Welcome to the compiler project!

In this project, you are supposed to design a compiler for an Object Oriented Language called Cool. Cool stands for ***Classroom Object Oriented Language*** and is designed to be implemented with reasonable effort in a one-semester course. During this project, we will cover most (not all!) of its features.

The whole project consists of three main phases:

1- Scanner   2- Parser   3- Code Generator

The output of the previous phase – with little change – is the input for the current phase. The final program containing all phases must be able to get a Cool program and generate codes ready to be run. Notice that if you do not implement a phase properly with minimum requirements, the next phase will be in great chaos.

Phase 1

In this phase, only the scanner is required, meaning that your program should be capable of getting a stream of characters (not necessarily a Cool program) and break it to the tokens. Your scanner must not consider the whitespaces out of string constants and should distinguish keywords, integer/real/string/boolean constants, operators and identifiers. Furthermore, it must just ignore the token if not in the sets above.

Token	Format
Reserved Keywords	**Bold and Blue**
Identifiers	Violate
Integer Numbers	Orange
Real Numbers	*Italic and Orange*
Strings and Characters	Green
Special Characters (both in string or character)	*Italic and Green*
Comments	Yellow
Operators and Punctuations	Black
Undefined Token	Red

❖ Reserved Keywords

void	int	real	bool	string	class
for	while	if	else	return	break
rof	let	fi	Array	void	in_string
out_string	new	break	continue	loop	pool
in_int	out_int	then	len		

❖ Identifier

A sequence of letters, digits and underline starting with a letter. Cool is case sensitive. The identifiers can be at most 31 characters long.

❖ Numbers

The specification of the numbers are described below.

Type	Description
Decimal Integer	A sequence of digits from 0-9. Example: 1642, 134
Hexadecimal	A sequence of digits and characters from A/a to F/f staring with 0X/0x. Example: 0x0, 0X12aE
Real Numbers	A sequence of digits having a '.' in between. Note that there can be no digit after '.' Example: 0.12 ✓, 12. ✓, .12 ×
Scientific Notation	A real number following an 'E' and an integer with an optional plus or minus sign. Examples: 12.2E+2 ✓, 12.E2 ✓, 1.2E-1 ✓, .12E3 ×

❖ Strings

Like C language, strings start and end with " and ends with ". Special characters are the ones starting with \ like \t.

❖ Comments

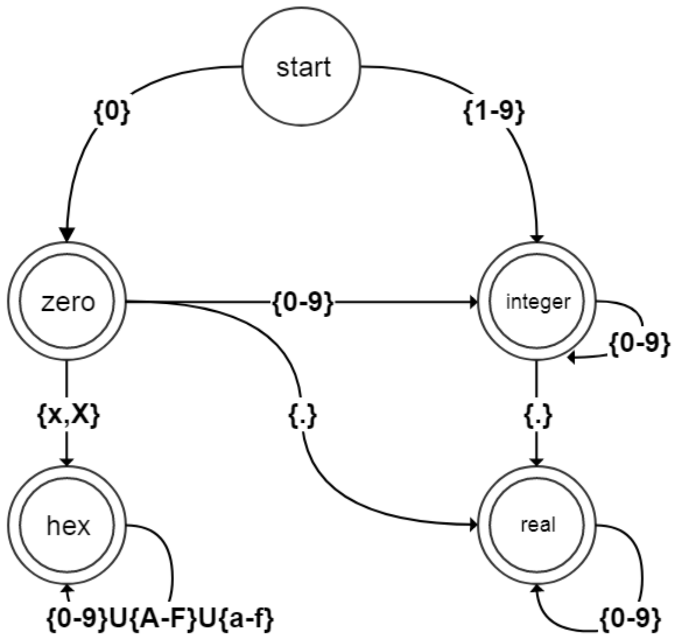
Comments have two types: ones that start with // and ones that start with /* and end with */ and can span in multiple lines.

- Operators and Punctuations

  The language also contains following symbols which must have black color.

Description	Symbol	Description	Symbol
add	+	unary minus	-
production	*	division	/
addition assignment	+=	subtraction assignment	-=
production assignment	*=	division assignment	/=
increment	++	decrement	--
less	<	less equal	<=
greater	>	greater equal	>=
not equal	!=	equal	==
assignment	<-	mod	%
logical and	&&	logical or	| |
bitwise and	&	bitwise or	|
string literal	“	bitwise xor	^
not	!	dot	.
colon	,	semicolon	;
opening braces	[	closing braces	]
opening parenthesis	(	closing parenthesis	)
opening curly braces	{	closing curly braces	}

Part of Scanner Graph



Notes

- The due date is Farvardin 31th
- Your program must output an HTML file that highlights text based on rules described above.
- What you must upload is a .zip file containing your program, and a .pdf report file explaining what you have done.
- This phase of the project can be done in groups of two.
- In case of using any resources, mention them in your report file.