home Exam 2

REGULATIONS

Due date: 10 June 2022, Friday *(Not subject to postpone)*

Submission: Electronically. You will be submitting your program source code written in a file which you will name as `the2.c` through the metuclass system (follow instruction in Sec All group). Resubmission is allowed (till the last moment of the due date), The last will replace the previous.

Team: There is no teaming up. The take home exam has to be done/turned in individually.

Cheating: **This is an exam.** All parts involved (source(s) and receiver(s)) get zero+parts will be subject to disciplinary action. Using code from WEB is also considered as "cheating".

I/O Specification: You are expected to write a program that 100% complies with the I/O specifications expressed in this sheet. Do not beautify you I/O.

INTRODUCTION

This time we will be dealing with so called **Well-Formed Propositional Logic Formulas**. From now on We will abbreviate it by "WFPLF". The syntax of a WFPLF is defined as follows:

⟨WFPLF⟩ ::= ⟨letter⟩ | `-` ⟨WFPLF⟩ | ⟨binop⟩
⟨binop⟩ ::= `(` ⟨WFPLF⟩ ⟨opr⟩ ⟨WFPLF⟩ `)`
⟨opr⟩ ::= `&` | `|` | `>` | `=`
⟨letter⟩ ::= `a` | `b` | ... | `z`

The semantics is as follows:

An *instance* of a WFPLF is defined to be a state where all letters (variables) used in that formula have a value. The letter can admit the values `T` and `F`, semanticly standing for *Truthness* and *Falsity*. The values associated with the compound WFPLF formulas that involve the operators '`-`', '`&`', '`|`', '`>`', '`=`' are defined by the so called *Truth tables* (the table where all instances are covered).

□	△	-□	(□&△)	(□|△)	(□>△)	(□=△)
T	T	F	T	T	T	T
T	F	F	F	T	F	F
F	T	T	F	T	T	F
F	F	T	F	F	T	T

PROBLEM

A program that you will write will take any WFPLF from the standard input and produce the truth table on the standard output.

SPECIFICATIONS

- The length of the formulas are not restricted. That means you shall not make any restrictive assumption about a maximal length. Of course there will be *natural* restrictions imposed by the compiler and hardware *(like pointer size, integer size, memory size, etc.)*. But you shall not impose additional restrictions.
- Though the alphabet is limited to the English lowercase characters it is possible to have formulas in which the same letter occurs at different positions.
- All blanks and end-of-lines are to be neglected, simply skip them.
- Parenthesis are not optional nor neutral. That means closing a WFPLF in an parenthesis is forbidden; negation does not have a parenthesis of its argument (doing so is wrong).
- Since, at the moment you receive this worksheet the `struct` and `union` declarations are not introduced, don't build tree structures and do not use struct/union types in your program.
- You will keep the formula as a string, and for each instance reevaluate it. Do not modify the string (or a copy of it) from instance to instance. We will check for this at evaluation.
- Do not use Dijkstra's algorithm. But make use of *recursion* in your parser/evaluator.
  For your information: Though there is no restriction on the code size, the whole implementation done by us, with nice coding, indentation etc, is 135 lines.
- Your first line of your output shall contain the used characters in the lexicographical order (i.e. form `a` to `z`), separated by exactly one blank. And the upper case letter `R` as the last character in the line (which stands for 'result'). Do not use any other character than `R`.

  The following lines will contain the truth values represented by the letters `T` and `F` (Do NOT use anything else like, `t`,`f`, `True`, `TRUE`, `1`, `0`, etc.)

  You are expected to start with the all variables `T` case as the first line of the truth table and then continue by changing the right-most fastest and the left-most slowest.
- You can assume that all tests and runs are error-free and complies with the above given syntax. Therefore do not perform any error check on the input.
- **Hints:** You may consider using the functions `realloc()`, `getchar()`, `islower()`, `isspace()`. Also you may have a look at "COURSE WEEK 9 LECTURE VIDEOS > String Exercises > Part 3+4" reachable under metuclass announcements.

EXAMPLE

```
     -(-(a&  k) > ( -((a|-k)
              | c   )|           d))
```

is a possible input. The expected output is:

```
a c d k R
T T T T F
T T T F F
T T F T F
T T F F T
T F T T F
T F T F F
T F F T F
T F F F T
F T T T F
F T T F F
F T F T T
F T F F T
F F T T F
F F T F F
F F F T F
F F F F T
```