**Programming Project 1: Boolean Expression Evaluator**

You will be writing a C++ program that reads a string, determines whether it is a syntactically correct Boolean expression, and, if so, determines its value. For a grade of 100, This program will be due on Thursday, Sept 28. Each week or part of a week thereafter counts off 10%. The program must pass all of the test cases created by the grader to receive credit. Programs that do not pass all test cases may be resubmitted the following week.

**1. Data model**

A ***symbol*** is one of the following nine strings: "T", "F", "^", "v", "~", "=>", "<=>", "(", ")". Intuitively, these are interpreted as *true*, *false,* *and*, *or*, *not*, *implies*, *if-and-only-if*, and left and right parentheses.

A ***symbol* *string***is the concatenation of zero or more symbols and/or spaces. For example, the following are symbol strings:

* "T "
* "=> Tv~ F(( F"
* "T => F ^ (F v F)"
* ""
* " "

and the following are not symbols strings

* "T X "
* " F= > T"
* "TF p\*^)"

A ***Boolean******expression*** is a C++ vector of strings satisfying certain conditions. We will write C++ vectors by writing their elements, separated in commas and enclosed in brackets; for example, we will write [10 ,20] for the vector of length 2 whose first element is 10 and whose second element is 20.. "Followed by" means *concatenated with.* For example, ["F"] followed by ["^", "T"] is the vector ["F","^","T]. The rules for forming Boolean expressions of various sorts are given below.

1. A ***Boolean constant*** is ["T"] or ["F"].
2. An ***unbreakable******expression*** is either a Boolean constant, or ["("] followed by a Boolean expression followed by [")"].
3. A ***negation*** is either an unbreakable expression, or ["~"] followed by a negation.
4. A ***conjunction***is either a negation, or a conjunction followed by ["^"] followed by a negation.
5. A ***disjunction*** is either a conjunction, or a disjunction followed by ["v"] followed by a conjunction.
6. An ***implication*** is either a disjunction, or a disjunction followed by ["=>"] followed by an implication.
7. A ***Boolean expression*** is either an implication, or an implication followed by ["<=>"] followed by a Boolean expression.

This grammar can be formalized in [BNF notation](https://en.wikipedia.org/wiki/Backus%E2%80%93Naur_form) as follows:

*Const* → "T" | "F"

*U*  → *Const* | "(" B ")" *// note, this rule has been corrected*

*N*  → *U* | "~" *N*

*C* → *N* | *C* "^" *N*

*D*  → *C* | *D* "v" *C // note, this rule has been corrected*

*I* → *D* | *D* "=>" *I*

*B*  → *I* | *I* "<=>" *B*

An *AST* (short for *abstract syntax tree,* the standard name for what the book calls an *expression tree*) is defined below. This follows the pattern for defining trees given in the Aho & Ullman book in Chapter 5, p. 232. We will use AST's as a data structure to store the semantic structure of Boolean expressions.

typedef struct AST\* pNODE;

struct AST {string info; pNODE children[2];};

The *info* member of an AST is a symbol, as defined above. If *info* is "T" or "F" then both children are NULL. If *inf*o is "~" then children[1] is NULL. Otherwise, both children are non-NULL. Sample code illustrating the use of this data structure can be found [here](https://docs.google.com/document/d/1lUo6rEdS_UfXR6bDYdTO0DCbZhdGQsibElnEvlDHz-k/edit?usp=sharing).

A *tokRslt i*s a struct with two fields:

* *success*, a bool
* *syms*, a C++ vector of strings.

A *parseRslt* is a struct with two fields:

* *success*, a bool
* *ast*, an AST

A TPERslt is a struct with two fields:

* *val,* a bool
* *msg,* a string

**2. Functions**

Implement the following five functions in a single C++ file:

tokRslt tokenize(string s)

1. If *s* is a string, *tokenize*(*s*).*success* is *true* if *s* is a string of symbols, and *false* otherwise.
2. If *s* is a string of symbols, then *tokenize*(*s*).*syms* is a vector of the symbols occurring in *s*, in order. For example, if *s* = "T vv =>" then *tokenize*(*s*).*syms* = ["T","v","v","=>"]

parseRslt parse(vector<string> V)

1. If *V* is a Boolean expression, then *parse*(V).*success* is *true* and *parse*(*V*).*ast* is the abstract syntax tree of *V* according to the [standard grammar of Boolean expressions.](https://docs.google.com/document/d/1r_5b5_EkDedJ3eF6BKkhVOJbyDZ9SVawaSEHAvmN94A/edit?usp=sharing)
2. Otherwise, *parse*(V).*success* if *false*.

bool eval(AST T)

1. *eval*(*T*) is the value of *T* according to the [standard semantics of Boolean expressions](https://docs.google.com/document/d/1r_5b5_EkDedJ3eF6BKkhVOJbyDZ9SVawaSEHAvmN94A/edit?usp=sharing).

TPERslt TPE(string s) (*tokenize, parse,* and *evaluate*)

1. If s is a string of symbols whose tokenization is a Boolean expression, then *TPE.msg* is "success" and *TPE*(*s*).*val* is the value of that Boolean expression.
2. If s is a string of symbols whose tokenization is not a Boolean expression, then *TPE.msg* is "grammar error".
3. If *s* is not a string of symbols, then *TPE.msg* is "symbol error".

string TPEOut(string s)

1. If *s* is a string of symbols whose tokenization is a Boolean expression, then *TPEOut*(s)is the value of that expression, converted to a string, which is either "true" or "false".
2. If *s* is a string of symbols whose tokenization is not a Boolean expression, then *TPEOut*(s) is "grammar error".
3. If s is not a string of symbols, then *TPEOut*(*s*) is "symbol error".

For example,

* If *s* = "T v F ^ T", then *TPEOut*(s) is "true"
* If *s* = "T => (F X T", then *TPEOut(s*) is "symbol error"
* If *s* = "T T (F & T => F)", then *TPEOut*(*s*) is "grammar error"

**Academic Integrity**

Students are allowed to discuss this assignment verbally with each other, search the Web for useful source code, and download and use code if it helps you (though I doubt one can find code that will use the same data structures we use). The following are prohibited:

1. Looking at another student's code
2. Sharing code with another student
3. Asking another person to write code for you or to see their code.

Violations of 1-3 will result in a request to withdraw from the class with a *W*, if done before the last day to drop. After this date, it will result in an *F*. If those options are not acceptable, the result will be a university-level academic honesty proceeding, with the aim of expulsion from the university.