Compiler Construction
Phase 1

You are required to perform the following task over the definition of our compiler's words.

1. Provide regular definitions for each category/class. *[Assignment 1 Part]*
2. Provide a single transition diagram that will process words from all the categories and classes of our compiler. *[Assignment 1 Part]*
3. Implement the transition diagram using the dynamic method (table-driven method). *[Project 1]*

Identifier ⇒ It starts with an underscore or letter but contains at least one underscore.

Number ⇒ It contains signed and unsigned numbers along with floating points and exponential.

Punctuations ⇒ [, { , (,) , } ,], ::

Operators ⇒ < , > , <>, :=, ==, + , -, ++, +=, <=, >=, %, ||, &&, !=, *, " , " , /, <<, >>

Keywords ⇒ asm, Wagarna, new, this, auto, enum, operator, throw, Mantiqi, explicit, private, True, break, export, protected, try, case, extern, public, typedef, catch, False, register, typeid, Harf, Ashriya, typename, Adadi , class, for, Wapas, union, const, dost, short, unsigned, goto, signed, using, continue, Agar, sizeof, virtual, default, inline, static, Khali, delete, volatile, do, long, struct, double, mutable, switch, while, namespace, template, Marqazi, Matn, input->, output<-

Sample Code for understanding the words of our compiler:

```
Adadi Marqazi ()
{
Adadi _num = 10 ::
Ashriya num_ = -10.5E+12 ::
Ashriya n_umber = +9.3 ::
Harf cou_se = "c" ::
Matn course_ = "Compiler Construction" ::
Mantiqi _flag = True ::
Agar ( _flag )
output<- "Ok" ::
Wagarna
input-> _num ::
Wapas 0 ::
}
```

Assignment Description:

For this task, you have to implement a **lexical analyzer**, also called a scanner. This assignment includes the following parts:

	PARTS	Output	Marks
1	Generate FA for each regular definition	Document (HandWritten)	~2-3 Abs
2	Code - Table-Driven Approach	Source Code Files + Demo Video	~ 4-6 Abs

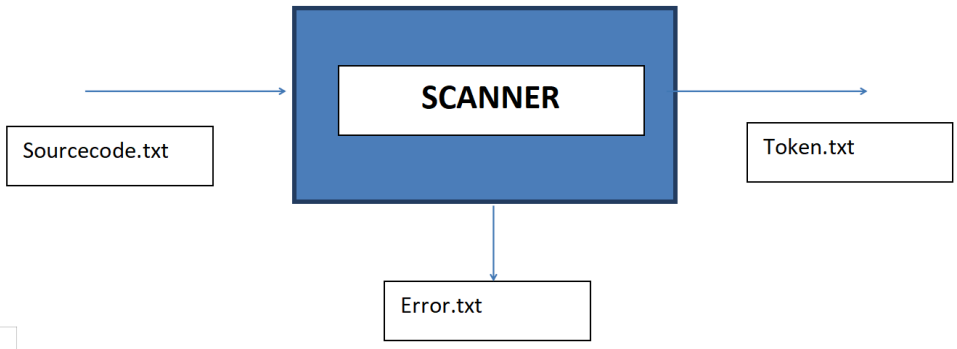
Tools:

Language (For Development): C++

Note: Student cannot use built-in data structure. Student can use his own data structure, Hash Table, Linked List which he/she developed in data structure course.

Evaluating Criteria:

1. The source code should accurately reflect the details specified in related documents.
2. A text file containing valid source code will serve as the input for the scanner, and the output will be a token file.
3. The assignment must incorporate all points discussed in class regarding scanner implementation.
4. The application should strictly adhere to its intended functionality and avoid performing unintended operations.
5. If the input source code is invalid, the tool must generate an error list.



GOOD LUCK